

---

**Roundtable Total Software Management System**

# **Roundtable User's Guide**

**Character Edition**

**10.1A**

**Tugboat Software**

---

Published 2006-07-03 11:17:39

Copyright © 2006 Ledbetter & Harp LLC

Roundtable® software products are licensed by Tugboat Software Inc. and copyrighted by Ledbetter & Harp LLC, with all rights reserved. This manual is also copyrighted and all rights are reserved. This manual may not, in whole or in part, be copied, photocopied, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from Tugboat Software or Ledbetter & Harp LLC.

The information in this document is subject to change without notice, and neither Tugboat Software, nor Ledbetter & Harp LLC assume responsibility for any errors that may appear in this document.

Printed in U.S.A.

Roundtable® is a registered trademark of Ledbetter & Harp LLC.

Microsoft Windows® is copyrighted by Microsoft Corporation. Microsoft® is a registered trademark of Microsoft Corporation. Windows™ is a trademark of Microsoft Corporation.

Progress® is a registered trademark of Progress Software Corporation.

Unix® is a trademark registered in the United States and other countries, licensed exclusively through X/Open Company, Ltd.

All company and product names are the trademarks or registered trademarks of their respective companies.

Tugboat Software Inc.  
20301 Birch Street, Suite 202  
Newport Beach, CA 92660-3122

---

---

# Contents

---

<b>Preface</b> .....	xiii
1. Purpose .....	xiii
2. Audience .....	xiii
3. Organization of This Guide .....	xiii
4. Typographical Conventions .....	xiv
<b>Software Configuration Management</b> .....	1-1
1.1. Introduction .....	1-1
1.2. Exploiting the Opportunity .....	1-2
1.3. Principal SCM Activities .....	1-2
1.3.1. Configuration Identification .....	1-3
1.3.2. Configuration Control .....	1-12
1.3.3. Configuration Auditing .....	1-15
1.4. Configuration Status Accounting .....	1-18
1.5. SCM Information Flow .....	1-19
1.6. Benefits of SCM .....	1-19
1.7. Workspace Networks .....	1-20
1.8. SCM for Product Releases .....	1-20
1.8.1. SCM for Incremental Deployment .....	1-22
1.9. SCM for Custom Variant Deployment .....	1-24
1.10. Distributed Development .....	1-26
1.10.1. Sites and Ownership .....	1-27
1.10.2. Deployments and Partner Site Loads .....	1-27
1.10.3. Receipt Workspace .....	1-28
1.10.4. Customers Doing Development .....	1-28
1.10.5. Custom Product Modules .....	1-30
<b>Navigating Roundtable</b> .....	2-1
2.1. Introduction .....	2-1
2.2. Roundtable Screens, Keys, and Menus .....	2-1
2.3. Tour of Roundtable Views .....	2-3
2.3.1. Tasks .....	2-3
2.3.2. Workspace Selection .....	2-3

2.3.3. Module Selection .....	2-3
2.3.4. Object Selection .....	2-4
2.3.5. Xref .....	2-4
2.3.6. Where Used .....	2-4
2.3.7. Change History .....	2-4
<b>Roundtable Administration .....</b>	<b>3-1</b>
3.1. Introduction .....	3-1
3.2. Starting Roundtable .....	3-1
3.3. Exiting Roundtable .....	3-2
3.4. RTBSETUP File Options .....	3-2
3.5. Setting Up Roundtable Users .....	3-7
3.6. Defining a Configuration Hierarchy .....	3-8
3.6.1. Application Group Directories .....	3-9
3.6.2. Directories by Component Type .....	3-10
3.6.3. Directories by Application Group and Component Type .....	3-10
3.6.4. Using Subtypes to Extend the Configuration Hierarchy .....	3-12
3.7. Products .....	3-13
3.7.1. Product Modules Menu .....	3-14
3.7.2. Adding a Product .....	3-15
3.7.3. Editing a Product Name .....	3-16
3.7.4. Deleting a Product .....	3-16
3.7.5. Product Report .....	3-17
3.7.6. Adding a Product Module .....	3-17
3.7.7. Editing a Product Module .....	3-18
3.7.8. Deleting a Product Module .....	3-19
3.7.9. Product Module Report .....	3-19
3.8. Workspace Module Definitions .....	3-20
3.8.1. Adding a Workspace Module Definition .....	3-21
3.8.2. Editing a Workspace Module Definition .....	3-21
3.8.3. Deleting a Workspace Module Definition .....	3-21
3.9. Subtypes .....	3-21
3.9.1. PCODE Subtype Menu .....	3-22
3.9.2. Subtype Theory .....	3-24
3.9.3. Subtype Example — Part I .....	3-24
3.9.4. Subtype Example — Part II .....	3-25
3.9.5. Subtype Example — Part III .....	3-26
3.9.6. Adding a Subtype .....	3-26
3.9.7. Editing a Subtype or Subtype Part .....	3-27
3.9.8. Deleting a Subtype .....	3-28
3.9.9. Sub-types Report .....	3-28
3.9.10. Build Program .....	3-29
3.9.11. Edit Program .....	3-30
3.9.12. System Parameters .....	3-33
3.10. Security .....	3-34
3.10.1. User Maintenance Screen Description .....	3-35
3.10.2. Adding, Editing, or Deleting a User .....	3-35
3.10.3. Changing Your Password .....	3-36
3.10.4. Group Access Screen Description .....	3-36

3.10.5. Adding, Editing, or Deleting a Group Access Code .....	3-38
3.10.6. Workspace Security Screen Description .....	3-38
3.10.7. Adding Workspace User Access Assignments .....	3-39
3.10.8. Editing User Access for Workspace Assignments .....	3-39
3.10.9. Deleting User Access for Workspace Assignment .....	3-39
3.10.10. Viewing Current Privileges .....	3-40
3.10.11. Sysop Exclusive Access .....	3-40
3.10.12. Repository Dump and Load .....	3-41
3.10.13. Defining Your Application .....	3-41
3.11. Loading Multiple Workspaces .....	3-42
3.11.1. Pre-Prod .....	3-43
3.11.2. Test .....	3-43
3.11.3. Devel .....	3-44
3.11.4. Setting Up the Workspace Flow .....	3-44
<b>Workspaces .....</b>	<b>4-1</b>
4.1. Introduction .....	4-1
4.2. Workspace Maintenance .....	4-1
4.2.1. Guidelines for Workspace Planning .....	4-2
4.2.2. Workspace Window Description .....	4-3
4.2.3. Adding a Workspace .....	4-3
4.2.4. Editing a Workspace .....	4-4
4.2.5. Deleting a Workspace .....	4-5
4.2.6. Viewing Workspace Event History .....	4-6
4.2.7. Virtual "Read-Only" Workspace .....	4-7
4.2.8. Hidden Workspace .....	4-7
4.2.9. Archived Workspace .....	4-7
4.2.10. Workspace Reports .....	4-7
4.2.11. Printing Reports .....	4-8
4.2.12. Print Destination Dialog Box .....	4-8
4.2.13. Workspace Report .....	4-9
4.2.14. Workspace Event History Report .....	4-9
4.2.15. Release Report .....	4-10
4.2.16. Changes Report .....	4-10
4.3. Workspace Sources .....	4-11
4.3.1. Workspace Sources Screen Description .....	4-13
4.3.2. Adding a Workspace Source .....	4-13
4.3.3. Deleting a Workspace Source .....	4-14
4.3.4. Editing Workspace Module Parameters .....	4-14
4.4. Object Variants .....	4-15
4.4.1. Object Repository .....	4-16
4.4.2. Versions Versus Variants .....	4-16
4.4.3. Example of Object Variations .....	4-17
4.5. Database Schema Updates .....	4-17
4.5.1. Database Integrity Check Report .....	4-18
4.5.2. What to Do when the Physical Schema Is Out of Synchronization .....	4-19
4.5.3. Schema Update Window Description .....	4-19
4.5.4. Unapplied Changes Report .....	4-19

4.5.5. Building the Schema Update List .....	4-20
4.5.6. Database Storage Areas .....	4-21
4.5.7. Deactivate Flags .....	4-21
4.5.8. Applying a Table as "New" .....	4-22
4.5.9. Skipping a List Item .....	4-22
4.5.10. Data Procedures .....	4-22
4.5.11. Creating Data Procedures .....	4-23
4.5.12. Updating Physical Schema .....	4-24
4.5.13. Deleting the Schema Update List .....	4-24
4.6. Releases .....	4-24
4.6.1. Releases Window Description .....	4-25
4.6.2. Adding a Release .....	4-26
4.6.3. Editing a Release .....	4-26
4.6.4. Deleting a Release .....	4-27
4.6.5. Finding a Release .....	4-27
4.6.6. Release Report .....	4-27
4.7. Imports .....	4-28
4.7.1. Building the Import Control Table .....	4-29
4.7.2. Import Analysis Report .....	4-30
4.7.3. Version Notes Report .....	4-30
4.7.4. Compare Source Report .....	4-31
4.7.5. Toggling the Import Status .....	4-31
4.7.6. Import .....	4-31
4.7.7. PCODE Data Imports .....	4-32
4.7.8. Deleting the Import Control Table .....	4-32
4.8. Workspace Populate Process .....	4-32
4.9. Build Names Table .....	4-33
4.10. Schema Xref Build .....	4-34
4.11. Deployments .....	4-34
4.11.1. Sites & Deployments Descriptions .....	4-35
4.11.2. Deployments Menu .....	4-36
4.11.3. Procedure Updates and Compiles on Remote Sites .....	4-47
4.12. Database Schema Updates on Remote Sites .....	4-47
4.12.1. Schema Update Process .....	4-48
4.12.2. Creating and Editing the schema.pf File .....	4-49
4.12.3. Schema Release Rules .....	4-50
4.12.4. Updating the Database Schemas Dialog Box .....	4-51
4.12.5. Database Copies .....	4-52
4.12.6. Deploying Both a Full and Incremental Schema Update ....	4-53
<b>Task Management .....</b>	<b>5-1</b>
5.1. Introduction .....	5-1
5.2. What is Task Management? .....	5-1
5.2.1. Benefits of Task Management .....	5-1
5.2.2. How Objects Relate to Task Management .....	5-2
5.3. Task Maintenance .....	5-2
5.3.1. The Task Maintenance by Workspace Menu .....	5-2
5.3.2. Task Selection By Programmer Menu .....	5-3
5.4. Task Activities .....	5-11



5.4.1. Checking Out an Object .....	5-11
5.4.2. Creating an Object Variant .....	5-12
5.4.3. Checking in an Object .....	5-12
5.4.4. Listing Objects in a Task .....	5-13
5.4.5. Changing the Share Status of an Object .....	5-14
5.4.6. Changing the Share Status of All Objects in a Task .....	5-14
5.4.7. Promoting Task Objects .....	5-14
5.4.8. Finding a Task .....	5-15
5.4.9. Selecting No Current Task .....	5-15
5.4.10. Building a Runtask.p Procedure .....	5-15
5.5. Task Groups Maintenance .....	5-16
5.5.1. Task Groups Window Description .....	5-16
5.5.2. Adding a Task Group .....	5-17
5.5.3. Editing a Task Group Directory .....	5-17
5.5.4. Deleting a Task Group .....	5-18
5.5.5. Adding a Task Group Assignment .....	5-18
5.5.6. Deleting a Task Group Assignment .....	5-19
5.6. Task Notations .....	5-19
<b>Objects .....</b>	<b>6-1</b>
6.1. Introduction .....	6-1
6.2. The Repository .....	6-1
6.2.1. Object Versioning and Tasks .....	6-2
6.3. Workspace Objects Menu .....	6-2
6.3.1. Object Details Frame .....	6-2
6.4. PCODE Objects .....	6-4
6.4.1. PROGRESS Code Object Menu Description .....	6-5
6.4.2. Adding a PCODE Object .....	6-7
6.4.3. Object Name Aliasing .....	6-8
6.4.4. Editing a PCODE Object's Configuration Fields .....	6-8
6.4.5. Editing a PCODE Object .....	6-9
6.5. DOC Objects .....	6-10
6.5.1. Adding a DOC Object .....	6-11
6.6. Schema Objects .....	6-12
6.6.1. Domains .....	6-13
6.6.2. Schema Object Versions .....	6-14
6.7. PDBASE Objects .....	6-14
6.7.1. PROGRESS Database Object Menu .....	6-15
6.7.2. Adding a PDBASE Object .....	6-17
6.7.3. Editing the PROGRESS Database Object .....	6-17
6.7.4. PDBASE for DataServer Database .....	6-18
6.7.5. DataServer Schema Change Restrictions .....	6-20
6.7.6. Adding a Database Alias .....	6-20
6.7.7. Deleting an Alias .....	6-21
6.7.8. Adding a Database Sequence .....	6-21
6.7.9. Edit a Database Sequence .....	6-22
6.7.10. Delete a Database Sequence .....	6-22
6.7.11. Load PROGRESS Schema .....	6-22
6.7.12. Managing Database Tables .....	6-22

6.7.13. Adding a Table .....	6-22
6.7.14. Table Assignment .....	6-23
6.7.15. Deleting a Table Assignment .....	6-23
6.7.16. PDBASE Object Report .....	6-24
6.7.17. Changing Local Table Name .....	6-24
6.7.18. Database Definition Report .....	6-24
6.7.19. Integrity Check Report .....	6-25
6.7.20. Index Usage Report .....	6-25
6.8. PFILE Objects .....	6-25
6.8.1. PROGRESS File Object Menu .....	6-26
6.8.2. Adding a PFILE Object .....	6-27
6.8.3. Editing the PFILE Object .....	6-27
6.8.4. Field Assignments .....	6-28
6.8.5. Adding a Field to a Table .....	6-29
6.8.6. Assigning a Field .....	6-30
6.8.7. Editing a Field Assignment .....	6-30
6.8.8. Deleting a Field Assignment .....	6-31
6.8.9. Indexes .....	6-31
6.8.10. Select Index Frame .....	6-31
6.8.11. Index Menu .....	6-32
6.8.12. Index Components Menu .....	6-32
6.8.13. Adding an Index .....	6-33
6.8.14. Deleting an Index .....	6-33
6.8.15. Renaming an Index .....	6-33
6.8.16. Editing an Index .....	6-34
6.8.17. Adding Table Triggers .....	6-34
6.8.18. Deleting a Trigger .....	6-35
6.8.19. PFILE Object Report .....	6-35
6.8.20. Table Definition Report .....	6-35
6.9. PFIELD Objects .....	6-36
6.9.1. Field Object Menu Description .....	6-36
6.9.2. Adding a PFIELD Object .....	6-37
6.9.3. Editing a PFIELD Object .....	6-38
6.9.4. Changing the Data Type or Extent of a Field .....	6-38
6.9.5. PFIELD Object Report .....	6-39
6.10. Assigning an Object .....	6-39
6.11. Object Reports .....	6-39
6.11.1. Versions in Product Module Report .....	6-40
6.11.2. Versions in Workspace Report .....	6-41
6.11.3. Program Usage Report .....	6-41
6.11.4. Call Diagram Report .....	6-42
6.11.5. Unused Object Report .....	6-43
6.11.6. External Objects Report .....	6-43
6.11.7. Where Used Report .....	6-43
6.11.8. Repository Check Report .....	6-44
6.11.9. Xrefs Menu .....	6-44
6.11.10. Informal Xrefs Menu .....	6-46
6.11.11. Where Used Menu .....	6-47
6.11.12. History Menu .....	6-49

<b>Tools</b>	7-1
7.1. Introduction	7-1
7.2. Global Change Finder	7-1
7.2.1. Global Change Finder Dialog Box	7-1
7.3. Compiling Tools	7-2
7.3.1. Compiling Rules	7-3
7.3.2. Compile Object Without Xref	7-4
7.3.3. Compile with Xref	7-5
7.3.4. Selective Compile Specifications Screen Description	7-5
7.3.5. Selective Compiles	7-6
7.4. Module Load	7-7
7.4.1. Running Module Load	7-7
7.4.2. Test Version Integrity	7-10
7.5. Source Compare Report	7-10
7.5.1. Comparing Files with the Source Compare Report	7-11
7.5.2. Partner Site Load	7-11
7.6. Load Schema	7-12
7.6.1. How to Run the Load Schema Function	7-12
7.6.2. How to Check for a Successful Load Schema Completion	7-14
7.6.3. Load Schema and Object Domains	7-14
7.6.4. Unload Schema Utility	7-15
7.6.5. Deploying Roundtable Objects to an AppServer	7-15
 <b>Interfaces</b>	 A-1
A.1. Introduction	A-1
A.2. Environment Setup on Execution: rtbrun.p	A-1
A.3. Naming Program: rtb/p/rtb_bnam.p	A-2
A.4. Build Program	A-4
A.5. Report Source Files Locations	A-5
A.6. Task Record Access Using the Ref-num Field	A-6
A.7. Hooks and Application Programming Interfaces	A-6
A.7.1. Hooks	A-6
A.7.2. Application Programming Interfaces (API)	A-8
A.8. External Session Compiling	A-8
A.8.1. Implementing External Session Compiling	A-9
 <b>Glossary</b>	 G-1

---

# Preface

---

## 1. Purpose

This book provides the Roundtable user with in-depth information about the Roundtable Total Software Management System.

## 2. Audience

This book is intended for Progress developers, Managers of Progress developers and staff responsible for quality assurance and deployment of Progress applications.

## 3. Organization of This Guide

- Chapter 1, “Software Configuration Management”

Provides a high-level overview of the configuration management discipline and explains how to use Roundtable to achieve effective Software Configuration Management.

- Chapter 2, “The Tabletop”

Provides an overview of the Roundtable Tabletop and demonstrates the integration between the Progress ADE and Roundtable.

- Chapter 3, “Roundtable Administration”

Discusses each area that must be addressed before you set up your system.

- Chapter 4, “Workspaces”

Introduces you to Workspaces and presents detailed information about utilizing Workspaces in your development environment.

- Chapter 5, “Task Management”

Explains Task management in depth.

- Chapter 6, “Objects”

Explains each of the Object Types available in the Roundtable system.

- Chapter 7, “Tools”

Documents a number of useful tools provided in the Roundtable environment.

- Appendix A, “Interfaces”

Provides examples of how to access Roundtable data directly and how to add your own interface routines.

## 4. Typographical Conventions

This guide uses the following typographical conventions:

- Bold typeface indicates: Commands or characters that the user types
- That a word carries particular weight or emphasis Italic typeface indicates:
  - Progress variable information that the user supplies
  - New terms
  - Titles of complete publications
- Monospaced typeface indicates:
  - Code examples
  - System output
  - Operating system filenames and pathnames

The following typographical conventions are used to represent keystrokes:

- Small capitals are used for Progress key functions and generic keyboard keys.

**END-ERROR, GET, GO, ALT, CTRL, SPACEBAR, TAB**

When you have to press a combination of keys, they are joined by a dash. You press and hold down the first key, then press the second key:

**CTRL-X**

When you have to press and release one key, then press another key, the key names are separated with a space.

**ESCAPE H**

**ESCAPE CURSOR-LEFT**

---

# Software Configuration Management

## 1.1. Introduction

The Roundtable product is a team-oriented extension for the Progress development environment that provides extensive Software Configuration Management (SCM) and programming productivity tools. This chapter provides a high-level overview of the configuration management discipline and explains how to use Roundtable to achieve effective SCM.

SCM is the discipline of managing the entire life cycle of a software project. While the term is often used to describe change control systems, just implementing a change control system does not mean that a organization is practicing SCM.

Practicing SCM requires the application of business and engineering policies and procedures to ensure an appropriate level of control and auditability throughout a software project. Implementing SCM involves introducing tools to help manage the many aspects of both the business and engineering domains of the software project.

SCM is comprised of four well-defined activities: Configuration Identification, Configuration Control, Configuration Auditing, and Configuration Status Accounting.

An organization that implements SCM usually focuses on the tools that the SCM software provides rather than on the SCM discipline. A key issue in implementing SCM is identifying the roles and responsibilities of the individuals involved in the software project. A primary concern of the SCM discipline is how teams work together to build systems. The organization that implements SCM can realize enormous benefit from implementing strong engineering and business management policies in support of the team process, regardless of the tools provided by the SCM software. An organization that effectively implements SCM unifies business and engineering management disciplines.

Just about everyone in the software industry has a horror story about the project that doubled or tripled in cost and failed to do anything useful. Most of these failures could have been avoided if the organizations had implemented SCM. SCM's control of and visibility into the software development process would have made it possible for management to adjust, redefine, or cancel the project before it ran aground.

The most difficult issue facing the business manager of a software development effort is auditability. It is extraordinarily difficult to know, with any degree of certainty, what the state of a software project is at any given time. Additionally, the trend is toward more complex software tools, and business systems of increasingly larger scope.

The ideal SCM system allows you to simplify, streamline, and seamlessly integrate the management of the software life cycle into the development environment so that business managers can play an active and meaningful role in the process of software definition, development, and maintenance. The challenge is to introduce the management principles of control, auditability, and status accounting in a way that engages the software engineer, designer, and quality assurance staff in the cooperative effort.

Software engineers often resist introducing management practices into the software development process. They are concerned that management practices will cramp creativity and increase paperwork. To deal with these issues and maintain the morale of the group, the new tools must do at least as much for the software engineer as they do for the business manager. The Roundtable system can and has accomplished this!

When you present engineering personnel with an SCM implementation, the main reason for their resistance is that they see the system as extra work and an affront to their abilities. Telling the software engineer that the SCM system will allow management to manage them better will not get engineers excited. To get their attention, you show them new tools that improve communication among the engineering staff, reduce meeting time, allow decisions on change requests to be made quickly, and keep management out of their hair. Roundtable tools integrate with the development environment and fill the needs of software, quality assurance, and support engineering as well as the needs of management.

Roundtable allows management to become more involved in the software development process without compromising or inhibiting the creativity or productivity of the engineering staff.

## **1.2. Exploiting the Opportunity**

Roundtable makes it possible for everyone involved in the SCM process to communicate and share information electronically. Roundtable's task management paradigm provides engineers and managers with almost immediate response to action item requests throughout the project. Additionally, it puts much of the project information at the fingertips of both managers and engineers right inside the development environment.

One of the most difficult steps in implementing SCM is choosing the correct SCM model for your organization. Roundtable allows your organization to design and implement an SCM model that achieves a high degree of concurrence between the capabilities of the system and the needs of the project being managed. The essential issue in designing the model is how much of the configuration auditing process to apply and how many workspaces to implement.

## **1.3. Principal SCM Activities**

An SCM system is comprised of tools that support the four basic activities of SCM:

- Configuration Identification [1–3]



- Configuration Control [1–12]
- Configuration Auditing [1–15]
- Configuration Status Accounting [1–18]

### 1.3.1. Configuration Identification

Configuration identification is simply the identification of a relative arrangement of software system components. An important part of configuration identification is the realization that a software project is comprised of much more than source code. A software project might include:

- Contract or marketing specifications
- Functional requirements documentation
- Architectural design documents
- Quality assurance guidelines
- Coding standards documentation
- Component analysis documents
- Component design documents
- Source code/binary code
- Tool configurations used to produce system builds
- Test data suites
- User documentation (including on-line help)

The configuration identification framework must provide an intellectual and mathematical basis for describing the relative arrangement of these system components at a specific point in the development process. Over time, software systems undergo an iterative, incremental evolution that can be described as a series of baselines. The following table lists some commonly used SCM baselines. The terms provide a good starting point for a discussion of which baselines a given product development effort needs.

Abbreviation	Description
FB	Functional baseline (often called requirements specification)
AB	Allocated baseline (often called functional specification)
DB	Design baseline (often called engineering baseline)
Alpha	Alpha baseline
Beta	Beta baseline
PB	Product baseline (golden masters)

The following table lists system components and indicates the baselines with which they are usually associated:

<b>System Components</b>	<b>FB</b>	<b>AB</b>	<b>DB</b>	<b>Alpha</b>	<b>Beta</b>	<b>PB</b>
Contract or marketing specifications	Yes	Yes	Yes	Yes	Yes	Yes
Functional requirements documentation	Yes	Yes	Yes	Yes	Yes	Yes
Architectural design documents		Yes	Yes	Yes	Yes	Yes
Quality assurance guidelines		Yes	Yes	Yes	Yes	Yes
Coding standards documentation		Yes	Yes	Yes	Yes	Yes
Component analysis documents		Yes	Yes	Yes	Yes	Yes
Component design documents			Yes	Yes	Yes	Yes
Source code/binary code				Yes	Yes	Yes
Tool configurations used to produce system builds				Yes	Yes	Yes
Test data suites				Yes	Yes	Yes
User documentation (including on-line help)					Yes	Yes

Your system may require more or fewer baselines, and some of the system components belong in different baselines. Your objective in identifying a baseline is to define what should be in the system at a given point in time.

Baselines are often incorrectly referred to as milestones. However, milestones identify points in the project schedule where you reach specifically identified goals. Often, there are many project milestones within each project baseline. Configuration auditing is the process of ensuring that a system does contain everything implied by its baseline status.

### **1.3.1.1. SCM and the Development Cycle**

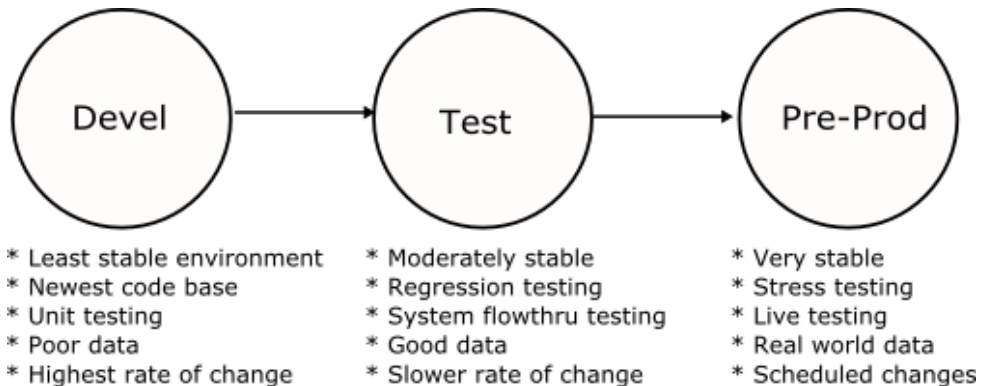
Early SCM efforts saw the waterfall development cycle entrenched in the policies, procedures, and tools used to implement the SCM discipline. Modern software development incorporates an iterative development process that includes rapid prototyping, JAD sessions, and component-oriented system construction. These and other factors make the waterfall development cycle a poor model around which to construct project- and software-management policy and procedures. If SCM is to be a natural and intuitive part of the software development process, its policies, procedures, and tools must accurately record and model the activities of the process. The SCM discipline must support iterative development and the management of parallel activities.

Iterative development involves the incremental improvement of a baseline through a series of measure, cut, and fit cycles. The underlying theory is that it is not possible to completely define the problem domain of a given application without the experience of trying to build some parts of it. Design methodologies are based on the concept of decomposition. You must break down a large system into smaller parts in order to understand the relationships of objects in the problem domain.

The relationships of components in a complex software system are not always obvious until the development process is underway. Changes in system design often occur as a result of issues that surface during the development process. Often, decision makers for a project are not technically oriented and can be reluctant to make decisions based on design abstractions. Instead, they want to see major portions of the system functionality surfaced before they make these decisions.

Roundtable provides an answer to this iterative development process by allowing multiple instances of the software system to exist concurrently. These instances are called workspaces. These workspaces support an incremental development cycle that is fully controllable under the SCM disciplines.

Refer to the three workspaces in the following diagram:



The arrows indicate a flow of new and updated system components from workspace to workspace. This flow is controlled by you, the developer, and generally occurs when a self-consistent set of changes is completed in one workspace and is ready for importation into the next (target) workspace. The important issue in this promotion of changes from one workspace to another is that all modified components are promoted together as a unit, and that comprehensive documentation of the impact of the modifications on the target workspace is available.

A workspace is defined primarily by its use, while a baseline defines the expected content of a workspace. Because you are in control of how many workspaces exist in your

development environment and how they are used, Roundtable has no strict rules regarding the relationship between workspaces and baselines. However, here are some general guidelines based on the simple workspace example previously shown:

- Development workspace — Primary development area
- Functional baseline
- Allocated baseline
- Design baseline
- Test workspace — Primary testing, limited changes (mostly bug fixing)
- Alpha baseline
- Beta baseline
- Pre-prod workspace — Primary deployment
- Production baseline

In some environments, there is a tendency to think of the Test and Pre-prod environments as builds of a system. This approach is clumsy and less than satisfactory for larger development efforts as it precludes the opportunity for maintenance or emergency patch activities in any workspace except Development. If there is significant change in the development environment after the build that represents the Test workspace, it may not be possible to produce a patch for the test environment in a timely way.

Roundtable provides extensive support for identifying potentially dangerous concurrent changes to software system components in different workspaces. This conflict is called orphan change management. For more detail, see Section 4.4, “Object Variants” [4–15]. This support for orphan change management allows parallel development to occur in two or more workspaces simultaneously with an appropriate level of control and visibility.

Your business model drives your decision to allow changes outside the Development workspace. Consider the following situation.

You oversee the development of an in-house MIS system of significant size. Management demands that some, but not all, of the system become operational as soon as possible. Your team members code these early deliveries first and promote them from the Development workspace into the Test workspace, where testing begins.

In the meantime, programmers working on the new system components make changes to some of the previous components to accommodate the interface and shared-data requirements.

In the Test workspace, the testers discover bugs and other problems in the system components. The programmers, working on the system in the Development workspace, inform you that while they can make the changes requested by the testers, they cannot

deliver the changes into the Test workspace because the system components in question have been modified to work with the new system components. However, management still wants the system components that are in the Test workspace.

This situation presents a compelling argument for making the bug fixes and required changes in the Test workspace. The Roundtable orphan change management functionality makes this possible with minimum risk.

Parallel development occurs when there are many variations (or flavors) of a system. For example, a core business system might be modified for different clients or for vertical markets. Roundtable maintains each variant in a separate workspace and manages the promotion of code.

A later section in this chapter addresses more advanced issues of workspace management for a variety of situations.

### **1.3.1.2. Configuration Item Identification**

In Roundtable, the basic configuration item is called an object. Each object is stored in a repository, which is a Progress database. You can query information on these objects using the Progress 4GL.

The repository stores a version ancestry for each object, which is comprised of each version of the object. Rather than store the full content of each object, most object versions are in a delta format which lists only the changes from the previous version.

Objects are stored in the repository using a unique key with the following components:

- Object type
- Product module
- Object name
- Version code

Any configuration of your system can be expressed as a list of object versions, with each object version specified by the four component values required to extract the object from the repository. When you specify the contents of a workspace, Roundtable extracts the appropriate objects from the repository and writes the contents of the object to OS files or, as in the case of schema objects, into the Progress database schema.

The object types supported by Roundtable include:

- PDBASE: A database definition.
- PFILE: A table definition.
- PFIELD: A field (column) definition.

- PCODE: A general file container, usually source code or binary resources.
- DOC: A document file.

The other components of the key are described in the following sections.

### **1.3.1.3. Configuration Hierarchy—Product Modules and Workspaces**

Configuration identification of a system involves decomposition of the system into smaller, more manageable, modules. Roundtable supports the mapping of a logical configuration hierarchy to a physical configuration hierarchy.

The logical configuration hierarchy is the hierarchy of the objects in the repository. Each object belongs to a product module, and each product module belongs to a product. This defines a three-level, logical view of the object stored in the repository:

- Product
- Product module (one or more per product)
- Object (one or more per product module)

The logical configuration hierarchy provides a way to define the logical relationships of objects quickly. It also serves as a form of documentation for your system design. Generally, it is a good practice to define your product modules along the functional boundaries of your system. For example, if you develop a general accounting application, you may have product modules for general ledger, accounts payable, and accounts receivable.

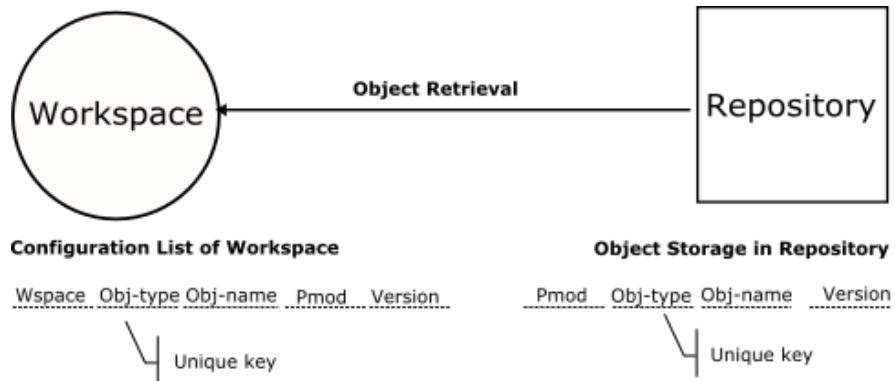
The physical configuration hierarchy is the definition of workspace contents, down to the specific directories in which to store objects. Each workspace represents a collection of object versions. Roundtable manages a table of these object versions, called the configuration list. The following fields define the unique key for objects in this table:

- Workspace ID (Wspace-id)
- Object type (PDBASE, PFILE, PFIELD, PCODE, or DOC)
- Object name

In conjunction with the object type and object name above, the following fields are included in the configuration list, allowing the unique identification of the object version contained in the workspace:

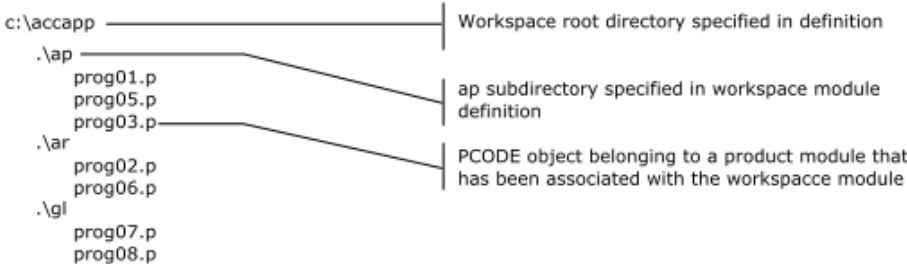
- Object Version
- Object Pmod

Refer to the following diagram:



Each entry in the configuration list also specifies a workspace module. Workspace modules map objects belonging to a product module, associated with the workspace module, to a specific directory structure. (The directory structure is always specified relative to a root directory for the workspace.) Each product module is associated with only one workspace module, but many product modules can be associated with that workspace module. This allows objects from multiple product modules to coexist in the same directory if necessary. This functionality allows the effective management of custom systems.

The following diagram illustrates a simple system:



Configuration List			
Workspace	Workspace Module	Object Pmod	Object Name
accapp	ap	ap	prog01.p
accapp	ap	ap	prog05.p
accapp	ap	ap	prog03.p
accapp	ap	ap	prog02.p
accapp	ap	ap	prog02.p
accapp	ap	ap	prog06.p
accapp	ap	ap	prog07.p
accapp	ap	ap	prog08.p

Workspace Modules	
Module	Directory
ap	ap
ar	ar
gl	gl

Product Modules	
Pmod	Modules
ap	ap
ar	ar
gl	gl

There is a one-to-one correspondence between the workspace module name and the directory name associated with the workspace module. In addition, there is a simple one-to-one correspondence between the workspace module name and the product module name. When you design a new system, you should retain these simple relationships.

The following diagram demonstrates a slightly more complex system:



```

c:\accapp -----| Workspace root directory specified in definition
  .\ap
    prog01.p
      .\rpts -----| Here, a special directory just for reports has
        prog05.p      | been created. Both a workspace module and
        prog03.p      | product module must be created to manage
  .\ar                | this subdirectory. See tables below.
    prog02.p
    prog06.p
  .\gl
    prog07.p
    prog08.p

```

Configuration List			
Workspace	Workspace Module	Object Pmod	Object Name
accapp	ap	ap	prog01.p
<b>accapp</b>	<b>ap-rpt</b>	<b>ap-rpt</b>	<b>prog05.p</b>
accapp	ap	ap	prog03.p
accapp	ar	ar	prog02.p
accapp	ar	ar	prog02.p
accapp	ar	ar	prog06.p
accapp	gl	gl	prog07.p
accapp	gl	gl	prog08.p

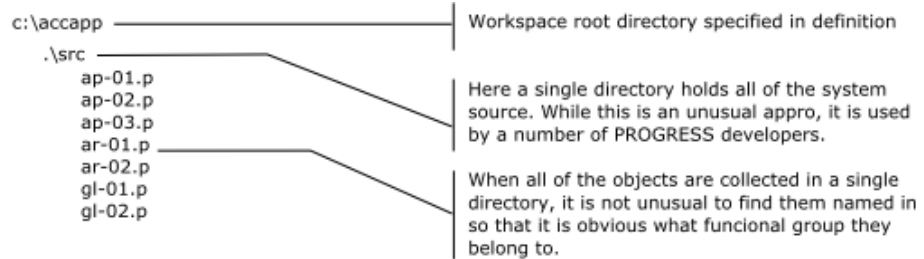
Workspace Modules	
Module	Directory
ap	ap
<b>ap-rpt</b>	<b>ap\rpt</b>
ar	ar
gl	gl

Product Modules	
Pmod	Modules
ap	ap
<b>ap-rpt</b>	<b>ap-rpt</b>
ar	ar
gl	gl

The subdirectory aprpt contains accounts payable reporting procedures. This example illustrates the need to create a workspace module and product module for each subdirectory in the system.

You might want to map objects that appear in the same physical directory to different product modules. The following diagram shows this approach:



Configuration List			
Workspace	Workspace Module	Object Pmod	Object Name
accapp	ap	ap	ap-01.p
accapp	ap	ap	ap-02.p
accapp	ap	ap	ap-03.p
accapp	ar	ar	ar-01.p
accapp	ar	ar	ar-02.p
accapp	ar	ar	ar-03.p
accapp	gl	gl	gl-01.p
accapp	gl	gl	gl-02.p

Workspace Modules	
Module	Directory
ap	src
ar	src
gl	src

Product Modules	
Pmod	Modules
ap	ap
ar	ar
gl	gl

So you can see by these examples, it is possible to map any level of directory structure to workspace modules in the Roundtable environment.

### 1.3.2. Configuration Control

The Configuration Control Board (CCB) plays a central role in the configuration control process. It is often comprised of both buyer and seller representatives and is responsible for making decisions about the changes to be made to the system definition during the course of the project. The CCB approves, monitors, and controls:

- The conversion of design objects into system (software) configuration items
- Changes to the system

To ensure the success of an SCM effort, it is important to adopt a formal SCM plan. A typical Configuration Management Plan contains the following checklist:

#### 1. System Description

- An overview of the system being built
- Configuration management (CM) organization
- The CCB
- Identification

- Control
  - Auditing
  - Status accounting
  - Other product assurance disciplines
2. CM Tools
    - Identification tools and labeling conventions
    - Control tools
    - Auditing tools
    - CM procedures
    - Design stages
    - Development stage
    - Deployment stage
    - Operational/maintenance stages
  3. CM Resources
    - Budget dollar
    - Budget staffing
    - Budget other

There are many good texts on SCM that provide information on how to implement a full SCM plan and define and assign the responsibilities of the CCB. A common-sense approach to implementing a CCB is usually enough, unless the software project is huge or is for the military. For work done under government contract, find out which of the many SCM standards documents the contract requires and implement those standards. The standards cover every phase of the software project life cycle, and even provide sample forms and report formats to facilitate communication during the project.

### **1.3.2.1. Task Management**

Roundtable provides a task paradigm for assigning and tracking work performed in a software project. No changes may be made to objects under Roundtable control without an active task to perform the work. Use the Roundtable security system and tasks together to implement a variety of controlled workflow policies. These policies can vary by workspace.

As the system director, you determine who can create, use, and complete tasks. In a loosely managed environment, it is not unusual for programmers to have this authority. When you require complete control over the changes made in a system, you can set security permissions so that only you can create a task, check out an object under the task,

and complete the task. In other words, the programmer can only work on objects you provide under the task. Roundtable security is set up by function and user, so you can achieve a balance between control and flexibility.

Tasks also provide an effective reporting mechanism. You can generate a report that describes the task and each of the objects being created or modified under the task. This report provides information to team members who are interested in your work because it overlaps theirs, and to managers who need to track the Progress of the project.

On smaller projects, the CCB may directly authorize all tasks, even if the tasks are actually created and entered by the programming staff.

### **1.3.2.2. Change Control**

Configuration control requires that your SCM system provide at least the following three simple version control capabilities:

- Protection from lost changes
- Version identification
- Version retrieval

Roundtable provides protection from lost changes as a check-in/check-out process that essentially grants write access to a single user for a system component in a workspace. Version identification tracks current and previous versions of a file using a unique numeric identifier. Version retrieval allows a user to request a copy of some previous version of a file.

Roundtable provides the following enhanced change control features:

- Version branching (parallel development/orphan change management)
- Variant identification and management (management of custom variants having independent version ancestries)
- Shared and private workareas (sometimes called sandboxes)

The Roundtable workspace management functionality provides another level of change control not found in most SCM systems, a form of version merging at the system-configuration level.

Roundtable provides a sophisticated mechanism for both protecting and sharing work being performed by team members. When you want work in Progress to be completely isolated from other users, you may check source objects out to a task directory that is not visible to other users of the workspace. The previously completed version of the source object remains in the workspace directory structure so that other users can continue to use the older version of the object. (Remember, all work done within a Roundtable-controlled

workspace is done under an assigned task. You may specify a task directory when you create the task. The task belongs to the programmer to whom it is assigned, and the task directories should be unique to the task, if possible.)

When two or more programmers must work closely together on part of the system, Roundtable can copy the source objects being worked on by a programmer into a group-directory that is associated with one or more tasks. Roundtable sets up the PROPATH with paths in the following order:

- Task directory
- Group directories
- Workspace directory

This capability allows you to make a source object that is not yet completed available to other programmers. This is useful when you need to modify a common include file or procedure.

You may also want to make a source object that is not yet completed available to all of the users of a workspace. Roundtable can copy a source object from the task directory into the workspace directory on demand. Roundtable can also reverse this decision and restore the previously completed version of the object into the workspace directory. This functionality makes it possible for everyone to test modifications before an object is complete.

Roundtable manages the placement of a source object in the task, group, and workspace directories based on the share status of the source object version. The share status of an object can be changed as necessary, and Roundtable performs all association housekeeping for you. This housekeeping includes recompilations to ensure accurate system views for each user.

### **1.3.3. Configuration Auditing**

Configuration auditing is the process of confirming that all system components that should be in a given baseline are in the baseline. The checklists in the next few sections illustrate some of the questions asked during the audit process for sample baselines. When a baseline audit is completed, the baseline is said to be sanctioned. Roundtable provides workspace, product, and task reports that identify the status of each registered component (object). It also provides textual information describing the component and its version identification. Use these reports to address some of the questions on the configuration audit checklists.

Configuration auditing is the mechanism management uses to ensure that a software project is on track and building what is actually required. Some argue that a given software project has so much R&D content that configuration auditing is not useful. This is not the case at all. If the team must use exploratory engineering to refine the product definition, this should usually be done without the overhead of full configuration auditing. However,

it is dangerous and foolhardy to commit to the development of a product if its required concepts, design, and implementation strategies cannot be stated at the level of detail required by the configuration audit process.

### **1.3.3.1. Functional Baseline Audit Checklist**

The functional baseline (FB) describes the system on a top-level functional basis. The FB is the first baseline produced. An SCM audit of the functional baseline should answer the following questions:

- Is there is a clear trace between system requirements and software requirements?
- Is it possible to sort out software from system requirements?
- Is there a system concept that is consistent with stated operational requirements?
- Is the functional design of subsystems consistent with stated system requirements?
- Do the user, seller, and buyer agree on the content of the functional baseline?

### **1.3.3.2. Allocated Baseline Audit Checklist**

The allocated baseline (AB) is the mapping of system design to functional components. It is important to know what parts of the system provide the functionality identified in the FB. An SCM audit of the AB should answer the following questions:

- Is there clear traceability of the functional specification in the FB and the allocated functional items in the AB. In other words, is it clear which parts of the system provide each function identified in the FB?
- Are all functions that are identified in the FB mapped to functional items in the AB?
- Are models and algorithms that define functions to be performed logically and mathematically correct and consistent in both a verification and validation sense?
- Does the AB clearly separate hardware, off-the-shelf software components, and new software functions?
- Are all questions concerning technical options, trade-offs, operation requirements, etc., that may impact subsequent detailed design, answered or at least identified?
- Do the user, seller, and buyer agree on the content of the AB?

### **1.3.3.3. Design Baseline Audit Checklist**

The design baseline (DB) is comprised of detailed designs for each functional component of the system. An SCM audit of the DB should answer the following questions:

- Does each detailed design item correlate with a functional component?

- Is the DB complete enough to be used in a design review process?
- Has a design review process been completed and all resulting issues resolved?
- Does the DB fulfill product assurance requirements (including design standards, procedures, and facilities for testing)?
- Is the overall architectural and design approach clear?
- Do the user, seller, and buyer agree on the content of the DB?

#### **1.3.3.4. Alpha Baseline Audit Checklist**

The alpha baseline should define the first cut of some or all of the system that can be examined as a working software system and used to gather information from potential users and do proof-of-concept testing. An SCM audit of the alpha baseline should answer the following questions:

- Are the screens and processes in the system a direct expression of the specifications in the AB and the design in the DB?
- Does the alpha baseline contain enough functionality to allow meaningful comments by the software reviewers?
- Has the alpha baseline satisfied product assurance requirements? (Note that these requirements may be greatly relaxed from those applied to the product baseline, described later.)
- Have provisions been made for the support of the alpha release?
- Has a plan been developed for incorporating alpha baseline comments into the system?
- Do the user, seller, and buyer agree on the content of the alpha baseline?

#### **1.3.3.5. Beta Baseline Audit Checklist**

The beta baseline should define the first cut of some or all of the system that can be examined as a working software system used in an actual production setting and used to gather information from potential users and do proof-of-concept testing. An SCM audit of the beta baseline should focus on establishing answers to the following questions:

- Are the screens and processes in the system a direct expression of the specifications in the AB and the design in the DB?
- Does the beta baseline contain enough functionality to allow its use in a production setting?
- Has the beta baseline satisfied product assurance requirements? (Note that these requirements may be marginally relaxed from those applied to the product baseline, described later.)
- Have provisions been made for the support of the beta baseline release?

- Has a plan been developed for incorporating beta baseline comments into the system?
- Does an expedite mechanism exist for reporting critical bugs discovered in the beta baseline?
- Are the appropriate processes in place to develop and deliver patches to beta baseline users?
- Have adequate policies and procedures been developed to track and manage beta baseline users?
- Do the user, seller, and buyer agree on the content of the beta baseline?
- Do the user, seller, and buyer concur that the beta baseline is complete and free of any defects that prohibit establishing the beta baseline?

#### **1.3.3.6. Product Baseline Audit Checklist**

The product baseline (PB) is comprised of the completed system. It defines the system and is often referred to as the "as-built" definition. An SCM audit of the PB should answer the following questions:

- Are the source code and data structures of the PB a direct translation of the DB?
- Do the FB, AB, DB, and PB correspond? In other words, is it possible to state with certainty that what was contracted to be built was in fact built?
- Has the PB satisfied product assurance requirements?
- Are all of the deliverable components that comprise the system registered as part of the PB?
- Does the PB fulfill all known and specified operational requirements?
- Have standards and good programming practices been met and approved by the product assurance department?
- Do the user, seller, and buyer concur that the PB is complete and free of any defects that prohibit establishing the PB?

### **1.4. Configuration Status Accounting**

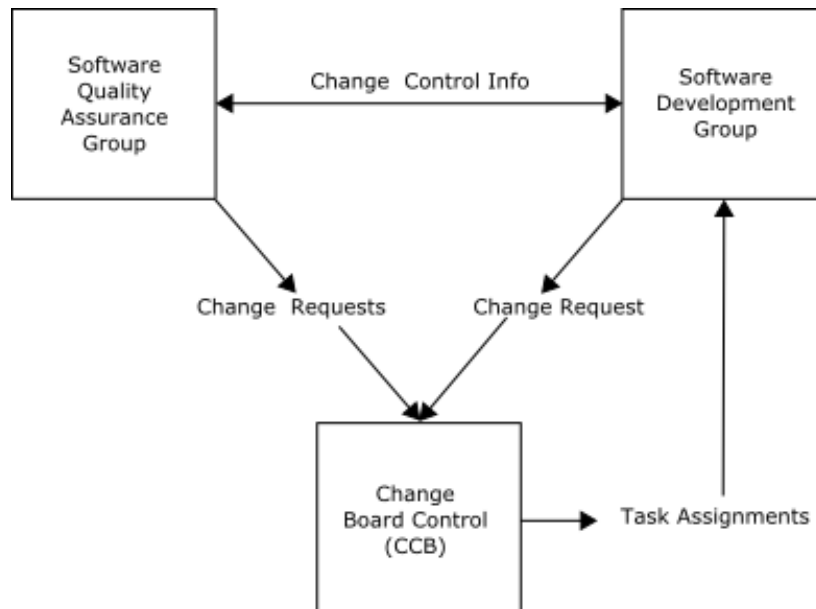
Configuration status accounting ensures that a complete and accessible record of the changes to a software system and reasons for such changes are available. Most programmers do configuration status accounting without knowing they do it, by keeping a programmer's notebook and inserting comments in the source code. The objective in configuration status accounting is to record why, when, and by whom a change is made. Configuration status accounting is often a reporting function that pulls the required information out of data stores created and managed by the configuration control tools in the SCM system.



Roundtable provides extensive configuration status accounting through the implementation of task and workspace management. Essentially, no change can be made to the software without it being tracked as part of a task in a specific workspace.

## 1.5. SCM Information Flow

You should develop a formal information flow among the groups involved in the SCM effort. A simple example of this information flow follows:



This simplified model places the CCB in a role of direct, day-to-day involvement in the development process through the creation of task assignments. In Roundtable, all work is performed under the auspices of a task assignment, and changes can be tracked back to that task. It is normal practice to enter a description of the change request or design reference into the task assignment so that changes to the system made under a given task can be tracked back to an original specification document. This example uses change request as a general term covering all types of change notification documents. Change notification documents are not directly managed in the Roundtable system, although there is an indexed userref# field in the task record that you can use to tie the task record to a user-supplied defect record or group of records.

## 1.6. Benefits of SCM

SCM illustrates the development process and provides better information for management

and more predictable delivery schedules. The goals of SCM are to:

- Improve profitability
- Save time
- Reduce the amount and improve the quality of communication between management and programmers
- Reduce the cost of producing software
- Achieve a net increase in productivity
- Shorten the development cycle time from design to delivery
- Improve the accuracy of the estimating process
- Improve the project management information flow within the development, testing, and deployment cycles
- Make the version control process as transparent as possible
- Make higher-quality information quickly available to each team member
- Make accessing system configuration information a natural and simple part of the development process

## **1.7. Workspace Networks**

You implement SCM policy by defining what work should be done in each Roundtable workspace and by defining the flow of changed objects allowed between workspaces. The Roundtable importation process searches source workspaces for objects that should be imported into the current target workspace. You define networks of workspaces by specifying a list of source workspaces for each workspace, and by providing the additional product configuration information that allows Roundtable to determine which objects in the source workspace are candidates for importation. The definition of source workspaces is part of the setup for a new workspace and does not have to be repeated for every importation.

## **1.8. SCM for Product Releases**

SCM policies and procedures for managing a software development effort depend on the kind of software development being undertaken. It is common for software development to result in a product release.

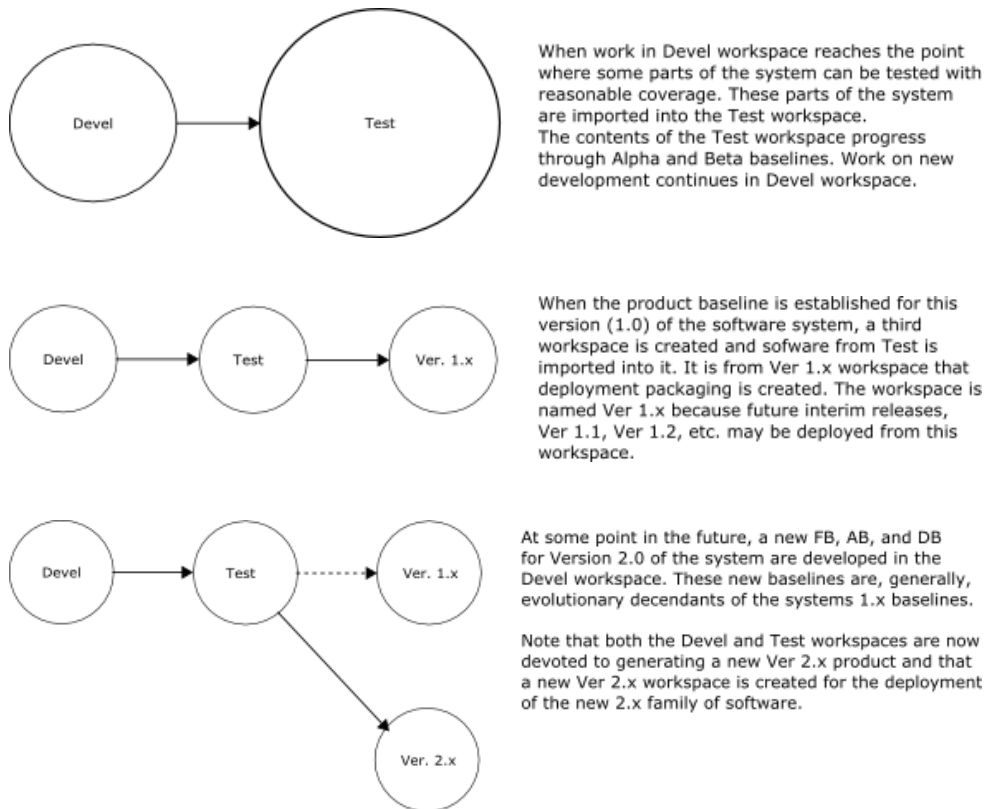
Product releases are generally deployed to large numbers of users and packaged for installation, by the user at the user's site, with little or no involvement by the software developer. The ongoing development of the Progress 4GL is a good example of a product-release-oriented development effort.

Product releases occur at well-defined intervals and often contain significant advances in

system functionality. It is not unusual for a year or more to pass between major releases of such a product.

Many justifications exist for pursuing the product release strategy. One of these is the tooling time required to manufacture everything from written documentation to packaging. Another reason is that as the complexity of a system grows, it becomes very difficult to predict the indirect impact of changes made to the system. Therefore, the system must be tested thoroughly, as a complete and stable entity, several times prior to release.

The following sequence of diagrams depicts the evolution of a product managed by the Roundtable system:

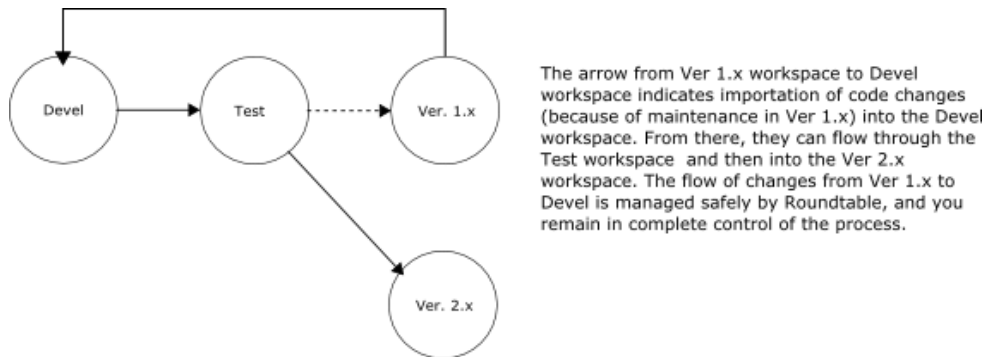


Roundtable retains a complete evolutionary development history of the system in the Devel workspace. The Ver 1.x workspace contains configuration histories for that generation of the software system. It is not unusual for maintenance efforts to be active in the Ver 1.x workspace to provide some level of support for the older software after the Ver 2.x generation is released. While it is possible to import new or modified software from

the Test workspace, this is usually a very selective and infrequent occurrence.

The strategy presented provides continued support of older generation products. Roundtable allows you to manage on-going development in any of the workspaces above and then merge the completed work into other workspaces. For example, if a bug is fixed in the Ver 1.x workspace and the modified procedure is unchanged in the Devel workspace, Roundtable informs you that you can bring the newer code into the Devel workspace so the bug can be eliminated from the Ver 2.x product.

Refer to the following diagram:



In general, it is possible to establish a flow between any two workspaces at any time. The key to effective workspace management lies in having clear policies for workspace promotions and understanding how these workspace promotion activities relate to the versions of your system.

### 1.8.1. SCM for Incremental Deployment

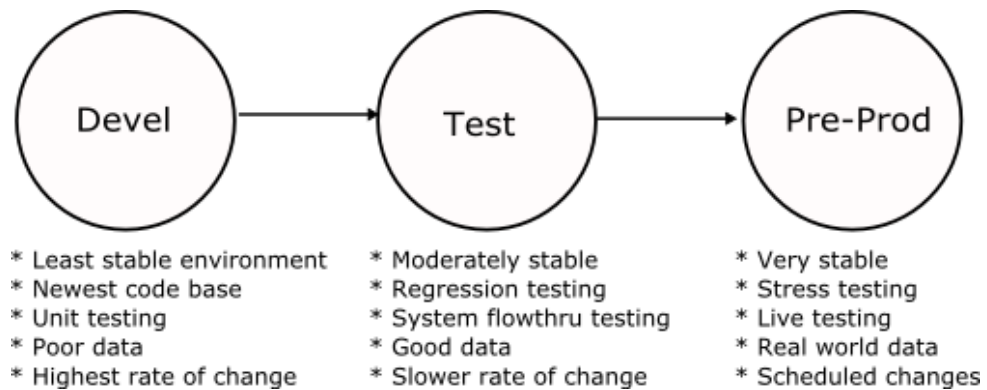
Another common software development life cycle involves the incremental deployment of an application as various components are completed. This is common in IS environments where the developers have a relatively small audience and a high degree of involvement with end users. This kind of development life cycle is also common where custom business systems are being developed and the user wants to see the progress of the application.

It is not unusual for development to commence before there is a firm and complete understanding of the scope and specific functionality to be included in the system. While many would argue that the answer to this problem is to insist on defining specifications before work begins, there are many pressures that lead to a compromise.

Regardless of what circumstance leads to implementing an incremental deployment strategy, Roundtable provides tools that allow you to manage this approach effectively,

according to accepted SCM practices.

Refer to the following diagram:



Each Roundtable workspace is a complete instance of your software application. Because Roundtable makes it possible to manage multiple workspaces easily, you can manage incremental deployment successfully.

You must be aware of some troublesome problems with incremental deployment, and you must develop policies and procedures to ensure effective management of your incremental deployment strategy. These problems involve the unavoidable parallel development that occurs in, at least, the Development and Test workspaces. Parallel development occurs whenever a change is made to the same object in two different workspaces at the same time. Roundtable, of course, ensures that each of these concurrent modifications results in different version numbers, but it is possible for changes in one of the versions to be lost. The version whose change would be lost is called the orphan version.

Consider the following situation:

1. An object is created in the Development workspace and completed with the version number 01.00.00.
2. The object is promoted into the Test workspace. Testing of the object begins.
3. The object is checked out into the Development workspace and work begins on major modifications that will take two weeks to complete.
4. A bug is discovered in the object in the Test workspace. The bug must be repaired before the remaining tests can be performed. Since the object in the Development workspace is now broken because of the new work being done on it, you need to fix the bug in the Test workspace.
5. You check out the object in the Test workspace. Since Roundtable knows that the object is checked out in the Development workspace, the system warns you that a

possible object orphan condition exists. You fix the bug in the object you have checked out in the Test workspace. You inform the programmer who is working on the object in the Development workspace of the changes you made so the fix can be incorporated into that version.

6. You check in the object in the Test workspace and choose to increment the revision number, so the version code of the object in the Test workspace is 01.01.00. This object, and other completed changes in the Test workspace, can then be imported into the Pre-Prod workspace.
7. The programmer checks in the object in the Development workspace and chooses to increment the version number, so the version code of the object is 02.00.00. Roundtable provides a report that the programmer can run before a check in or task completion process that reports on any orphan conditions that exist.
8. When you choose to import objects from the Development workspace into the Test workspace, the object (discussed in this exercise) version 01.01.00 in the Test workspace is replaced by the object version 02.00.00 completed in the Development workspace.

The most important action of this process is informing the programmer in the development area that a change was made to the same object in the Test workspace. Otherwise, the programmer may not catch the bug that was found in the original procedure and it will resurface when the new version 02.00.00 is imported into the Test workspace.

Roundtable warns the user when orphan conditions exist that could lead to lost changes.

Another potential problem area with an incremental life cycle occurs when preparing updates to system installs in the field. The Roundtable deployment system makes this simple. You can define one or more sites to be updated from a given workspace. For each of these sites, you can track each deployment sent to the site. Roundtable tracks the content of each deployment, so there is never any doubt about which objects are at a site and what the version of each object is.

Roundtable creates incremental deployments after the first deployment. An incremental deployment consists of only the objects that have changed since the last deployment, and a list of those procedures that must be recompiled because of the changes being delivered. Because only the changes in the system are being delivered, the size of the update package is usually very small compared to the size of the whole system. This often means that the update can be sent by modem rather than by tape or on a number of disks.

## **1.9. SCM for Custom Variant Deployment**

implementation of SCM. It is important to recognize the difference between customizing a core application and simply building custom systems from scratch. If you are building custom systems from scratch, each system can, and should, be treated as a separate configuration, thus avoiding any complications in the application of the SCM discipline. However, if you have a core application to which you add or change functionality,

enormous benefits come from managing the core application and each custom variant so that changes in the core application can be easily promoted into selected custom systems.

For example, customers who want to purchase your application with custom enhancements are often concerned about the difficulty of getting updates to your core application and to the custom enhancements. In fact, many of these customers may be abandoning their current systems because they can no longer get cost-effective updates from an existing vendor. Roundtable provides an effective solution to this problem.

The objects managed in the Roundtable system are stored in the repository by the following unique key:

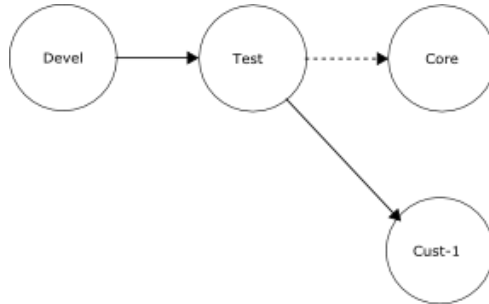
- Object type
- Product module (Pmod)
- Object name
- Version code

The pmod component of this key allows Roundtable to store two objects with the same object type and name in the repository. Refer to the following table:

Object Type	Product Module	Object Name	Version
PCODE	core_ap	menu.p	01.00.00
PCODE	core_ap	menu.p	01.01.00
PCODE	cust1_ap	menu.p	01.00.00
PCODE	cust1_ap	menu.p	02.00.00

This table shows that the object menu.p exists under both a core\_ap and cust1\_ap product module. The menu.p object belonging to the cust1\_ap is a custom variant. It represents a replacement for the menu.p object belonging to the core\_ap module.

The ability to store two objects of the same type and name, but with a different product module, allows Roundtable to track custom variant objects in a workspace. Refer to the following diagram:



This development workspace shows a development, test, and core workspace for the core application. A workspace in which custom development will occur is shown as well.

The sample object, order.p is associated with a different product module in cust-1 workspace.

**Assignment of order.p object in each workspace**

Workspace	Workspace Module	Product Module	Object Type	Version
Core	ar	core-ar	PCODE	01.00.00
Cust-1	ar	cust1-ar	PCODE	01.00.00
Devel	ar	core-ar	PCODE	01.01.00
Test	ar	core-ar	PCODE	01.00.00

To manage the development of a custom system, begin by creating a workspace in which the custom system will be developed (Cust-1 in the example). Then create one or more product modules to contain the custom objects for the system (cust1-ar in the example). You can now completely replace an object with a custom variant of the object (order.p in the example).

Objects with different product modules have completely separate version ancestries. For example, a version 01.00.00 exists for order.p under each product module. This makes it obvious that the object is a complete replacement for the core object and not simply a branch version.

The Roundtable importation process is custom variant smart. It will not unwittingly overwrite your custom work when you import changes from the core system into your custom workspace. The rule that Roundtable applies is very simple: if the object in the target workspace is not from the same product module found in the source workspace, Roundtable overwrites it only when specifically instructed to do so.

## 1.10. Distributed Development

It is sometimes necessary to coordinate and manage related development activities occurring at locations not directly connected to a central repository of configuration information. This situation might occur because of geographic separation, the involvement of separate organizations, or the use of incompatible tools in the development process.

Roundtable provides the ability to transfer configuration information among repositories to facilitate distributed development. Each of these repositories is identified by a site number



and is often referred to as a site. The following sections discuss the basic issues involved when doing distributed development and offer specific approaches to common distributed development scenarios.

This discussion about distributed development focus on implementation strategies. Use the references to chapter topics provided to supplement the materials in this overview.

### **1.10.1. Sites and Ownership**

Each distributed development group runs its own copy of the Roundtable system and has an independent repository where configuration information is stored. Each repository is identified by a unique site number. A site number of 0 is allowed. It is treated in a special manner and is the default value on first-time installations of the Roundtable system. The site designated with the number 0 is called the central site.

Each additional site in the distributed development network of repositories will have unique site numbers in the range of 1 to 999. A site with a number other than 0 is called a partner site. The site numbers can be assigned in any order but cannot be changed once assigned.

Site numbers are used to provide a unified naming convention for information belonging to each site. This is accomplished by using the site number as a separate key field in some repository tables and by using the site number as a code prefix for other tables. This identification of information by site provides the foundation for the sharing of information among distributed repositories.

Much of the information you place in the configuration repository must be identified by a user supplied code. The format of this code is important and is validated to conform to the following rules:

- If entering a code on the central site (site 0), then the code must not begin with a numeric character.
- If entering a code on a partner site (site number other than 0), then the code must begin with three numeric digits that identify the partner site.

Using this strategy, the system is able to identify the originating site of each record in the distributed network. The system allows only the originating site to modify a record.

The names of products, product modules, and subtypes are examples of items in the Roundtable system that are affected by site numbering rules.

### **1.10.2. Deployments and Partner Site Loads**

The system provides the ability to do a special form of deployment to partner sites. Deployments to partner sites include selected repository information dumped to text file

format. A utility called the Partner Site Load utility reads this repository information into the partner site repository.

The process of deploying to a partner site is no different than deploying to a customer as discussed earlier, but the type of site must be set to "partner". Partner sites can deploy to other partner sites and to the central site. The central site can deploy to any partner site.

A site receives the deployment package and runs the partner site load utility to update the repository with the new information. The receiving site will usually manage a receipt workspace that represents the latest configuration received. When a receipt workspace is used it must also be updated from the deployment.

### 1.10.3. Receipt Workspace

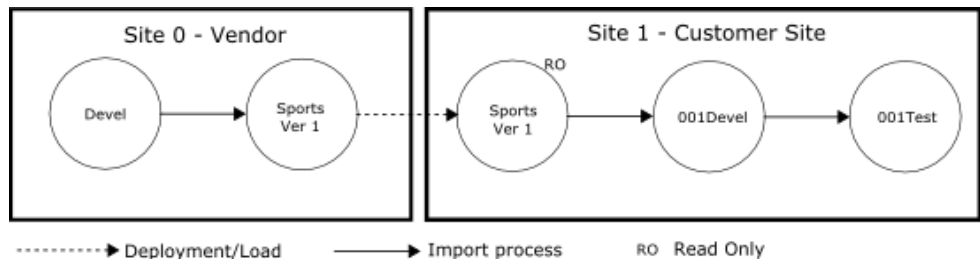
Deployments represent the configuration of a workspace at a given release level. A deployment to a partner site is comprised of two components:

- A source deployment containing a full or incremental update for the remote site
- Additional repository information sufficient to manage the delivered configuration as a workspace at the partner site

A receipt workspace is a read-only copy of the workspace that exists at the sending site. It cannot be altered at the receiving site. However, information can be imported from the receipt workspace into a local workspace. Importing objects from a receipt workspace into locally managed workspaces is performed by the import process.

### 1.10.4. Customers Doing Development

Many vendors provide source to customers so that the customer's development staff can enhance and extend the vendor's application. When the customer also has the Roundtable system, it is possible for the vendor to deliver partner site deployments to the customer. Partner site deployment facilitates the integration of application upgrades to the customer in a controlled and managed way. Refer to the following diagram:

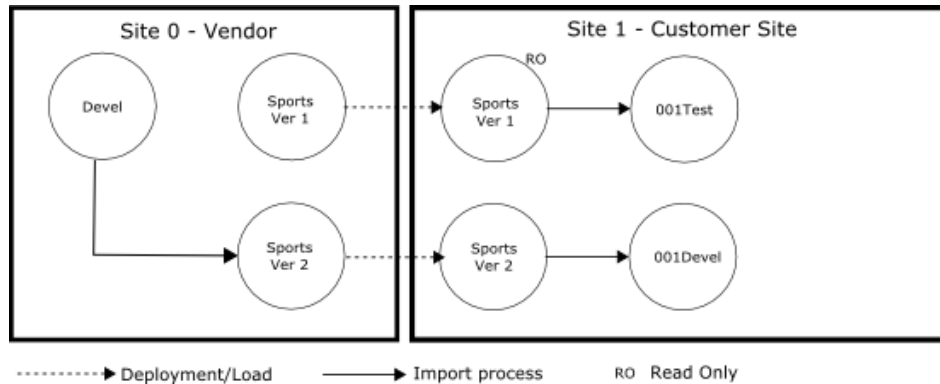


As the vendor develops new enhancements to the Sports product, they are delivered to the

customer as partner site deployments. Partner site deployments contain both repository information and the normal remote site update package.

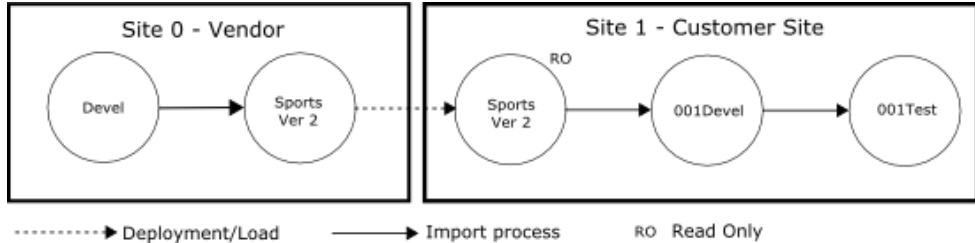
At site 1, the customer imports from the Sports Ver1 workspace into the 001Devel workspace. Where necessary, the customer can replace objects in the application with customized objects (variants) to modify the vendor's application. These custom objects are protected from overwrite in future importation processes. The customer can also extend the functionality of the application by adding new objects to the application.

The diagram shows the vendor's application deployment workspace with a name of Sports Ver 1. This is done to emphasize that the releases received by a remote site in this manner are usually incremental releases, often called dot releases. When a vendor wants to distribute a new major version of a software application, it is sometimes necessary for both the vendor and the customer to manage a legacy system for a period of time. The following diagram shows how this might be done:



In the preceding diagram the new Sports Ver 2 application is delivered to the customer in a new receipt workspace, and the workspace promotion strategy is changed at the customer's site to do integration and development with the new application. Legacy systems generally require minor maintenance and modifications, and this can be done in the 001Test workspace. The major work of integrating the new functionality of Sports Ver 2 is performed in the 001Devel workspace.

When the integration of the new system is completed, the legacy system can be dropped as shown in the following diagram:



### 1.10.5. Custom Product Modules

In the preceding section you saw how it is possible for an application system to be delivered into a receipt workspace at the customer site. Objects are then imported from the receipt workspace into the customer's development workspace. Note that the customer can replace objects in the application system with custom objects (variants) as required in the development workspace without fear that these will be overwritten during future imports.

This protection from overwrite is accomplished by creating custom products and product modules that belong to the customer site. The new objects created to replace an object in the vendor's application are created under a product module belonging to the customer's site. The import process will not overwrite an object in the target workspace when the object's product module is different from that found in the source workspace. This protects the custom objects from being overwritten.

To replace an object with a custom object in the application:

1. Create a custom product and product module that will own the new custom object if necessary. You might already have custom products and product modules from some previous modification.
2. Assign the custom product and product module to the customer's development workspace sources if they are not already assigned. This is done through Roundtable's Workspace Sources Window.
3. Add a new object as usual, but use the name of the existing object, and use your custom module for the new object. The existing object source is provided as a starting point for your new object.

#### 1.10.5.1. Distributed Development Scenario # 1

One simple form of distributed development occurs when a single organization has two development sites that cannot be connected to a central configuration repository. For this example, assume that a cooperative software development effort is pursued by two development teams separated geographically. Team A is located in India and team B is located in California.

Each team has one configuration repository and one or more workspaces relating to their different responsibilities. The teams have the following responsibilities:

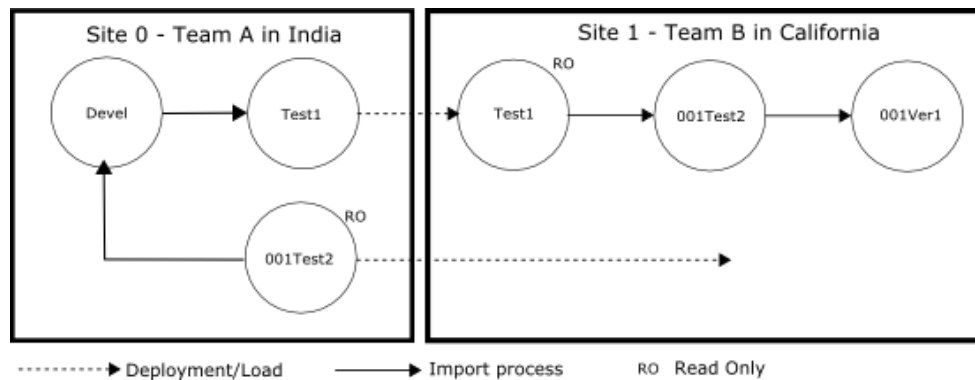
Team A - India

- Primary Development
- Unit Testing

Team B - California

- System Testing
- Deployment to Customers
- Support

The following diagram describes the distributed development process with a workspace network map:



The preceding workspace network map shows six workspaces. The objects in the **Devel** and **Test1** workspaces at site 0 can be modified by the members of team A at site 0. The objects in the **001Test2** workspace at site 0 cannot be modified by team A. Instead, only the partner site load utility can modify the contents of the **001Test2** workspace because it is a receipt workspace. A receipt workspace is always a read-only workspace as indicated by the RO designation in the diagram.

The objects in the **Test1** workspace at site 1 cannot be modified by team B. Again, only the partner site load utility can modify the contents of the **Test1** workspace because it is a receipt workspace.

The purpose of each of these workspaces is defined by the workflow of the teams:

Workspace	Work Performed
Devel	Primary development area.
Test1	Updates are imported from Devel into Test1. Unit testing is performed in this area. Deployments to the partner site in India are performed from this workspace.
001Test2	<p>Updates are imported from Test1 into 001Test2. System testing and emergency fixes are performed in this workspace. Objects fixed in this workspace begin as a copy of the object imported from Test1. These object copies belong to a product module that is owned by site 1.</p> <p>Deployments for site 0 are made out of the 001Test2 workspace. This makes it possible for the development team at site 0 to see any fixes made to the software application by team B at site 1. By using the visual difference facility team A can identify what changes should be incorporated into the system.</p>
001Ver1	Deployments to create product packaging are made from this workspace.

### 1.10.5.2. Distributed Development Scenario # 2

A more complex form of distributed development occurs when a single organization has two development sites that cannot be connected to a central configuration repository and primary development is shared. For this example, assume that a cooperative software development effort is pursued by two development teams separated geographically. Team A is located in India and team B is located in California.

Each team has one configuration repository and one or more workspaces relating to their different responsibilities. The teams have the following responsibilities:

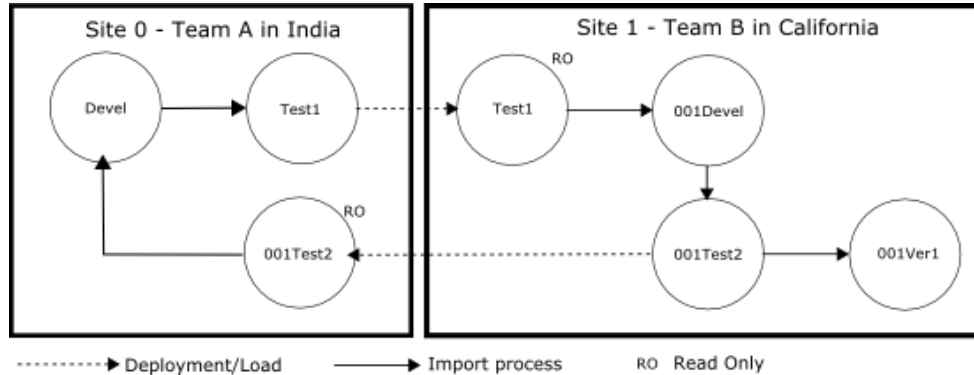
Team A - India

- Primary Development Accounting
- Unit Testing Accounting

Team B - California

- Primary Development Operations
- Unit Testing Operations
- System Testing for full system
- Deployment to Customers
- Support

The best way to describe the distributed development process is with a workspace network map as shown below:



The workspace network map above shows seven workspaces. The objects in the Devel and Test1 workspaces at site 0 can be modified by the members of team A at site 0. The objects in the 001Test2 workspace at site 0 cannot be modified by team A. Instead, only the partner site load utility can modify the contents of the 001Test2 workspace because it is a receipt workspace. A receipt workspace is always a read-only workspace as indicated by the RO designation in the diagram.

The objects in the Test1 workspace at site 1 cannot be modified by team B. Again, only the partner site load utility can modify the contents of the Test1 workspace because it is a receipt workspace.

The purpose of each of these workspaces is defined by the workflow of the teams:

Workspace	Work Performed
Devel	Primary development area for Accounting. Used by team A at site 0. The Operations code developed by team B at site 1 is imported from the workspace 001Test2 into the Devel workspace. Thus the Devel workspace contains the entire system. Members of team A can only modify objects that belong to product modules owned by site 0.
Test1	Updates are imported from Devel into Test1. Unit testing is performed in this area. Deployments to the partner site 1 are performed from this workspace.
001Devel	Primary development area for Operations. Used by team B at site 1.
001Test2	Updates are imported from Test1 into 001Test2. System testing and emergency fixes are performed in this workspace. Objects fixed in this workspace begin as a copy of the object imported from Test1. These object

Workspace	Work Performed
	<p>copies belong to a product module that is owned by site 1.</p> <p>Deployments for site 0 are made from the 001Test2 workspace. This makes it possible for the development team at site 0 to see any fixes made to the software application by team B at site 1. By using the visual difference facility team A can identify what changes should be incorporated into the system.</p>
001Ver1	Deployments to create product packaging are made from this workspace.



---

## Navigating Roundtable

### 2.1. Introduction

This chapter guides you through Roundtable's interface:

- Roundtable Screens, Keys, and Menus
- Tour of Roundtable "Views"

### 2.2. Roundtable Screens, Keys, and Menus

Roundtable is run from your home directory with the `_rtb` script, if it has been installed as suggested. See the Setting Up Roundtable Users section. Enter the following commands to run Roundtable:

```
$ cd  
$ _rtb
```

The Roundtable Main Menu appears.

Selections from this Main Menu are made by either pressing the first letter of a selection or by using the arrow keys to move onto an option and pressing the **ENTER** key. This menu can be modified by editing the contents of the `RTBSETUP` file.

See Section 3.4, “RTBSETUP File Options” [3–2].

If you select the Quit option or press **F4** to exit the menu, you are placed in the PROGRESS editor. To restart Roundtable from the PROGRESS Editor, enter:

1. run `help.p`
2. Press the **GO** key.

Data entry screens in Roundtable follow standard conventions and layout guidelines.

The three sections of the Module Selection screen include:

<b>Table</b>	This frame is a scrolling table. Scrolling is accomplished with the arrow keys and the pgup and pgdn keys. In addition, the home key positions you on the first record in the table. In this example, the table is a list of modules in the xxxdemo workspace.
<b>Details</b>	This frame displays additional information about the row currently selected in the table frame. As you scroll through the table, the information in the details frame changes.
<b>Menu</b>	<p>The menu area contains:</p> <ol style="list-style-type: none"><li>1. <b>Menu</b> Name Name of the menu</li><li>2. <b>Task#</b> Current task, if any</li><li>3. <b>Workspace</b> Current workspace, if any</li><li>4. <b>Program</b> Roundtable program that is currently running</li><li>5. <b>Bar Menu</b> Bar menu for the screen</li></ol> <p>The menu bar is used extensively throughout Roundtable. It consists of menu options with descriptions. You move between menu options using the spacebar, backspace, or arrow keys. Press the <b>ENTER</b> key to execute the option. You can also press the capitalized letter of a menu option to immediately select and execute that option.</p>
<b>FF1</b>	The <b>F1</b> key is accepted as an <b>ENTER</b> keystroke while on a menu.
<b>FF4</b>	All menus have a <b>Quit</b> option; the <b>F4</b> key is treated as a <b>q</b> when a menu is active.

On some screens there is no scrolling table. Instead, each record is shown as a full screen form. In these cases you can still scroll through records using the same keys used to scroll through a table frame.

Throughout this manual references are made to function keys. You can also use the following PROGRESS control key equivalents:

<b>F1</b>	<b>Ctrl-x</b>
<b>F2</b>	<b>Ctrl-w</b>
<b>F3</b>	<b>Ctrl-t</b>
<b>F4</b>	<b>Ctrl-e</b>
<b>F5</b>	<b>Ctrl-g</b>
<b>F6</b>	<b>Ctrl-p</b>
<b>F7</b>	<b>Ctrl-r</b>
<b>F8</b>	<b>Ctrl-z</b>

<b>F9</b>	<b>Ctrl-n</b>
<b>F10</b>	<b>Ctrl-d</b>
<b>F11</b>	<b>Ctrl-b</b>
<b>F12</b>	<b>Ctrl-a</b>

Your application can remap these function keys without confusing Roundtable because Roundtable restores the mapping it requires whenever it is run or returned to after running your application.

## 2.3. Tour of Roundtable Views

Screens allow the user different views of the data held in the repository, these include:

Tasks [2–3]	Tasks can be viewed by workspace or by programmer.
Workspace Selection [2–3]	Showing all workspaces available.
Module Selection [2–3]	Shows modules belonging to current workspace.
Object Selection [2–4]	Shows objects belonging to current module.
Xref [2–4]	Shows dependancies for the selected object.
Where Used [2–4]	Shows all objects using selected object.
Change History [2–4]	Shows all changes made to the selected object.

### 2.3.1. Tasks

The Tasks screens allow the user to create new tasks from two views, these include:

- Tasks by Workspace is accessed from the main menu by selecting Workspace → Task
- Tasks by Programmer is accessed by selecting Task from the main menu

See Chapter 5, *Task Management* [5–1].

### 2.3.2. Workspace Selection

The Workspace Selection screen allows the user to navigate the system and is accessed by selecting Workspaces from the main menu.

See the Workspace Maintenance section.

### 2.3.3. Module Selection

The Module Selection screen is accessed from the main menu by selecting Workspace →

Select , while positioned on the preferred workspace.

See the Product Modules and Workspace Module Definitions sections for information on creation and management of modules.

### **2.3.4. Object Selection**

The Object Selection screen is accessed from the main menu by selecting Workspace → Select → Select , while cursor correctly positioned to select, first, the workspace, then the module.

Choose Select to view the details for the object that you have selected.

See the following sections for examples of the detailed view for each type of object, including: PCODE, PFIELD, PFILE, PDBASE, and DOC.

### **2.3.5. Xref**

Shows all objects used by a specified object, from the Object Update screen.

Workspace → Select → Select , while cursor correctly positioned to select, first, the workspace, then the module. Choose Select to view the details for the object that you have selected. From the object's detailed menu choose Xref to access the Xref View.

See Section 6.11.9, “Xrefs Menu” [6–44] and Section 6.11.10, “Informal Xrefs Menu” [6–?] sections.

### **2.3.6. Where Used**

Shows all objects using a specified object.

Workspace → Select → Select , while cursor correctly positioned to select, first, the workspace, then the module. Choose Select to view the details for the object that you have selected. From the object's detailed menu choose Where to access the Where Used View.

See Section 6.11.11, “Where Used Menu” [6–47].

### **2.3.7. Change History**

Shows all changes made to selected object in current workspace.

Workspace → Select → Select , while cursor correctly positioned to select, first, the workspace, then the module. Choose Select to view the details for the object that you have selected. From the object's detailed menu choose Hist (or More → History ) to access the Change History View.

See Section 6.11.12, “History Menu” [6–49].

---

## Roundtable Administration

### 3.1. Introduction

This chapter discusses the following administrative tasks:

- Starting and exiting Roundtable
- Customizing Roundtable using the RTBSETUP file
- Mapping your current PROGRESS application
- Adding and maintaining products and product modules
- Adding and maintaining subtypes
- Setting up your system parameters
- Setting up security
- Dumping and loading your repository database
- Configuring Roundtable, with a step-by-step summary
- Loading your application, with a step-by-step summary

### 3.2. Starting Roundtable

In Windows Character, the Server must be running to initiate Roundtable, by clicking on the shortcut icon created from the install notes.



The shortcut working directory should be the directory where you created your RTBSETUP file.

Here is an example of the command line that you might use. Replace startup parameters etc. with values appropriate for your environment.

**`_progres -N TCP -S rtb-service -H rtb-host -p C:Roundtablehelp.r`**

In Unix, if Roundtable's database server is not yet running, then **ENTER** the following commands.



If your Roundtable database is on a remote server, then how you start your Roundtable database server will depend on your environment.

```
$ cd /u1/rtb
$_server
```

Then, from a user's account that is set up for Roundtable, Enter the following commands:

```
$ cd
$_rtb
```

For more detailed information refer to your install notes.

### 3.3. Exiting Roundtable

In Windows Character or Unix, choose **Quit** from the Roundtable Main Menu. Roundtable quits to the PROGRESS editor. To exit the PROGRESS Editor, choose File → Exit .

To restart Roundtable from the PROGRESS Editor:

1. **ENTER** `run /SCM/rtb/p/help.p`
2. Press the **GO** key.




This example assumes you installed Roundtable in a directory named "**SCM**."

### 3.4. RTBSETUP File Options


The RTBSETUP file is read to set options for the Roundtable environment. All of these options except *RTBPATH* are optional in most environments. Use only UNIX-style directory separators (/) in this file.


Options	Description
<i>RTBPATH</i>	This parameter must be supplied. It is prepended to the PROGRESS <i>PROPATH</i> on entry to the Roundtable system and contains

Options	Description
	<p>comma-delimited path segments. The first segment must contain the path to the directory in which Roundtable has been installed. The second segment must be the same as the first segment with /rtb/p appended. The third segment should be a period (.) to allow PROGRESS to find *.p programs in your rtbwork subdirectory. You may want to add additional segments to <i>RTBPATH</i> to allow PROGRESS to find special programs and tools while you are in Roundtable.</p>
<i>EDITOR</i>	<p>The name and parameters for an editor of your choice to edit a PCODE object's files. The default editor in UNIX is a modified version of the PROGRESS Procedure Editor. If you want to use an editor like the vi editor, enter the line:</p> <pre data-bbox="506 736 772 760">EDITOR "vi \$file"</pre> <p>The \$file token is replaced with the name of the file by Roundtable when Roundtable calls the editor. It is not necessary to make an entry if you want to use the default.</p>
<i>VIEW</i>	<p>The name and parameters for an editor of your choice to view a PCODE object's files. The default editor in UNIX is a modified version of the PROGRESS Procedure Editor. If you want to use an editor like the read-only version of the vi editor, enter the line:</p> <pre data-bbox="506 1114 772 1138">VIEW "view \$file"</pre> <p>The \$file token is replaced with the name of the file by Roundtable when Roundtable calls the editor. It is not necessary to make an entry if you want to use the default.</p>
<i>DOCEDITOR</i>	<p>The name and parameters for an editor of your choice to edit a DOC object's files. The default editor in UNIX is a modified version of the PROGRESS Procedure Editor. If you want to use an editor like the vi editor, enter the line:</p> <pre data-bbox="506 1491 819 1515">DOCEDITOR "vi \$file"</pre>

Options	Description
	<p>The \$file token is replaced with the name of the file by Roundtable when Roundtable calls the editor. It is not necessary to make an entry if you want to use the default.</p>
<i>DOCVIEW</i>	<p>The name and parameters for an editor of your choice to view a DOC object's files. The default editor in UNIX is a modified version of the PROGRESS Procedure Editor. If you want to use an editor like the read-only version of the vi editor, enter the line:</p> <pre>VIEW "view \$file"</pre> <p>The \$file token is replaced with the name of the file by Roundtable when Roundtable calls the editor. It is not necessary to make an entry if you want to use the default.</p>
<i>SILENT</i>	<p>This parameter requests that Roundtable call your editor and viewer programs with the UNIX SILENT command in PROGRESS. This is necessary for X window environments to get rid of the "press spacebar to continue" message on termination of the subshell. See the INSTALL file for more information on setting up your system with a PROGRESS X executable. Do not use this option if you are running a terminal emulator under X windows.</p>
<i>MAINMENU</i>	<p>This option allows you to add options to the main menu of the Roundtable application. Each option is entered as a pair of comma-delimited values of the form ",menuitem,program."</p> <p>As an example, the PROGRESS help.p program is shown added to the Roundtable menu with "Help,/home/dlc/help.p."</p> <div> Note the full path specification for the help program. Another menu option is:</div> <pre>MAINMENU "Help,/home/dlc/help.p, MY PROGRAM,./mystuff.p."</pre>



Options	Description
	 <p>You do not have to use the program's full path specification if it is available on the <i>PROPATH</i>.</p>
<i>PRINTDEST</i>	This parameter allows you to specify the default print destination to use. Your choices are: P,F,T (Printer, File, Terminal).
<i>PRINTOPT</i>	<p>This parameter allows you to specify options for the print destination. This option is only effective under the UNIX environment. It has different meanings depending on the print destination as shown in the following examples:</p> <p>Printer: "lp \$file" (Roundtable will supply \$file at the time of printing.)</p> <p>File: "myfile" (Default name of output file.)</p> <p>Terminal: "wyse60w" (Name of terminal mode to use for output display.)</p>
<i>PRINTPGLN</i>	<p>This parameter allows you to specify the page length to use for the destination.</p> <p>You can have multiple <b>PRINTDEST/PRINTOPT/PRINTPGLN</b> commands in your file to set the defaults for each possible destination. Make sure that the destination you want as a default is the last destination specified. For instance, the following sequence sets up each print designation with the Printer as the last output device specified and therefore the default:</p> <p><b>PRINTDEST "T"</b></p> <p><b>PRINTOPT "wyse60w"</b></p> <p><b>PRINTPGLN 0</b></p> <p><b>PRINTDEST "F"</b></p>

Options	Description
	<p><b>PRINTPGLN 55</b></p> <p><b>PRINTDEST "P"</b></p> <p><b>PRINTOPT "lp -ow -pjoeslocal \$file"</b></p> <p><b>PRINTPGLN 55</b></p> <p> These options values are probably not useful on your system.</p> <p>In addition to the PRINT* options, which are used for Roundtable reports, the <b>CODEPRINT</b> and <b>DOCPRINT</b> options allow you to control the printing of text from within Roundtable.</p>
<i>CODEPRINT</i>	<p>This parameter allows you to specify how Roundtable prints source files associated with PCODE objects in a workspace. This option is only effective under the UNIX environment. This is often a different command than that specified for the printer destination in the previous parameters. For instance, the following produces printed source listings with line numbers on a SUN system:</p> <pre>CODEPRINT "cat \$file / nl -ba / lp -s"</pre>
<i>DOCPRINT</i>	<p>This parameter allows you to specify how Roundtable prints source files associated with DOC objects in a workspace. This option is only effective under the UNIX environment.</p> <pre>DOCPRINT "lp -s \$file"</pre>
<i>DIFF</i>	<p>This parameter allows you to specify the default OS command to use in the source compare utility.</p> <pre>DIFF "diff"</pre>

Options	Description
<i>FIELDSORT</i>	<p>This option allows you to specify the default sort sequence used to display fields assigned to PFILE objects. For example, enter the specification below to have Roundtable display the fields in the Fieldname order:</p> <pre><i>FIELDSORT</i> "Fieldname"</pre>
<i>TEMPDIR</i>	<p>This option allows you to specify a temporary directory other than the default. This would, for instance, allow you to force all of your users to share a common directory. This is not recommended but has been a requirement at some sites. It is also possible to insert a \$user token in this path that will be resolved by Roundtable to the user's login name. If you use this feature you must ensure that your user names have no invalid characters for directory creation.</p>

### 3.5. Setting Up Roundtable Users

After ensuring that the user is a member of the user group(s) allowed to access Roundtable, you can give the user access to the Roundtable application.

Log in as the user and enter the commands below. Remember to replace /u1/rtb with the actual directory in which you installed Roundtable and /home/dlc with the actual directory in which PROGRESS is installed.

```
$ cd
$ cp /u1/rt/_rtb
$ chmod +x _rtb
$ mkdir rtbwork
$ cd rtbwork
$ vi RTBSETUP
```

Edit this parameter in the RTBSETUP file:

**RTBPATH " /u1rtb/u1/rtb/rtb/p,."**

The RTBSETUP file allows the user to specify many additional parameters. See the "UNIX Installation" section for more information on the options that can be set in this file. Note that each user needs their own working directory.



- If you are a System V user, you might want to modify the **\_rtb** script to use **newgrp(1)** to switch the user to the appropriate group prior to running Roundtable.
- The **\_rtb script** contains a **umask 002** command. If this is inappropriate for your installation, please change it.
- The **\_rtb script** runs the mpro script provided by PROGRESS. If you have not set up the **mpro** command for your environment, you should do so prior to running the **\_rtb script**. You can replace the mpro command in the **\_rtb script** with a direct reference to the **\_progres** command if required. Make sure that the environmental variables required by PROGRESS are set properly if you do this.

### 3.6. Defining a Configuration Hierarchy

The first and perhaps most difficult step in setting up the Roundtable system is defining your current configuration hierarchy. The configuration hierarchy is a map of the contents of your application and usually corresponds closely to the architecture of your system.

If you are building a system from scratch, then defining an application architecture that is intuitive and simple to manage in Roundtable can be accomplished by following these rules:

- Divide your system into products that are more or less independent of one another. Often one central product provides general services to other products, which are dependent on those services, but the central product should not be dependent on any other product.

For example, consider a suite of applications consisting of General Ledger (GL), Accounts Payable (AP), Accounts Receivable (AR), Pawnshop Point of Sale, and Furniture Store Management. All of these applications might need a set of general services, such as menu management, security, and telecommunications. An effective way of defining products is to group GL, AP, and AR into a product called Accounting. Make the Pawnshop Point of Sale and Furniture Store Management separate products and make the general services into a product called Core System. Because these products are defined separately, it will be easy to package them individually for deployment later.

- Divide each product into product modules that are well-defined subsets of the functional content of the product. In the previous example, the Accounting product has at least three modules: GL, AP, and AR. Both the Pawnshop Point of Sale and the Furniture Store Management product might have many modules.
- Associate a directory with each product module by assigning workspace modules.

- Roundtable provides three logical levels of configuration hierarchy: product, product module, and object. If you have an existing physical configuration hierarchy expressed as a deep set of subdirectories, these can be mapped using workspace modules. Before loading an existing configuration hierarchy, you must understand it. Then you develop a strategy for mapping these subdirectories to product modules.
- There are three common methods of organizing file systems to express a configuration hierarchy:
  - Application group directories
  - Directories by component type
  - Application group and component type

### 3.6.1. Application Group Directories

The most common way of organizing system components is to group source files by product module definitions. This is common because structured design methodologies involve the top-down decomposition of a software application by function. The functional hierarchy translates naturally into a hierarchical directory structure. For example, in an accounting package with an accounts payable product module and an accounts payable workspace module, your directory structure might look like this:

Directory	Directory Content
/abc_system	ABC accounting and POS system directory
/abc_system/ar	Accounts Receivable
/abc_system/ap	Accounts Payable
/abc_system/gl	General Ledger
/abc_system/pos	Point of Sale

This simple configuration hierarchy easily maps into the three-level configuration hierarchy provided by Roundtable. The /abc\_system directory becomes the workspace root path and each subdirectory is then mapped as shown below:

Product	Prod Module	Workspace Module	Subdirectory
Accounting	AR	AR	ar
	AP	AP	ap
	GL	GL	gl
POS	POS	POS	pos

### 3.6.2. Directories by Component Type

Another common way to arrange directory structures is by the types of components found in the system. This organization is typical when many discrete components are reused across functional boundaries in a software application system.

A directory structure organized by component type might look like this:

Directory	Directory Content
/abc_system	ABC accounting and POS system directory
/abc_system/edit	Data entry procedures
/abc_system/rpt	Reporting procedures
/abc_system/batch	Batch procedures

This configuration hierarchy maps into the product and product module configuration hierarchy provided by Roundtable, as shown below:

Product	Prod Module	Workspace Module	Subdirectory
Accounting	AR-Edit	Edit	Edit
	AP-Edit	Edit	Edit
	GL-Edit	Edit	Edit
	AR-Rpt	Rpt	Rpt
	AP-Rpt	Rpt	Rpt
	GL-Rpt	Rpt	Rpt
	AR-batch	Batch	Batch
	AP-batch	Batch	batch
	GL-batch	Batch	batch
POS	POS-Edit	Edit	edit
	POS-Rpt	Rpt	rpt
	POS-batc	Batch	batch

In this mapping, objects that belong to multiple product modules coexist in the same subdirectory. If you are developing your system from scratch, do not use this type of directory hierarchy in your application. Instead, use a functional hierarchy if possible.

### 3.6.3. Directories by Application Group and Component Type

It often makes sense to combine a functional hierarchical system architecture with one based on component types. A combined directory structure may look like this:

Directory	Directory Content
/abc_system	ABC accounting and POS system directory
/abc_system/ar/edit	Accounts Receivable data entry procedures
/abc_system/ar/rpt	Accounts Receivable report procedures
/abc_system/ar/batch	Accounts Receivable batch procedures
/abc_system/ap/edit	Accounts Payable data entry procedures
/abc_system/ap/rpt	Accounts Payable report procedures
/abc_system/ap/batch	Accounts Payable batch procedures
/abc_system/gl/edit	General Ledger data entry procedures
/abc_system/gl/rpt	General Ledger report procedures
/abc_system/gl/batch	General Ledger batch procedures
/abc_system/pos/edit	Point of Sale data entry procedures
/abc_system/pos/rpt	Point of Sale report procedures
/abc_system/pos/batch	Point of Sale batch procedures

This configuration hierarchy can be mapped into Roundtable as shown below:

Product	Prod Module	Workspace Module	Subdirectory
Accounting	AR-Edit	AR-Edit	ar/edit
	AP-Edit	AP-Edit	ap/edit
	GL-Edit	GL-Edit	gl/edit
	AR-Rpt	AR-Rpt	ar/rpt
	AP-Rpt	AP-Rpt	ap/rpt
	GL-Rpt	GL-Rpt	gl/rpt
	AR-batch	AR-Batch	ar/batch
	AP-batch	AP-Batch	ap/batch
	GL-batch	GL-Batch	gl/batch
POS	POS-Edit	POS-Edit	pos/edit
	POS-Rpt	POS-Rpt	pos/rpt
	POS-batc	POS-Batch	pos/batch

### 3.6.4. Using Subtypes to Extend the Configuration Hierarchy

If you have an especially complex configuration hierarchy, use subtypes to model some subdirectory organization rules. You define subtypes to ascribe attributes to PCODE objects. One of the attributes you can define is a subdirectory in which a file associated with the PCODE object will be stored.

Consider again the directory structure from the previous section:

Directory	Directory Content
/abc_system	ABC accounting and POS system directory
/abc_system/ar/edit	Accounts Receivable data entry procedures
/abc_system/ar/rpt	Accounts Receivable report procedures
/abc_system/ar/batch	Accounts Receivable batch procedures
/abc_system/ap/edit	Accounts Payable data entry procedures
/abc_system/ap/rpt	Accounts Payable report procedures
/abc_system/ap/batch	Accounts Payable batch procedures
/abc_system/gl/edit	General Ledger data entry procedures
/abc_system/gl/rpt	General Ledger report procedures
/abc_system/gl/batch	General Ledger batch procedures
/abc_system/pos/edit	Point of Sale data entry procedures
/abc_system/pos/rpt	Point of Sale report procedures
/abc_system/pos/batch	Point of sale batch procedures

This configuration hierarchy can be mapped into Roundtable more simply with the use of subtypes as shown below:

Product	Prod Module	Workspace Module	Subdirectory
Accounting	AR	AR	ar
	AP	AP	ap
	GL	GL	gl
POS	POS	POS	pos

Subtype	Subdirectory
EditProc	edit
EditIncl	edit
BatchProc	batch



Subtype	Subdirectory
BatchIncl	batch
RptProc	rpt
RptIncl	rpt

When you create a new PCODE object in a module, Roundtable constructs part of the path to the object based on the product module assignment. It then constructs the rest of the path based on the subtype you assigned to the object. For example, if you created a new data entry program called order.p for the AR product module and assigned it a subtype of EditProc, Roundtable performs the following logic to decide what directory to put it in:

1. Given the assignment of the AR product module, Roundtable finds the AR workspace module.
2. The directory associated with the AR product module is "ar", so Roundtable adds this to the workspace root path to create "/abc\_system/ar".
3. Roundtable finds the subtype EditProc and adds its associated directory, "edit", to the directory path to create "/abc\_system/ar/edit".

Using subtypes in this manner can be useful when it is necessary to map an existing configuration hierarchy. However, you should avoid this style of configuration hierarchy because it leads to duplication in the subtypes. In the example above, there might be no difference between the EditProc and BatchProc subtype specifications except for their associated subdirectories.

For more detailed information on subtypes, see Section 3.9, “Subtypes” [3–21].

### 3.7. Products

Once you have defined a strategy for mapping your existing or new system into Roundtable, you must create the necessary products and product modules. This section provides detailed instructions for adding this information to the Roundtable repository.

The following table lists the Products Menu fields:

Field	Description
Product	A field for the product code.
Name	A field for the product name.

The Products Menu includes the following options:

Option	Description
<b>Select</b>	Go to the Product Modules Menu.
<b>Add</b>	Add a product.
<b>Edit</b>	Edit Product Description.
<b>Delete</b>	Delete an unpopulated Product.
<b>Find</b>	Find a Product.
<b>Print</b>	Print a Product.
<b>Quit</b>	Return to the previous menu

### 3.7.1. Product Modules Menu

The following table lists the Product Module Menu fields:

Field	Description
Pmodule	A product module code that is unique across all products.
Description	Description of the product module.
Module	Workspace module definition code.
Description	Description of the workspace module.
Directory	Relative directory from the workspace root directory in which source for objects assigned to workspace modules will be stored. Further subdirectories below this directory can be specified in the PCODE subtype.
r-directory	This is an optional directory in which you want to store .r code. If this directory is not explicitly specified, the compile process stores .r code in the same directory in which the source exists.
Naming Code	A code that can be used by the naming program for a PCODE subtype. This is optional and only used if naming programs you have defined require it.

The Product Modules Menu includes the following options:

Option	Description
<b>Add</b>	Add a new product module. Product module codes are unique. A product module code cannot be used twice, even if it is defined under different products. When adding a product module, entering a new workspace module code in the WS module field will result in a new workspace module definition being created. Entering an existing workspace module code will assign the

Option	Description
	product module to the workspace module. One or more product modules can be assigned to a workspace module. These assignments cannot be changed once made. The workspace module definition is displayed in the Workspace Module Details Frame.
<b>Edit</b>	Edit the Product Description. This option allows you to edit the product module's Description field and the contents of the Workspace Module Details Frame. Be very careful with editing the Directory field as Roundtable will not move code assigned in the directory to any new directory specified.
<b>Delete</b>	Delete a product module if no objects are assigned to it.
<b>Find</b>	Find a product module.
<b>Print</b>	Print Product Module Report.
<b>Quit</b>	Return to the Products Menu.

See Section 3.8, “Workspace Module Definitions” [3–20] for a description of the Workspace Module Details fields.

### 3.7.2. Adding a Product

Follow these steps to add a new product:

1. From the Main Menu, choose Products.
2. If Roundtable security is enabled and you are not logged in as the sysop user, Roundtable prompts for the sysop account password. Type the sysop password. (If you do not supply this password, you are permitted only to view information on the following screen.)
3. Press **ENTER**. The Roundtable Products Menu appears.
4. Choose **Add**.
5. In the Product field, type the code for the new product. The alphanumeric code should describe the major function of the product (acct for accounting, ops for operations, etc.). If your site number is not 0, you must enter the three-digit site number as a prefix for the code. If your site number is 0, you cannot begin your code with a number.
6. Press **ENTER**. Type the name or a description of the new product in the Name field.
7. Press go to create the new product.
8. Add product modules belonging to the new product. For information on how to add product modules, see the Section 3.7.6, “Adding a Product Module” [3–17] section.

You can add product modules at a later date if necessary.

9. Choose **Quit** to return to the previous menu.

### 3.7.3. Editing a Product Name

You can only change the product name field for a product. If you want to change the product code, delete the product and re-add it.

Follow these steps to edit a product name:

1. From the Main Menu, choose **Products**.
2. If Roundtable security is enabled and you are not logged in as the sysop user, Roundtable prompts for the sysop account password. Type the sysop password. (If you do not supply this password, you are only permitted to view information in the window.)
3. The Product Menu appears.
4. Select the product name you want to edit.
5. Choose **Edit**.
6. Edit the description of the product, then press the **GO** key.
7. Choose **Quit** to return to the previous screen.

### 3.7.4. Deleting a Product

You must delete all product modules within a product before you delete the product itself.

Follow these steps to delete a product:

1. From the Main Menu, choose **Products**.
2. If Roundtable security is enabled and you are not logged in as the sysop user, Roundtable prompts for the sysop account password. Type the sysop password. (If you do not supply this password, you are only permitted to view information in the window.)
3. The Products Menu appears.
4. Select the product you want to delete.
5. Choose Delete. The message, "Delete this record?" appears.
6. Type **Yes** to delete the product or type **No** to cancel this operation.
7. Press the **GO** key.

8. Choose **Quit** to return to the previous screen.

### 3.7.5. Product Report

The Product Report prints a list of the contents of the product.

Follow these steps to print the report:

1. From the Main Menu, choose **Products**.
2. If Roundtable security is enabled and you are not logged in as the sysop user, Roundtable prompts for the sysop account password. Type the sysop password. (If you do not supply this password, you are only permitted to view information in the Products window. This is fine if you only want to print a report.)
3. The Products Menu appears.
4. Select the product for which you want to print a report.
5. Choose **Print**. The Product Report dialog box appears.
6. Select the report options you want, then press **GO**.
7. Choose **Quit** to leave the Products window.
8. See Appendix A, *Interfaces* [A–1] for an example of each report in the system.
9. The following table lists the Product Report fields:

Field	Description
Show Workspace Usage?	Prints a list of the workspaces that include the product modules in this product.
Show Objects?	Prints the latest object version defined in each product module.
Show Object Details?	Prints the object version description.

### 3.7.6. Adding a Product Module

After you create a product, follow these steps to add the product modules:

1. From the Main Menu, choose **Products**.
2. If Roundtable security is enabled and you are not logged in as the sysop user, Roundtable prompts for the sysop account password. Type the sysop password. (If you do not supply this password, you are only permitted to view information in the window.)

3. The Products Menu appears.
4. Select the product to which you want to add a product module.
5. Choose **Select**. The Product Modules Menu appears.
6. Choose **Add**.
7. Type the Pmodule. This alphanumeric code should describe the function of the module. If your site number is not 0, you must enter the three-digit site number as a prefix for the product module. If your site number is 0, you cannot begin your code with a number. Press **ENTER**. The pointer appears in the Description field.
8. Fill in the Description field for the product module Type the name of an existing workspace module definition to associate with the new product module. If no existing workspace module definition is appropriate for the new product module, you can create one at this time by typing the name of a new workspace module definition. The primary purpose of the workspace module definition is to specify the physical location of code belonging to the product module associated with the workspace module definition.
9. Complete the fields for the Workspace Module Details.
10. Press the **GO** key to complete your entries.
11. Choose **Quit** to return to the previous screen.

### 3.7.7. Editing a Product Module

You can change the product module description field but not the product module code. To change a product module code, delete the product module and re-add it with the new code. You cannot delete a product module if any objects are assigned to the product module code.

Follow these steps to edit a product module description:

1. From the Main Menu, select **Products**.
2. If Roundtable security is enabled and you are not logged in as the sysop user, Roundtable prompts for the sysop account password. Type the sysop password. (If you do not supply this password, you are only permitted to view information in the window.)
3. Press **ENTER**. The Products screen appears.
4. Select the product that contains the product module you want to edit. Choose **Select**.
5. From the Product Modules list, select the product module to edit.
6. Choose **Edit**.
7. Change the fields for the product module as necessary.

8. Press **GO**.
9. Choose **Quit** to leave the screen.

### 3.7.8. Deleting a Product Module

If you have ever used the product module, you cannot delete it (even if you are not using the product module now).

Follow these steps to delete a product module:

1. From the Main Menu, choose Products.
2. If Roundtable security is enabled and you are not logged in as the sysop user, Roundtable prompts for the sysop account password. Type the sysop password. (If you do not supply this password, you are only permitted to view information in the window.)
3. Choose **ENTER**. The Products Menu appears.
4. Select the product that contains the product module you want to delete.
5. Choose Select. The Product Modules Menu appears.
6. From the Product Modules list, select the product module to delete.
7. Choose Delete. The message, "Delete this record" appears.
8. Type Yes to delete the product module.
9. Choose **Quit** to leave the screen.

### 3.7.9. Product Module Report

The Product Module Report prints a detailed report of the objects in a product module, including the object code, description, version number, status, subtype, and additional notes for each object.

Follow these steps to print the Product Module Report:

1. From the Main Menu, choose Products.  
  
If Roundtable security is enabled and you are not logged in as the sysop user, Roundtable prompts for the sysop account password. Type the sysop password. (If you do not supply this password, you are only permitted to view information in the window. This is fine if you only want to print the report.)
2. Press **ENTER**. The Products Menu appears.

3. From the Product list, select the product containing the product module for which you want a report.
4. Choose Select. The Product Modules Menu appears.
5. Choose Print. The Product Module Report dialog box appears.
6. Select the report options you want, then press the **GO** key to print the report.
7. Choose **Quit** to leave the Product Modules screen.

The following table lists the Product Module Report fields (see table below).

Field	Description
Show Workspace Usage?	Prints a list of the workspaces that include this product module.
Show Objects?	Prints the latest object version defined in each product module.
Show Object Details?	Prints the object version description.

### 3.8. Workspace Module Definitions

The primary purpose of workspace module definitions is to specify the physical location of code relative to the workspace root directory. Essentially they contain a relative subdirectory that is concatenated to the workspace root directory to render a full path to location of objects in the system. Workspace module definitions allow you to map an arbitrarily deep level of directory structures to the two-level configuration hierarchy provided by Roundtable. While the workspace root directory is always different among the workspaces managed by the system, workspace module definitions are consistent across all workspaces. Workspace module definitions are associated with product modules in a one-to-many relationship. A single workspace module definition can be associated with zero or more product modules. Unassigned workspace module definitions are ignored by the system.

The following table lists the Workspace Module Detail fields:

Field	Description
Module	Workspace module definition code.
Description	This is a description of the workspace module.
Directory	Relative directory from the workspace root directory in which source for objects assigned to workspace modules will be stored. Further subdirectories below this directory can be specified in the PCODE subtype.



### 3.8.1. Adding a Workspace Module Definition

In UNIX Roundtable, you create a new workspace module definition record at the same time you add a new product module record. See Section 3.7.6, “Adding a Product Module” [3–17]. Follow the steps described in that section to add a new product module. However, when you complete the module field, enter a workspace module definition code that does not yet exist. Roundtable automatically creates the new workspace module definition record for you.

### 3.8.2. Editing a Workspace Module Definition

You can edit any of the fields in the workspace module and you can safely change the description of the workspace module definition.

In UNIX Roundtable, you edit a workspace module definition record at the same time you edit a product module record. See Section 3.7.7, “Editing a Product Module” [3–18] for more information. Follow the steps described in this section to edit a product module and its related workspace module definition fields. Remember that a workspace module definition can be used by more than one product module. Therefore, editing the workspace module definition fields can impact more than one product module.

### 3.8.3. Deleting a Workspace Module Definition

In UNIX Roundtable, the workspace module definition records cannot be deleted. They have no impact on any of Roundtable’s functions if they are not being used by any product modules.

## 3.9. Subtypes

Subtypes control the naming, management, creation, and storage of PCODE objects. Using subtypes can help enforce coding standards and reduce the number of objects tracked in your system. They can also make training new programmers on your system much easier since Roundtable provides reminders about the requirements of each subtype. By using subtypes, you reduce the number of repetitive steps necessary for creating new PCODE objects. When creating a subtype, you can:


- Specify a naming program, which tests the naming convention of the object to ensure consistency.
- Specify a build program to run when creating an object, like a screen generator.
- Specify a template to use when Roundtable builds an object.
- Specify up to nine parts (individual files), related by a predefined naming convention, that Roundtable will treat as one object. Then, when checking out objects, you check out only one object instead of nine. This ensures that an object with multiple parts is treated as a single entity.

### 3.9.1. PCODE Subtype Menu

Use the Subtype screen to define subtypes.

The following table lists the PCODE Subtype Menu fields:

Field	Description
Subtype	The name of the subtype.
Assigned?	Roundtable indicates this field as Yes once the subtype is used by an object. Once a subtype has been assigned to an object, the subtype cannot be deleted and some editing operations are not allowed.
Naming Program	A field for the full pathname of a naming program. Roundtable has an example program (rtb_bname.p) that takes the Naming Code defined in the Workspace Modules Details screen and adds a four-digit number to the end. For example, if the Naming Code was "ord", then the default names for objects created in this module would be ord0001.p, ord0002.p, etc. You can also use your own naming program. If you use your own program, type its entire pathname. Examine the rtb_bname.p procedure for the parameters and return values you need to use. The use of a naming program is entirely optional. See Appendix A, <i>Interfaces</i> [A–1] for more information on naming programs.
Build Program	A fill-in for the name of the build program, which creates source files for objects of this subtype. The build program can be simple or complex. For example, a build program can be a full blown screen generator or it can copy a template into the source file.
Edit Program	A fill-in for the name of the edit program. For example, if objects of this subtype are to have a specific format, the edit program can be a custom editor.
Program?	Type Yes to indicate whether objects of this subtype are PROGRESS -compilable programs. (Include files are not considered programs.)
r-code Dir	<p>Enter the relative r-code directory for the subtype. With the addition of the r-code directories by subtype, the path for r-code is determined as follows:</p> <ol style="list-style-type: none"><li>1. If r-code directory defined for workspace module only:</li></ol> <pre>&lt;wspace-path&gt;/ \ &lt;module-rcode-path&gt;/object.r</pre>

Field	Description
	<p>2. If r-code directory defined for subtype only:</p> <pre>&lt;wspace-path&gt;/ \ &lt;subtype-rcode-path&gt;/object.r</pre> <p>3. If r-code directory for both workspace module AND subtype:</p> <pre>&lt;wspace-path&gt;/ \ &lt;module-rcode-path&gt;/ \ &lt;subtype-rcode-path&gt;/object.r</pre> <div>  <p>Roundtable will not move .r code for you if you change the value of the .r code directory for this subtype. If you change this value, you will have to move any subtype .r code manually, or recompile programs of this subtype.</p> </div>
Parts	A subtype can be comprised of up to nine system files.
Description	A fill-in for the description of the subtype part. Each subtype can have up to nine individual file parts. This feature is not commonly used in the PROGRESS Version 7 and Version 8 environments.
Directory	A fill-in for the directory, relative to the workspace module directory, where the part file is stored. Leave this field blank if the part is stored directly in the workspace module directory.
Template	A fill-in for the full path and filename of a template file used to build the part file. This optional field is used only in conjunction with the build program.
Suffix	A fill-in for the suffix added to the end of the part filename (prior to the extension). If left blank, no suffix is added to the object's name.
Extension	A fill-in for the extension of the part (for example, "p" for a program). This optional field is commonly used to enforce a naming convention. For instance, a .p extension is used for most PROGRESS procedures created with the procedure editor and a .w extension is used for most procedures generated by the UIB.
Compiles	A logical (yes/no) field that designates whether or not the subtype

Field	Description
	part should be compiled.

### 3.9.2. Subtype Theory

In the PROGRESS Version 7 and Version 8 programming environments, multiple code parts are not commonly used. Typically, you define subtypes with only one part each. Possible subtype definitions include:

Subtype	Program	Part
Program	Yes	P
Include	No	I

Defining multiple-part subtypes is generally useful only when you are using templates and code generators, with each main program (.p) calling a number of its own, private include files.

### 3.9.3. Subtype Example — Part I

PROGRESS code (PCODE) objects are assigned a user-defined subtype that controls the naming, management, creation, and storage of one to nine text files (parts) belonging to the object. Consider the following simple subtype definition:

**Subtype:** Program  
**Program:** Yes  
**Naming Program:** nameprog.p  
**Build Program:** bldprog.p

Part #	Part	Suffix	Ext.	Template	Description
1	P		P	program.tpl	Simple program

This subtype describes programs that consist of a single text file. New programs of this subtype are named automatically by the nameprog.p procedure. In addition, the new program is constructed by calling the bldprog.p procedure. The bldprog.p procedure uses the template program.tpl to construct the program. The bldprog.p procedure is supplied by the user. For example, the bldprog.p procedure might do nothing more than copy the contents of program.tpl into the newly created program file. The Build Program, Naming Program, and Template are all optional and can be omitted.

### 3.9.4. Subtype Example — Part II

In the following example, a simple include file subtype is defined:

<b>Subtype:</b>	Include
<b>Program:</b>	No
<b>Naming Program:</b>	-
<b>Build Program:</b>	-

Part #	Part	Suffix	Ext.	Template	Description
1	1				Simple include

With just the program and include subtypes described so far, you could build any PROGRESS application. However, much more exciting opportunities exist. Consider the following subtype definition to define programs using a generic scrolling table template:

<b>Subtype:</b>	Table
<b>Program:</b>	Yes
<b>Naming Program:</b>	-
<b>Build Program:</b>	bldtbl.p

The overly simplified program in this figure demonstrates why the table subtype is useful.

```
EXAMPLE PROGRAM OF THIS SUBTYPE
/* ordlines.p */
/* tbl.i is a general purpose
table template used by many programs */
{tbl.i
ordlines.d
ordlines.u
ordlines.a
}
```

Without the table subtype, the four ordlines.\* text files would have to be separate objects, making them cumbersome to manage. By formalizing the relationship among these text files, Roundtable can treat them as a single object.

Using subtypes can help enforce coding standards and vastly reduce the number of objects tracked in your system. They can also make training new programmers on your system much easier because Roundtable reminds them of the requirements of each subtype.

### 3.9.5. Subtype Example — Part III

It is possible to use part suffixing instead of extensions to name the parts of an object. Consider the example below:

<b>Subtype:</b>	Table
<b>Program:</b>	Yes
<b>Naming Program:</b>	-
<b>Build Program:</b>	bldtbl.p

Using this scheme, the text files associated with an object named orders.p would be:

**orders.p**

**orders-d.i**

**orders-u.i**

**orders-a.i**

Suffix naming is less flexible than extension naming because the length of the suffix reduces the length of the base name (in this case orders) usable by the programmer. For example, a two-character suffix of "-d" leaves only six characters available for the base name if you want to conform to the eight-character DOS naming convention. When you add a PCODE object, Roundtable tests the naming conventions based on these fields. For example, if you designate ".p" as the extension for the program subtype part, you cannot create a Program part with a suffix of ".u". Roundtable displays an error message until you enter the correct suffix.

Subtypes must be created before you add PCODE objects to your PROGRESS application. After creating the subtypes, use them to create PCODE objects. You can also edit and delete subtypes and print the Subtypes Report.

### 3.9.6. Adding a Subtype

First, follow these steps to enter the basic information for the subtype.

1. From the Main Menu, select **Code Subtypes**. If Roundtable security is enabled and you are not logged on as a sysop user, you are prompted for the sysop account password. enter your sysop password.



If you do not supply this password, you are permitted only to view subtype information.

2. The PCODE Sub-type Definitions screen appears. To create a new subtype, select **Add** from the Pcode Sub-type Menu.
3. In Subtype field, enter the new alphanumeric subtype code and then click **OK**.



If your site number is not 0, you must enter the three-digit site number as a prefix for the code. If your site number is 0, you cannot begin your code with a number.

4. Fill in the necessary information in the Subtype panel of the PCODE Sub-type Definitions screen.
5. Selecting defaults on the Pcode Sub-Type Menu opens the Code Subtype Config Defaults dialog box. Select the default values to be used when creating an object that uses the subtype.
6. After you have selected the default values you want to use, choose OK to return to the PCODE Sub-type Definitions screen.
7. Choose Select on the Pcode Sub-Type Menu to enable the Sub-type Parts panel.
8. Edit each needed part by selecting the part number and selecting Edit from the Pcode Sub-Type Menu.



The part code is usually the extension of the part you are creating (for example, "p" for a program part).

9. In the Part Details panel, edit the parts information for the subtype part.
10. For each subtype part (parts 2 through 9) that is compilable (such as the client-proxy of an SDO), set Compiles to **Yes**.
11. Repeat these steps for each part required in the subtype or click **Done** to close the window.

### 3.9.7. Editing a Subtype or Subtype Part

You can edit some of the fields of a subtype after it has been created.

Follow these steps to edit a subtype.

1. From the Main Menu, choose Code Sub-types.
2. If Roundtable security is enabled and you are not logged in as the sysop user, Roundtable prompts for the sysop account password. Type the sysop password. (If you do not supply this password, you are permitted only to view information in the window.)

3. The PCODE Sub-Type Definitions Menu appears.
4. Select the subtype you want to edit.
5. Choose Edit and make changes.
6. Press the **GO** key to complete your changes.
7. Choose Select to go to the Sub-Type Parts Menu.
8. Select the part to edit and choose Edit.
9. Press the **GO** key to complete your changes.
10. Choose **Quit** to return to the previous screen.

### 3.9.8. Deleting a Subtype

Note that you cannot delete a subtype after it has been assigned to one or more objects. The Assigned field at the top of the Sub-types screen indicates whether the sub-type has been assigned.

Follow these steps to delete a subtype:

1. From the Main Menu, choose Code Sub-types.
2. If Roundtable security is enabled and you are not logged in as the sysop user, Roundtable prompts for the sysop account password. Type the sysop password. (If you do not supply this password, you are permitted only to view information in the window.)
3. The PCODE Sub-Type Definitions Menu appears.
4. Select the subtype to delete.
5. Choose Delete. The message, "Delete this record" appears.
6. Type Yes to delete the sub-type or type No to cancel this operation.
7. Choose **Quit** to return to the previous screen.

### 3.9.9. Sub-types Report

The Sub-types Report provides a detailed review of each subtype. The report includes all the fields listed on the Roundtable Sub-types screen.

Follow these steps to print a Sub-types report:

1. From the Main Menu, choose Code Sub-types.

If Roundtable security is enabled and you are not logged in as the sysop user,



Roundtable prompts for the sysop account password. Type the sysop password. (If you do not supply this password, you are permitted only to view information in the window.)

The PCODE Sub-Type Definitions menu appears.

2. Choose Print. The Sub-type Parts Print Destination dialog box appears.
3. Complete the Print fields.
4. Choose **Quit** to close the screen.

### 3.9.10. Build Program

This field contains the name of a program that will create source files for the object. Build programs are optional but very useful. A build program can be as simple as commands to copy templates into the new object's source files or as sophisticated as a full-blown screen generator.

To make the build program simple to implement, these shared variables are provided:

```
DEFINE SHARED VARIABLE Urtb-object AS \
  CHARACTER NO-UNDO.
DEFINE SHARED VARIABLE Urtb-part AS \
  CHARACTER EXTENT 10 NO-UNDO.
DEFINE SHARED VARIABLE Urtb-part-desc AS \
  CHARACTER EXTENT 10 NO-UNDO.
DEFINE SHARED VARIABLE Urtb-path AS \
  CHARACTER EXTENT 10 NO-UNDO.
DEFINE SHARED VARIABLE Urtb-name AS \
  CHARACTER EXTENT 10 NO-UNDO.
DEFINE SHARED VARIABLE Urtb-tmpl AS \
  CHARACTER EXTENT 10 NO-UNDO.
DEFINE SHARED VARIABLE Urtb-num-parts AS \
  INTEGER NO-UNDO.
DEFINE SHARED VARIABLE Urtb-sub-type AS \
  CHARACTER NO-UNDO.
DEFINE SHARED VARIABLE Urtb-userid AS \
  CHARACTER NO-UNDO.
DEFINE SHARED VARIABLE Urtb-task-num AS \
  INTEGER NO-UNDO.
DEFINE SHARED VARIABLE Urtb-task-desc AS \
  CHARACTER NO-UNDO.
DEFINE SHARED VARIABLE Urtb-propath AS \
  CHARACTER NO-UNDO.
DEFINE SHARED VARIABLE Urtb-ws-propath AS \
  CHARACTER NO-UNDO.
DEFINE SHARED VARIABLE Urtb-mod-dir AS \
```

CHARACTER NO-UNDO.

Variable	Description
part-num	Integer variable with a value from 1 to 9.
Urtb-object	Name of the object without extension.
Urtb-part[part-num]	See the next section defining Part Code.
Urtb-part-desc[part-num]	See the next section defining Description.
Urtb-path[part-num]	Full pathname of the object part in the workspace.
Urtb-name[part-num]	Name of the part less its path.
Urtb-tmpl[part-num]	Full pathname to the template for the part.
Urtb-num-parts	Number of parts.
Urtb-sub-type	Sub-Type Code.
Urtb-userid	Current Userid.
Urtb-task-num	Current Task Number.
Urtb-task-desc	Description of the current task.
Urtb-propath	PROPATH for Roundtable.
Urtb-ws-propath	PROPATH for workspace.
Urtb-mod-dir	Full pathname to the module directory.

### 3.9.11. Edit Program

Sometimes it is necessary to keep track of versions of data in an application. This data could be static code tables, menu content, program generation data, or seed data to be delivered with an application. It is possible to manage this data by developing user-defined edit programs specific to the application data on which version control is desired. This field contains the name of such a program.

A specialized edit program should follow these general guidelines:

- The only way one should be able to run one of these programs is from the Roundtable environment. This will prevent untracked changes from being made in the application data.
- Each program should have an Edit, View only, Import, Export, and Delete mode:
- Edit Mode should allow the entry or editing of the application data. Roundtable will run the program in Edit Mode when the object is WIP and owned by the current task.
- View Mode should allow only the viewing of the application data. Roundtable will run

the program in View mode when the object is not WIP or belongs to a task other than the current task.

- Import mode should delete the current data (if any) in the application database and then load the object by reading in new content from one or more text files having formats specific to the type of data being managed. Roundtable will run the program in Import mode when All PCODE Data Imports option is chosen from the WS-config menu or just prior to the first Edit of the program after a recent assignment of the object to the workspace. The Import process assigns objects to a workspace. It is also possible to manually assign a version of an object to the workspace. For more information about the PCODE Data Imports program, see Section 4.7.7, “PCODE Data Imports” [4–32].
- Export mode should create one or more text files containing the data stored in the application database. This process is run prior to the completion of the object and the registration of the object’s text files into the repository.
- Delete mode should cause the deletion of the data associated with the object from the application database.
- An optional View Hist mode can be supported by the application edit program if desired. This mode would allow the view of a prior version of the object. When you select a previous version of an object and choose the Edit option off the PCODE Object Menu, Roundtable creates one or more temporary files by retrieving the requested version from the repository. The edit program in View Hist mode should allow the user to review this earlier version of the object without corrupting the current application database. This mode is optional.
- An optional Xref mode can be supported by the application edit program if desired. This mode allows the program to generate Roundtable Xref entries. This option can be especially useful for applications where the application menus are driven by tables in the application database. This mode is optional.

Here is a sample application specification to illustrate how an edit program interface can be implemented.

### 3.9.11.1. MENU MANAGER

This edit program allows the Roundtable system to manage menu tables in an application database.

The application menus are contained in two tables in the application database:

menu\_header      Describes the menu and its placement on the screen

```
menu name x(32)
menu title x(76)
menu row integer
menu col integer
```

menu\_item      Describes each menu item

```
menu name x(32)
item seq integer
item type P)rogram, M)enu
item name x(32)
item desc x(76)
```

A PCODE Subtype called menuman is created in Roundtable. An edit program named menuman.p is specified. This program has the following prototype definition:

```
/* menuman.p MENU MANAGER */
DEFINE INPUT PARAMETER Pmode AS
    CHARACTER NO UNDO.
DEFINE OUTPUT PARAMETER Perror AS
    CHARACTER NO UNDO.

/* Shared variable definitions passed by
   Roundtable See manual for definitions */
DEFINE SHARED VARIABLE Urtb object AS
    CHARACTER NO UNDO.
DEFINE SHARED VARIABLE Urtb part AS
    CHARACTER EXTENT 10 NO UNDO.
DEFINE SHARED VARIABLE Urtb part desc AS
    CHARACTER EXTENT 10 NO UNDO.
DEFINE SHARED VARIABLE Urtb path AS
    CHARACTER EXTENT 10 NO UNDO.
DEFINE SHARED VARIABLE Urtb name AS
    CHARACTER EXTENT 10 NO UNDO.
DEFINE SHARED VARIABLE Urtb tmp1 AS
    CHARACTER EXTENT 10 NO UNDO.
DEFINE SHARED VARIABLE Urtb num parts AS
    INTEGER NO UNDO.
DEFINE SHARED VARIABLE Urtb sub type AS
    CHARACTER NO UNDO.
DEFINE SHARED VARIABLE Urtb userid AS
    CHARACTER NO UNDO.
DEFINE SHARED VARIABLE Urtb task num AS
    INTEGER NO UNDO.
DEFINE SHARED VARIABLE Urtb task desc AS
    CHARACTER NO UNDO.
DEFINE SHARED VARIABLE Urtb propath AS
    CHARACTER NO UNDO.
```

```

DEFINE SHARED VARIABLE Urtb ws propath AS
    CHARACTER NO UNDO.
DEFINE SHARED VARIABLE Urtb mod dir AS
    CHARACTER NO UNDO.

IF Pmode = "Edit" THEN
DO: /* Allow user to update data related to object
    in the application database */
END.
ELSE IF Pmode = "View" THEN
DO: /* Allow user to view data related to object in
    the application database */
END.
ELSE IF Pmode = "Import" THEN
DO: /* Delete current data. Load new data from
    text files. */
END.
ELSE IF Pmode = "Export" THEN
DO: /* Export data from application database to
    text files */
END.
ELSE IF Pmode = "Delete" THEN
DO: /* Delete data related to object from the
    application database */
END.
ELSE IF Pmode = "View Hist" THEN
DO: /* Allow user to view data placed in temporary
    text files retrieved from repository */
END.
ELSE IF Pmode = "Xref" THEN
DO: /* Create rtb_xref records based on the content
    of the object's data in app database */
END.
ELSE
Perror = "Unknown Mode".
RETURN.

```

### 3.9.12. System Parameters

You can use the System Parameters dialog box to enter your registered user name and number. The site number is used if you have implemented Roundtable to manage software development at multiple sites.

Follow these steps to set the system parameters:

1. From the Main Menu, select Admin.
2. Then select System Parameters from the Admin Main Menu.

3. Fill in the fields, then press the **GO** key.
4. Choose **Quit** to return to the previous screen.

If Roundtable security has been enabled and you are not logged in as sysop, then you are prompted for the sysop password. You must enter the sysop password to edit or view the site information.

The System Parameters dialog box appears.

The following table lists the System Parameters dialog box fields:

Field	Description
Registered User	A fill-in for the name of the registered user, normally the organization's name, as this value is stored once in the repository. Do not enter a name for each client.
Registration #	Not currently used.
Site#:	The Site Number value is used only when you have more than one Roundtable repository and must move information from one repository to another. Using multiple repositories is an advanced feature of the Roundtable system. See Section 1.10, "Distributed Development" [1–26].

### 3.10. Security

Roundtable provides a complete security framework to allocate responsibility for activities to different users of the system. A user can have different security access privileges in each workspace. Roundtable security is managed through the following windows:

- User Maintenance Screen
- Group Access Screen
- Workspace Selection-Security Access for User Screen (for access assignments)

When you first install Roundtable, security is not active. Activate security by adding Roundtable users in the User Maintenance screen. After adding the users, you can add group access records. Each group access record consists of a group access code you provide and a list of system activities. Use the Workspace Selection-Security for User screen to associate one or more users with one or more group access records within a specified workspace.



Users registered in the Roundtable system are recorded in the `_user` file of the Roundtable database. These users are not recorded in your application databases by Roundtable. When connecting to your application databases, Roundtable passes your Roundtable user ID and password as part of the database connection process. You can set up your database connection parameters to pass alternate values using the `-U` and `-P` connect options. For more information about database connection parameters, see Section 6.7, “PDBASE Objects” [6–14].

### 3.10.1. User Maintenance Screen Description

Maintain users in the User Maintenance screen:

This window allows you to add or delete users, edit user names, and set user passwords. The `sysop` user is a special user found in every Roundtable system. The `sysop` user enjoys unlimited access to all system functions and exclusive access to many important system functions. When you execute a function in the system that requires `sysop` privileges, Roundtable prompts you for the `sysop` user ID and password.

The system creates the `sysop` user automatically when you enter the User Maintenance Screen for the first time. You are prompted for a `sysop` password.

### 3.10.2. Adding, Editing, or Deleting a User

Follow these steps to add, edit, or delete a user:

1. From the Main Menu, select Users.

If Roundtable security is enabled and you are not logged in as the `sysop` user, you are presented with the following menu: Choose Maintain Users (Sysop).

Roundtable prompts for the `sysop` account password. Type the `sysop` password. (If you do not supply this password, you are not permitted to view the User Maintenance screen.) Press the **ENTER** key to continue. The User Maintenance Menu appears.

2. Choose Add to add an user.

**OR**

Choose Edit to edit an user’s name.

**OR**

Choose Set-Password to set or change a password.

**OR**

Choose Delete to delete a user entirely.

3. Complete the fields and press the **GO** key.
4. Choose **Quit** to return to the Main Menu.

### 3.10.3. Changing Your Password

You can only change your password if users have been defined in Roundtable. Follow these steps to change your password:

1. From the Main Menu, choose Users.
2. Choose Change Password. Roundtable prompts you for your new password.

### 3.10.4. Group Access Screen Description

Use this screen to define a group access record that you will later assign to one or more users in specified workspaces. Each group access record contains a group access code that you provide and a list of allowable system activities.

The selection list contains columns displaying the group access codes. Type a Y to select a group access code. The following table describes the group access codes:

Code	Description
<b>Orphans Allowed</b>	Users are allowed to create orphan versions.
<b>PCODE Objects</b>	Users are allowed to work on PCODE Objects.
<b>PDBASE Objects</b>	Users are allowed to work on PDBASE Objects.
<b>PFILE Objects</b>	Users are allowed to work on PFILE Objects.
<b>PFIELD Objects</b>	Users are allowed to work on PFIELD Objects.
<b>DOC Objects</b>	Users are allowed to work on DOC Objects.
<b>Update Schema</b>	Users are allowed to perform the schema update process.
<b>Import</b>	Users are allowed to perform the import process.
<b>Task Create</b>	Users are allowed to create new tasks.
<b>Task Complete</b>	Users are allowed to complete tasks.
<b>Release Menu</b>	Users are allowed to access the Release menu.
<b>Deployment Menu</b>	Users are allowed to access the Deployment menu.
<b>Change Finder</b>	Users are allowed to perform the global change finder process.



Code	Description
<b>Module</b>	Users are allowed to modify workspace module parameters.
<b>Object</b>	Users are allowed to edit object group and level.

The following is a list of the activity items affect by the security group access codes:

<b>Orphans-Allowed</b>	Versions Menu: Version/Revision/Patch Options
<b>PCODE Objects</b>	Workspace Objects Menu: Add/Del options  PROGRESS Code Object Menu: Edit/Notes/Ver options
<b>PDBASE Objects</b>	More Menu: Build/Copy/Modify with Sed options Workspace Objects Menu: Add/Del options  PROGRESS DBASE Object Menu: Edit/Ver/Load options  Database File Assignments Menu: Add/Edit/Delete options
<b>PFILE Objects</b>	Database Aliases Menu: Add/Delete options Workspace Objects Menu: Add/Del options  PROGRESS File Object Menu: Edit/Ver options  Index Menu: Add/Edit/Delete/Primary options  Index Components Menu: Add/Edit/Delete/Ascending/abbreviate options  Field Assignments Menu: Add/Edit/Delete/Reorder options
<b>PFIELD Objects</b>	Workspace Objects Menu: Add/Del options  Document Object Menu: Edit/Notes/Ver options
<b>Update Schema</b>	Ws-Config Menu: Update Schema option
<b>Import</b>	Ws-Config Menu: Import/Workspace Sources option
<b>Task-Create</b>	Task Maintenance * Menu: Complete options
<b>Releases</b>	Releases Menu: Add/Edit/Delete options

<b>Deployments</b>	Remote Sites Menu: Add/Edit/Delete options
	Deployments Menu: Add/Edit/Delete/Schema/Make/Complete options
<b>Change-Finder</b>	Ws-Config Menu: Global Change Finder Option
<b>Module</b>	Workspace Modules Menu: Edit option
<b>Object</b>	Workspace Objects Menu: Edit option

### 3.10.5. Adding, Editing, or Deleting a Group Access Code

Follow these steps to add, edit, or delete a group access code:

1. From the Main Menu, choose Groups.

If this is the first time you have accessed this function, Roundtable prompts you to create the sysop user account. Enter a sysop password and press **ENTER**.

If you are not currently logged in as the sysop user Roundtable prompts you to enter the sysop password before allowing you to make changes. Enter the sysop password and press **ENTER**. The Group Access menu appears.

2. To add a group access record choose Add.
3. Enter a code for the group access record and press the **GO** key.
4. Type **Y** to indicate which activities users with that group access code can perform.
5. Press the **GO** key to complete the entry.

**OR**

To edit an existing group access record, choose Edit. Then type **Y** to indicate which activities to include or exclude.

Press the **GO** key to complete the edit.

**OR**

To delete a group access record, choose Delete. A question confirming the deletion appears. Type Yes and press the **GO** key to delete the group access record.

6. Choose **Quit** to return to the previous screen.

### 3.10.6. Workspace Security Screen Description

Use the Workspace User Access screen to associate one or more users with one or more access groups for a specific workspace. In short, the user access assignments are done with

this screen.

The first column contains a list of users, and the second column contains group access codes. Each list can be up to 35 characters long.

Do not include the sysop user in your assignments because the sysop user always has full access privileges.

### 3.10.7. Adding Workspace User Access Assignments

Follow these steps to add a new workspace user access assignment:

1. Log in as sysop.
2. From the Workspaces Menu, select the workspace that you want to maintain users for.
3. Choose Users. The User Access for Workspace menu appears.
4. Choose Add.
5. Enter a comma-delimited list of user names and press the **ENTER** key. The pointer appears in the group-list field.
6. Enter a comma-delimited list of groups that the users belong to and press the **GO** key.
7. Choose **Quit** to return to the previous window.

### 3.10.8. Editing User Access for Workspace Assignments

Follow these steps to edit a new workspace user access assignment:

1. Log in as sysop.
2. From the Workspaces Menu, select the workspace that you want to change user access for.
3. Choose Users. The User Access for Workspace menu appears.
4. Select the user access list to edit.
5. Enter the information you want to change and press the **GO** key.
6. Choose **Quit** to return to the previous screen.

### 3.10.9. Deleting User Access for Workspace Assignment

Follow these steps to delete a workspace user access assignment:

1. Log in as sysop.
2. From the Workspaces Menu, select the workspace that you want to change user access for.
3. Choose Users. The Workspace User Access Menu appears.
4. Select the user access list to delete.
5. Choose Delete. The message, "Delete this record" appears. Type **Yes**.
6. Choose **Quit** to return to the previous screen.

### 3.10.10. Viewing Current Privileges

If you are logged in to the system as anyone other than the sysop user you can view your access privileges for the currently selected workspace by following these steps:

1. From the Workspaces Menu, select the workspace that you want to view your security access for.
2. Choose User. The Workspace Selection-Security Access for user screen appears.
3. Press the **SPACE BAR**.
4. Choose **Quit** to return to the previous screen.

### 3.10.11. Sysop Exclusive Access

Certain activities in the system are limited to sysop access. These include:

Menu	Options
Groups Menu	All
User Menu	All
Workspace Users Menu	All
Workspaces Menu	Add/Edit/Delete/Users options
Code Sub-types Menu	Add/Edit/Delete options
Sub-Type Parts Menu	Edit/Clear options
Products Menu	Add/Edit/Delete options
Product Modules Menu	Add/Edit/Delete options
Ws-Config Menu	Populate Workspace/Schema Xref Build/Build Names Table

### 3.10.12. Repository Dump and Load

Do not use the PROGRESS database administration utilities to dump and load a Roundtable repository database. This is because the Roundtable repository database stores many RECID values. If you use the PROGRESS database administration utilities to dump and load your Roundtable repository database, the RECID values will be lost, and your new Roundtable repository database will be unusable.

Roundtable comes with its own tools for dumping and loading the repository database. These tools preserve the RECID values.

Follow these steps to dump and reload the Roundtable repository database:

1. Ensure that adequate disk space is available to accommodate a database dump. It is difficult to predict how much space this should be, but dumps often require between half to three quarters the size of the original database.
2. From the PROGRESS editor of a Roundtable session, run `rtb_dmp1.p` to dump the Roundtable repository data to ascii ".d" files in the current directory. This program is located in the `rtb_util` subdirectory of the Roundtable installation and makes calls to a second program in that directory, `rtb_dmp2.p`, expecting the same relative path (that is, RUN `rtb_util/rtb_dmp2.p`).
3. Use the PROGRESS data dictionary tools to dump the user table of contents.
4. Use the PROGRESS data dictionary tools to dump the Roundtable repository database schema to an ascii ".df" file in the current directory.
5. If the new Roundtable repository database is to be in a different location, change to that new location and move in or copy in the ascii ".d" and ".df" files created earlier.
6. Use the PROGRESS `proddb` command to create a new Roundtable repository database from the PROGRESS empty database.
7. From the PROGRESS editor of a PROGRESS session on the new Roundtable repository database, use the PROGRESS data dictionary tools to load the schema from the ascii ".df" file in the current directory.
8. From the PROGRESS editor of a PROGRESS session on the new Roundtable repository database, use the PROGRESS data dictionary tools to load the user table of contents.
9. From the PROGRESS editor of a PROGRESS session on the new Roundtable repository database, run `rtb_lod1.p` to load the Roundtable repository data from the ascii ".d" files in the current directory. Like `rtb_dmp1.p`, this program is located in the `rtb_util` subdirectory of the Roundtable installation and makes calls to a second program in that directory, `rtb_lod2.p`, expecting the same relative path (that is, RUN `rtb_util/rtb_dmp2.p`).

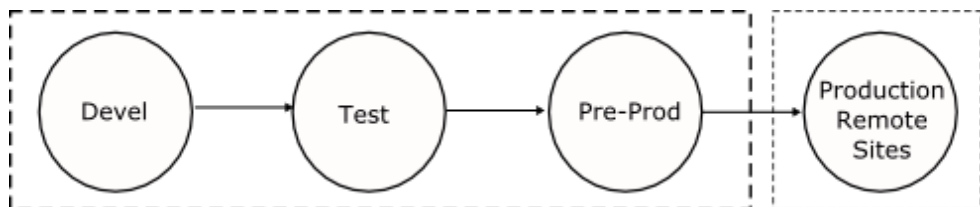
### 3.10.13. Defining Your Application

There are a number of steps required to set up Roundtable. The following list is intended to provide you with an outline of these steps. At each of these steps, you should review the manual for detailed instructions and descriptions.

1. Create Module Definitions. See Section 3.8.1, “Adding a Workspace Module Definition” [3–21].
2. Create a Product. See Section 3.7.2, “Adding a Product” [3–15].
3. Create Product Modules. See Section 3.7.6, “Adding a Product Module” [3–17].
4. Create Subtypes. See Section 3.9.6, “Adding a Subtype” [3–26].
5. Create a Workspace. See Section 4.2.3, “Adding a Workspace” [4–3].
6. Associate Products with the Workspace. See Section 4.3.2, “Adding a Workspace Source” [4–13] section.
7. Back up your Roundtable database (rtb.db). Do this so that you can easily change your mind about the configuration steps above, and then re-run the steps below.
8. Create a Task for loading schema. See Section 5.3.2.2, “Adding a Task” [5–6].
9. Create a PDBASE object. See Section 6.7.2, “Adding a PDBASE Object” [6–17].
10. Load the application database schema. See Section 6.7.11, “Load PROGRESS Schema” [6–22].
11. Complete the Task. See Section 5.3.2.6, “Completing a Task” [5–9].
12. Create a Task for loading application source files.
13. Register objects using Module Load. See Section 7.4, “Module Load” [7–7].
14. Complete the Task.
15. Create a Release. See Section 4.6.2, “Adding a Release” [4–26].

### 3.11. Loading Multiple Workspaces

If you have a chain of existing workspaces, you might want to capture the status of each of the workspaces in Roundtable. Refer to the following diagram, which describes a typical development environment:



Different versions of the application source and schema exist in each of these workspaces. You want to capture the different object versions that exist in each workspace. To do this, you define and load the contents of the Pre-Prod, Test, and Devel workspaces into Roundtable.

### 3.11.1. Pre-Prod

You start by loading the Pre-Prod workspace first because it contains the earliest version of the system. The steps below assume that you have already defined your subtypes, products, and product modules. See Section 3.10.13, “Defining Your Application” [3–41].

1. Create a workspace named Pre-Prod.
2. From Workspace Sources, add the Pre-Prod workspace as the primary workspace source to itself for each product and product module.
3. Create a new task for loading your Pre-Prod schema.
4. For each of the databases in your Pre-Prod workspace, create a new PDBASE object and run Load Schema.
5. Complete your task.
6. Create a new task for loading your workspace source.
7. Run Module Load for each of the modules in your Pre-Prod workspace.
8. Complete your task. If any of the programs in your task fail to compile, flip their compile flags to ‘No’.
9. Create a release in the Pre-Prod workspace. This is Pre-Prod release 1.

### 3.11.2. Test

Follow these steps to create a Test workspace in Roundtable and populate it:

1. Create a workspace named Test. Specify R-code = ‘No’ and S-Code = ‘No’.
2. Use Workspace Sources to specify Pre-Prod as a workspace source for Test and then include all products and product modules available in the Pre-Prod workspace.
3. Use the Import process to copy object definitions from Pre-Prod to Test. Note that you are only copying the definitions of the objects. You specified S-Code = ‘No’ for the workspace, so no source will be copied into the Test workspace directory. This is necessary so that you do not overwrite the source that is already in the Test workspace.
4. Edit the workspace definitions so that R-code = ‘Yes’ and S-code = ‘Yes’.
5. Create a task for loading new and modified schema definitions.

6. For each of the databases in your Test workspace, run the Load Schema process. This will find any new and modified schema, and bring your Roundtable schema object definitions in sync with your physical Test workspace schema.
7. Complete your task.
8. Create a task for loading new and modified source.
9. Run the Global Change Finder to find all source files in the Test workspace that are different than those that you loaded into the repository for the Pre-Prod workspace. The Global Change Finder works by comparing the source in the file system with the source registered in the Roundtable repository. The Global Change Finder allows you to create new object versions for each of the source files found to be different.
10. Run Module Load for each of the modules in the Test workspace. This will find any new programs in the Test workspace that do not exist in the Pre-Prod workspace.
11. Complete your task. If any of the programs in your task fail to compile, flip their compile flags to 'No'.
12. Create a release in the Test workspace. This is Test release 1.

### **3.11.3. Devel**

Follow the same steps as outlined in the Test workspace, except that you will be copying object definitions from the Test workspace into the Devel workspace.

### **3.11.4. Setting Up the Workspace Flow**

Now that you have the Devel, Test, and Pre-Prod workspaces registered into Roundtable's repository, you can specify the normal flow of system changes. This is done from Workspace Sources.

1. Add the Test workspace as a source to the Pre-Prod workspace. Include all products and product modules.
2. Delete any records specifying Pre-Prod as a primary source.
3. Add the Devel workspace as a source to the Test workspace. Include all products and product modules. It is not necessary to remove Pre-Prod as a source workspace to the Test workspace. It might be useful to have it as a source for Test if you want to have a quick way of moving patches made in Pre-Prod back into Test.
4. For each product and product module, add the Devel workspace as the primary source to itself. It is not necessary to remove Test as a source workspace to the Devel workspace. It might be useful to have it as a source for Devel if you want to have a quick way of moving revisions made in Test back into Devel.



---

# Workspaces

## 4.1. Introduction

Roundtable provides a secure development and testing environment by managing configurations of your software applications in workspaces. A workspace is a copy of your application program and related databases, the content of which is known and managed by Roundtable. Roundtable manages the creation, deletion, and modification of objects (code, databases, tables, fields, and text) within the workspace.

This introduction to workspaces presents detailed information about utilizing workspaces in your development environment. Specifically, this chapter discusses:

- Workspace Maintenance
- Workspace Sources
- Object Variants
- Database Schema Updates
- Releases
- Imports
- Workspace Populate Process
- Build Names Table
- Schema Xref Build
- Deployments
- Procedure Updates and Compiles on Remote Sites
- Database Schema Updates on Remote Sites

## 4.2. Workspace Maintenance

As part of your software development process, Roundtable keeps copies of objects in isolated areas called workspaces. You can have multiple workspaces for a single development project, with each workspace representing a different stage of development. Changes made within a workspace do not affect other workspaces. Typically, you create workspaces that parallel the different development stages of your product, such as development, testing, and deployment.

When creating the first workspace for a project, you develop the objects and databases from scratch. After that, you can create workspaces by importing objects from other workspaces. Roundtable compares the objects in the target and source workspaces, and then imports the objects with differences into the target workspace.

To help you understand how to use workspaces, the following scenario describes four common workspaces for one base product. Each description includes how to create the workspace, what the workspace includes, and what type of quality assurance testing is best for that workspace. Remember, this is only one scenario; tailor workspaces to your products and development cycle, as described in the Introduction section in Software Configuration Management.

- **Development workspace:** Use the development workspace to create, modify, and delete new objects and data. Since this workspace is volatile, it is only suitable for limited, informal testing, usually done by programmers.
- **Test workspace:** Create the application in the Test workspace by importing objects and data from the Development workspace. Although this workspace changes less frequently than the Development workspace, some changes are necessary to complete the testing process. This workspace is stable enough for significant quality assurance Alpha testing.
- **Pre-production workspace:** After testing, objects and data are imported from the Test workspace into the Pre-production workspace. Since this workspace should function with few or no major problems, you can release it to sophisticated users for Beta testing and evaluation. Typically, you should require approval to make changes in this workspace.
- **Production workspace:** Import objects and data from the Pre-production workspace into the Production workspace. This workspace is a "live" system that must function flawlessly under real-world conditions. You should typically require approval to make changes in this workspace.

#### 4.2.1. Guidelines for Workspace Planning

Roundtable can manage many workspaces. For example, quality assurance, testing, and development should be done in different workspaces. Multiple workspaces can aid in the quality assurance strategy, but more is not necessarily better. Consider the timing impact of each additional workspace when planning your overall development and quality assurance strategy. Use the following guidelines:

- Organize your work into logical categories, such as research and development, product development, custom development, or maintenance.
- Use a consistent naming convention. For example, always use the abbreviation DEV when referring to development workspaces, such as BASEDEV, CUST1DEV, CUST2DEV, etc.

- Do not create more workspaces than you have resources to manage. In most circumstances, separate Test and Deployment workspaces, along with the necessary development workspaces, are sufficient to complete your product.

### 4.2.2. Workspace Window Description

Roundtable workspaces are defined in the Workspaces window:

The following table provides the Workspace Window field descriptions:

Field	Description
Workspace ID	This field contains a unique code that identifies the workspace.
Description	This field contains a description of the workspace.

### 4.2.3. Adding a Workspace

Follow these steps to create a new workspace:

1. If Roundtable security is enabled, log in as sysop.
2. Choose Workspaces from the Main Menu. The Workspaces Menu appears.
3. Choose Add. A row is added for the new workspace.
4. Enter the Workspace ID.

Make sure the New Workspace ID describes the product and its stage of development. Try to name your workspaces consistently. For example, if you have one base product and two custom products, name your Development workspaces BASEDEV, CUST1DEV, and CUST2DEV.

The workspace ID can be up to sixteen characters long. If your site number is not 0, then you must enter the three-digit number as a prefix for the workspace name. If your site number is 0, you cannot begin your code with a numeric character. By default, your site number is zero. Only partner sites are normally assigned a site number.

5. Enter the Description and Xref Level. The Details and PROPATH for Workspace appears.
6. Enter the R-code, S-code, and PROPATH components.

The workspace path tells Roundtable where to store the workspace and its components. The first workspace path should always be the root directory to your workspace. No two workspaces should share the same root directory. Roundtable uses the other specified paths to locate your source code. Do not enter a drive letter or

colon in any workspace path.

The workspace path must begin with a slash (/).

Progress and Tugboat recommend that you do not use the \$SRCPATH and \$RUNPATH tokens in your workspace path. They expand your ProPath to contain every module source path and every module R-code path, which can cause a number of issues including:

Manually building the same large ProPath at your deployment site. If code is referenced without a relative path (e.g. RUN object.p instead of RUN ap/object.p), it will test correctly under Roundtable since ap would be in your ProPath. However, the code will fail at the deployment site without a ProPath modification.

Performance may be affected since Progress will have to search a very large ProPath to find files.

If many modules are defined, they may exceed Progress limits.

If you want to add module directories to your ProPath without using tokens in your workspace path, manually list the explicit directories in the workspace path definition. These entries should be explicit entries containing the root workspace path. Do not enter relative directories or directories not off of the root workspace path. The following is an example of a valid workspace path:

```
C:/workspaces/devel  
C:/workspaces/devel/includes  
C:/workspaces/devel/code
```

7. The Workspace Compile Parameters dialog box appears. Enter any additional COMPILE syntax parameters to be used when compiling objects in this workspace. This field can normally be left blank. It is most often used for compiling workspaces with support for multiple languages.
8. Press the **GO** key.

#### 4.2.4. Editing a Workspace

You can change the description, Xref Level, R-code, or S-code compile parameters for a workspace at any time; however, you cannot change the workspace ID.



Changing the Xref Level, R-code, or S-code after you have assigned and compiled an object has no immediate effect on objects in the workspace. You must recompile for the Xref Level and R-code to take effect.

Follow these steps to edit a workspace:

1. If Roundtable security is enabled, log in as sysop.
2. Select the Workspace to edit from the Workspace ID selection list.
3. From the Workspaces Menu, choose Edit.
4. Fill in the new Description, Xref Level, R-code, and S-code fields for the workspace and modify the Workspace Paths as necessary. Note that changing the workspace root directory in this screen will not cause the contents of the physical directories to move. You are responsible for moving or renaming directory structures to match the specifications present in this window.
5. Press the **GO** key. The Workspace Compile Parameters dialog box appears. Enter any additional **COMPILE** syntax parameters to be used when compiling objects in this workspace. This field can normally be left blank. It is most often used for compiling workspaces with support for multiple languages.
6. Press the **GO** key.

### 4.2.5. Deleting a Workspace

You can delete a workspace if you have sufficient security access and no one else is currently working in the workspace. Note that deleting a workspace does not delete any completed objects from the repository. A workspace is simply a list of object versions that defines a configuration of your software application and a related set of application components in the directories specified in the workspace paths. However, once you delete the workspace, the configuration is lost and cannot be reclaimed. Although tasks done under the workspace are not deleted, they may be far less available for casual review if the workspace to which they belong is deleted. In addition, all release, deployment, and general histories of the workspace are deleted.



Deleting a workspace removes all source code at the operating system level. Work-in-process code might be deleted and will not be recoverable.

Follow these steps to delete a workspace.

If Roundtable security is enabled, log in as sysop.

1. From the Workspaces Menu, select the workspace that you want to delete.
2. Choose Delete. The message, "Delete this record" appears.
3. Enter **Yes** to complete your workspace delete.

#### 4.2.6. Viewing Workspace Event History

Use the Event History dialog box to view all events that have occurred in a workspace. You are able to filter the display based on a date range and/or user ID.

1. From the Workspaces Menu, select the workspace in which you want to view the event history.
2. Choose Ws-config. The Workspace Config. Menu appears.
3. Choose Event History Menu. The Set Event History Filters dialog appears.
4. Enter your selections, and then choose OK. (To create an unfiltered display, toggle Include all dates on and specify **All** user IDs). The Event History screen appears.

The following table provides the Event History screen field descriptions:

Field	Description
Event#	Event numbers for object that are checked out are always 99,999,999. When the object is checked in the final event number is assigned.
Date	The date on which the final event number is assigned. This is the date that the action was taken on the completed (checked in) object.
Actions	The action code describes what action was taken on the object in the workspace. Action codes include: Add – New object version was added. Delete – Object assignment was deleted. Assign – Object assignment was created.
Type	The type of object: PCODE, DOC, PDBASE, PFILE, PFIELD.
Object	The name of the object affected by the action.
User ID	The user that performed the action.
Version	The version of the object affected by the action.
Pmodule	The product module the affected object belongs to.
Module	The workspace module the affected object appears in.
Object Group	The object group code assigned to the object assignment at the time of the action.
Level	The object level code assigned to the object assignment at the time of

Field	Description
	the action.
Task#	The task number associated with the object version affected by the action.
Subtype	The subtype of the object affected by the action.

#### 4.2.7. Virtual "Read-Only" Workspace

By turning off the S-code and R-code flags for a workspace, you have the ability to keep a full "readonly" Workspace on file without taking up disk space. When you view a current or past version of an object, Roundtable extracts the object to a temporary file and opens it as read-only.

#### 4.2.8. Hidden Workspace

Marking a workspace as hidden removes it from the workspace list on the Roundtable Workspace Selection screen. To hide a workspace, choose Hide Workspace on the WS-Config menu. To restore the workspace, choose Show Workspace on the WS-Config menu.

#### 4.2.9. Archived Workspace

Marking a workspace as archived removes it from the workspace list (hidden workspace) and removes all source from the workspace. Since the workspace is "archived" and not deleted, all Xref information is retained in the repository. To archive a workspace, choose Archive Workspace on the WS-Config menu. To restore the workspace, choose Unarchive Workspace on the WS-Config menu.

#### 4.2.10. Workspace Reports

Roundtable offers a number of reports about your workspaces.

The following table describes the workspace reports:

Report	Definition	Benefit
Workspace Report	A complete list of the objects in a workspace ordered by workspace module.	Provides a hard copy of the current workspace configuration.
Unapplied Changes	A list of all schema changes that have been entered but not updated.	Verifies the status of the schema in the workspace.

Report	Definition	Benefit
Event History Report	The history of a workspace between two events, sorted from newest to oldest event.	Traces the history of individual objects or groups of objects belonging to a product module.
Release Report	A detailed list of the changes in a workspace between two releases.	Useful for testers and/or customers when receiving new releases. Use the Object Details and Show Update History fields for this purpose.
Changes Report	A list of the changes in a workspace across a range of events.	Provides testers and/or customers with extensive information about what has changed in a workspace between any two events.

#### 4.2.11. Printing Reports

Follow these steps to print a workspace report:

1. From the workspace selection, select the Workspace on which to report.
2. From the Workspaces Menu, choose Reports. The Workspace Reports Menu appears.
3. From the Reports menu, select the report you want to print. Roundtable displays a report options dialog box for many of the reports. These options are explained in the following sections.
4. When the report dialog box appears, enter the report options you require. The Print Destination dialog box appears.

#### 4.2.12. Print Destination Dialog Box

You use the print destination dialog box to tell Roundtable where to send reports and other printed output.

Follow these steps to select a destination:

1. Enter the new print destination. If you enter "N" for no change, you are not prompted for print destination parameters. Otherwise, enter "P" for printer, "F" for file, or "T" for terminal.
2. Enter the print destination parameters:



If you select "P", you are prompted for Options and Page Length. The option command must contain the printer command and the token "\$file". For example: "lp -s \$file". The Page Length is the number of lines to print per page.

If you select "F", set the Options field to the name of the file to set the output destination. Enter the length of each page, by lines per page. Enter **Yes** or **No** in the Append field to set whether or not you want to append the new report to the end of an existing file.

If you select "T", you are prompted for Options and Page Length. In the Options field, Enter the terminal mode to use for output display.

Refer to the RTBSETUP File Options section for more information.

### 4.2.13. Workspace Report

From the Workspaces Menu, choose Reports. Select the Workspace Report. The Workspace Report dialog box appears.

The following table provides the Workspace Report field descriptions:

Field	Description
Show Workspace Objects?	Type <b>Yes</b> to print the objects in the workspace or No to print configuration information only.
Object Details?	Type <b>Yes</b> to print the object description.
Show Update History?	Type <b>Yes</b> to print the notes the programmer entered when changing the objects.

### 4.2.14. Workspace Event History Report

From the Workspaces Menu, choose Reports. Select the Event History Report. The Workspace Event History Report dialog box appears.

The following table provides the Workspace Event History Report field descriptions:

Field	Description
User ID	Select a user from the dropdown list. Select "All" for all users.
Date Range	Supply "From" and "To" date range for the report.
Event Range	Fill in the beginning and ending event numbers to include in the report.
Pmodule	Fill in the product module number. Use an asterisk (*) to denote any product module number.
Object Types	Select the type of object (PCODE, PFILE, PDBASE, etc.) from the

Field	Description
	dropdown list. Select the asterisk (*) to denote any object type.
Object	Fill in the name of the object (menu.p, cust.p, etc.). Use an asterisk (*) to denote any object.

### 4.2.15. Release Report

From the Workspaces Menu, choose Reports. Select the Release Report. The Release Report dialog box appears.

The following table provides the Release Report field descriptions:

Field	Description
To Release	Fill in the last release number to include in the report.
From Release	Fill in the first release number to include in the report.
Show Release Note?	Type <b>Yes</b> to print the note on the Release window.
Show Update History?	Type <b>Yes</b> to print the notes the programmer entered when changing the objects.
Headers?	Type <b>Yes</b> to print the titles of the report fields.
Show Only Latest?	Type <b>Yes</b> to print only the latest changes to an object made between the two event numbers.
PCODE Details?	Type <b>Yes</b> to print the details of a PCODE object.
SCHEMA Details?	Toggle on to print the details of a schema object.
Version Info?	Type <b>Yes</b> to print the version number, task number, etc.

### 4.2.16. Changes Report

The Workspaces Changes Report dialog box appears when you choose Report and Changes Report from the Workspaces Menu. This dialog box allows you to specify the information to include in the Workspace Changes Report.

The following table provides the Changes Report field descriptions:

Field	Description
From Event#	Fill in the first event number to include in the report.
To Event#	Fill in the last event number to include in the report.
Show Only Latest?	Type <b>Yes</b> to print only the latest changes to an object made between

Field	Description
	the two event numbers.
Object Details?	Type <b>Yes</b> to print the object description.
Show Update History?	Type <b>Yes</b> to print the programmer's notes.

### 4.3. Workspace Sources

After you create a workspace, you define its contents by assigning product modules to it. This assignment of product modules is the configuration hierarchy of the software system managed in the workspace.

After you assign product modules to the workspace configuration, you can create new objects in the workspace, each of which belongs to one of the product modules. When you are loading objects into the system for the first time, you can use the Module Load utility to create many new objects quickly. For a detailed description of this utility, see Section 7.4, “Module Load” [7–7].

You can also assign existing objects from the Roundtable repository to the new workspace. This assignment is done one object at a time or by using the Roundtable import process.

The import process examines a source workspace to determine what objects need to be assigned to or deleted from a target workspace to make the workspaces' configurations, where these configurations coincide, as similar as possible. The following simplified explanation of the import process shows why workspace sources are necessary and useful:

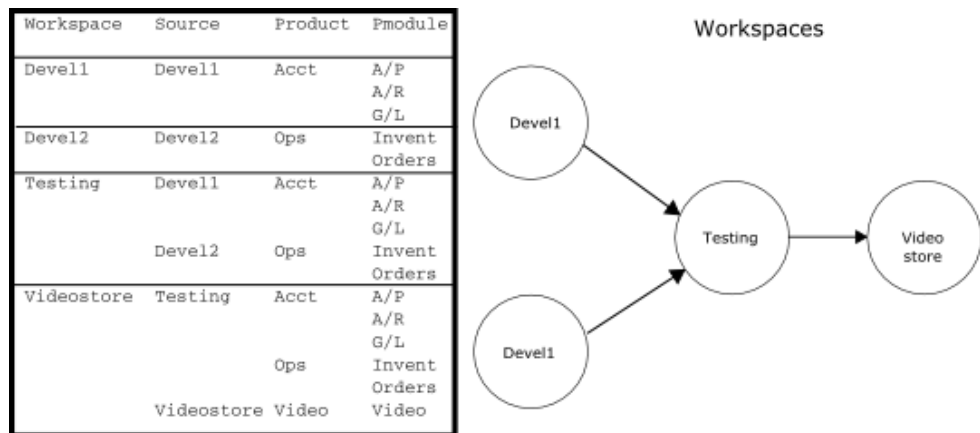
1. The import process compares the configuration hierarchy of each workspace. When the source workspace contains a product module that is also contained in the target workspace, it does an object-by-object comparison within that product module to see if objects in the target workspace have the same object versions as objects in the source workspace. If they are different, the import process places a reference to the object version found in the source workspace in the import control table with an action code of Update.
2. If a product module in the source workspace contains objects that do not exist in the corresponding product module in the target workspace, references to these new objects are placed in the import control table with an action code of Add.
3. If an object has been deleted from the source workspace, but still appears in the target workspace, a reference to the deleted object is placed in the import control table with an action code of Delete. With your permission, the import process can remove the object from the target workspace.
4. If you accept the suggestions supplied by the import process, it assigns the specified

object versions from the Roundtable repository to the target workspace. It is important to remember that these object versions are being extracted out of the repository, not simply copied from the source workspace to the target workspace. One reason for this is that only the latest completed objects in the source workspace are considered by the import process. If the import process copied from the source workspace to the target workspace, it might copy a work-in-process version of the object.

The import process is actually more sophisticated than the example implies. For more detail on the import process, see Section 4.7, “Imports” [4–28].

When assigning product modules to workspaces, it is necessary to specify a source workspace. When there is no source workspace, such as when you assign workspace modules to your first workspace, specify the name of the current workspace as the source workspace. Roundtable automatically marks each product module assigned in this way as PRIMARY to the current workspace. This is a way of saying that the current workspace is the headwater of the flow of objects for the specified product module.

Refer to the following example:



The first two columns contain the names of each workspace and its workspace sources. The Devel1 and Devel2 workspaces are primary workspaces because their workspaces are identical to their sources. The third and fourth columns list the products and the product modules (within those products) that are assigned to the workspace.

The Testing workspace has Devel1 and Devel2 designated as its workspace sources. Unless specifically excluded, Roundtable imports all objects belonging to the products and product modules assigned to Devel1 and Devel2 into the Testing workspace.

The Videostore defines Testing as its source workspace. Note that another product, Video, has been added to the Videostore workspace. Therefore, objects in this workspace are both

created and imported. The import process imports all objects in the Testing workspace unless otherwise specified or blocked by object variants belonging to the Video product. For more information on object variants, see Section 4.4, “Object Variants” [4–15].

### 4.3.1. Workspace Sources Screen Description

After you create a workspace, you define its contents by assigning product modules to it. Assign product modules using the Workspace Sources menu.

The following table provides the Workspace Sources Menu field descriptions:

Field or Menu Item	Description
Status (for Product)	Contains one of the following values: PRI — Source workspace is the current workspace. NEW — Newly assigned. User must change to INC or EXC. INC — Product will be included in imports. EXC — Product will be excluded from imports.
Source Workspace	Contains the name of a source workspace.
Product	Contains the Product-ID belonging to the source workspace.
Name	Contains the description of Product.
Add	Add a Workspace Source.
Delete	Delete a Workspace Source.
Toggle	Toggle the status: Included/Excluded. This menu item has no affect on PRI status.
Status (for Product Module)	Contains one of the following values: PRI — Source for Product Module is the current workspace. NEW — Newly assigned. User must change to INC or EXC. INC — Product Module will be included in imports. EXC — Product Module will be excluded from imports.
Pmodule	Contains the Product Module code.
Description	Contains the description of Product Module.
Select	Choose Select to go to the Product Modules Assignment to Workspace menu.

### 4.3.2. Adding a Workspace Source

Follow these steps to add a workspace source:

1. From the Workspaces Menu, select the workspace to which you want to add sources.
2. From the Workspaces Menu, choose Ws-config. The Workspace Config. Menu

appears.

3. From the Workspace Config. Menu, select the Workspace Sources Menu.
4. Choose Add. Roundtable inserts a row for the new workspace source record.
5. Enter the name of the workspace that is to be a source for the current workspace, and press the **GO** key. Roundtable creates a source workspace/product record for each product that is assigned to the workspace source that you selected. Each product module that belongs to the specified source is inserted into the Product Modules for Assignment to Workspace table with a status of NEW.
6. Choose Toggle to change the status of the workspace source to INC.
7. Choose Select to go to the Product Modules Assignment to Workspace Menu.
8. For each product module with a status of NEW, choose Toggle to change the Pmodule status of the product module to INC or EXC.
9. Choose **Quit** to leave the Workspace Sources Menu.

### 4.3.3. Deleting a Workspace Source

Follow these steps to delete a workspace source:

1. From the Workspaces Menu, select the workspace to which you want to add sources.
2. From the Workspaces Menu, choose WS Config.
3. From the Workspace Config Menu, select the Workspace Sources Menu.
4. Select the workspace source to delete.
5. Choose Delete to delete the workspace source.
6. The message, "Delete this record?" appears.
7. Type **Yes** and press **ENTER** to delete the workspace source.



Roundtable deletes all rows in the table for the source workspace and all entries in workspace. The Import process removes objects from the workspace that have no current workspace source.

8. Choose **Quit** to leave the Workspace Sources Menu.

### 4.3.4. Editing Workspace Module Parameters

R-code, S-code, and Xref level values can be set for each workspace module independently. These workspace module values override the workspace level

specifications for these values you have entered in the Workspaces window. See Section 4.2.2, “Workspace Window Description” [4–3] for a detailed description of the R-code, S-code and Xref Level fields.



Changing the Xref level, R-code, or S-code after you have assigned and compiled an object has no effect on the object. You must recompile for the new Xref and R-code to take effect.

Follow these steps to edit workspace module values:

1. From the Main Menu, select workspaces. The Workspaces Menu appears.
2. Scroll to the workspace you want to modify and choose Select. The Workspace Modules Menu appears.
3. Scroll to the workspace module you want to modify and choose Edit.
4. Enter the module’s Xref level. Set the Xref level to **0** to default to the workspace’s Xref level, or set it to another number to override the workspace’s Xref level. Enter **Yes** or **No** in the R-code field to specify whether R-code is stored in the module, or set this field to **?** to default to the workspace’s R-code value. Enter **Yes** or **No** in the S-code field to specify whether source code is stored in the module, or set this field to **?** to default to the workspace’s S-code value.
5. Enter the module lead in program, and press the **GO** key. The Module Compile Parameters dialog box appears.
6. Enter any compile syntax parameters to be used when compiling objects in this workspace module and press the **GO** key. This field can normally be left blank. It is most often used for compiling modules with support for multiple languages. If you enter compile parameters for a module, then the workspace compile parameters will be ignored. Leave this field blank to default to the workspace’s compile parameters.

## 4.4. Object Variants

Roundtable allows you to create more than one variation of the same object called variants. Variants have the same name and type but belong to different product modules. For example, you might need three variations of an install program, one each for UNIX, DOS, and Windows. Each of these is a variant. You use variants when you need support for:

- Custom system development
- Vertical markets
- Different hardware platforms

- See Section 5.4.2, “Creating an Object Variant” [5–12] for instructions on creating an object variant.

### 4.4.1. Object Repository

Roundtable keeps objects in the Roundtable repository. The repository stores all of the objects in the system and their definitions. The unique key for the object definition includes the following fields:

- Object type (PDBASE, PFILE, PFIELD, PCODE, or DOC)
- Product module (Pmodule) code (base\_ap, cust\_ap)
- Object name (menu.p, cust.p, etc.)
- Version code (01.00.00, 01.01.00, etc.)

The object definition allows an object of the same type, name, and version to exist in two different product modules, creating an object variant. Although Roundtable stores all object variants in the repository and objects can exist in more than one product module, only one variation of an object can be assigned to any given workspace. Each workspace represents a collection of object versions. Roundtable manages a table of these object versions called the configuration list. The following fields define the unique key for objects in this table:

- Workspace ID (Wspace-id)
- Object type (PDBASE, PFILE, PFIELD, PCODE, or DOC)
- Object name

### 4.4.2. Versions Versus Variants

The difference between versions and object variants can be confusing at first. Consider the following table:

Object Type	Product Module	Object Name	Version
PCODE	core_ap	menu.p	01.00.00
PCODE	core_ap	menu.p	01.01.00
PCODE	cust1_ap	menu.p	01.00.00

All of the objects in this table are stored together in the object repository. The first two show two versions of an object named menu.p. Remember that the unique key on objects stored in the repository includes the object type, product module, object name, and version code. The last object is an object variant that belongs to a product module named



cust1\_ap. You must always remember to consider the product module when looking at an object version.

### 4.4.3. Example of Object Variations

This example shows how you can use object variants. Assume you are creating an invoice printing program for many different types of businesses. Most of these businesses can use your core (or generic) program, but the Videostore application needs a special variation. To handle this, you need to create two variations of the same program.

You begin by creating the core program, invprnt.p, in the development workspace (refer to line 1, below). Then, import the core program into the Testing workspace (line 2). After testing, import the program into two workspaces, a core pre-production workspace (line 3) and a Videostore workspace (line 4). In the Videostore workspace, you can create a custom variant of invprnt.p using the video\_ap product module. You now have two variations of the same program (lines 3 and 4).

#	Object Type	Workspace	Product Module	Object Name	Version
1	PCODE	Devel	core_ap	Invprnt.p	01.00.00
2	PCODE	Testing	core_ap	Invprnt.p	01.00.00
3	PCODE	Preprod	core_ap	Invprnt.p	01.00.00
4	PCODE	Video	video_ap	Invprnt.p	01.00.00

### 4.5. Database Schema Updates

Relational database schemas are defined by PDBASE, PFILE, and PFIELD objects in the Roundtable repository—the schema objects assigned to a workspace are called the logical schema. The Progress databases managed by a workspace also contain schema definitions—these are called the physical schema. When you modify the logical schema objects, Roundtable does not immediately update the physical schema. This allows you to make extensive changes to the database schema definitions while others continue to use the databases managed by the workspace. You run the update schema process to post changes to the PROGRESS schema when it is convenient for all users of the workspace.

A complete on-line history is kept of every change made to the logical schema. This allows the update schema process to determine which changes must be made in the physical schema due to changes in the logical schema. Because the system tracks both the committed and edited state of the logical schema, it is not necessary to check in schema objects after each update schema process. Instead you can edit the logical schema and update these changes to the physical schema many times before completing the schema objects.

Roundtable provides data preservation capabilities for schema operations that would normally delete data from the database. For instance, Roundtable allows you to specify a change of data type for a field. The only way to post a change of this nature to the physical schema is to delete the fields and then add it back with a new data type. Using Roundtable, you specify a data transformation routine to convert the old data to the new data type. The data transformation will occur as part of the update schema process. See Section 4.5.10, “Data Procedures” [4–22].



Roundtable 10.1A Schema Update is NOT compatible with Schema Update lists built with earlier versions of Roundtable. Schema Update Lists built with earlier versions of Roundtable must be processed by those versions. Schema Update Lists built with Roundtable 10.1A cannot be processed with earlier versions of Roundtable.

Follow these steps to update the schema objects:

1. Build the Schema Update List. See Section 4.5.5, “Building the Schema Update List” [4–20].
2. Set the Deactivate Flags where necessary. See Section 4.5.7, “Deactivate Flags” [4–?].
3. Create and Edit Data Procedures. See Section 4.5.10, “Data Procedures” [4–22].
4. Update Physical Schema. See Section 4.5.12, “Updating Physical Schema” [4–24].
5. Delete the Schema Update List. See Section 4.5.13, “Deleting the Schema Update List” [4–24].

### 4.5.1. Database Integrity Check Report

If changes are made to a physical database schema outside of Roundtable, difficulties may arise. Check that the physical database schema matches the definition managed by Roundtable by performing the Database Integrity Check Report. You do not need to run this report often. However, if you have any reason to believe that someone might have made changes to a database managed in a Roundtable workspace directly, then you should run this report. The report lists all the differences between what Roundtable believes the physical schema should be and what it actually is.

Follow these steps to run the Database Integrity Check Report:

1. From the Workspace Objects Menu, find the PDBASE object that you want to report on.
2. Choose Select. The Progress Database Object Menu appears.

3. Choose Reports. The More Menu appears.
4. Choose the Integrity Check Report.

### 4.5.2. What to Do when the Physical Schema Is Out of Synchronization

If you find that the physical schema of a database is no longer synchronized with Roundtable, you should resolve the differences as quickly as possible. This can be done in one of two ways.

If the changes in the physical schema are acceptable, then run Load Schema. See Section 7.6, “Load Schema” [7–12]. This process will load the physical schema differences into Roundtable, and bring Roundtable’s schema object definitions in sync with your physical workspace database schema.

If the changes in the physical schema are not correct or will conflict with changes already made in the logical schema but not yet updated to the physical schema, then edit the physical schema using the Progress Data Dictionary to remove the differences. Ensure that you have removed all of the differences by running the Integrity Check Report again.

### 4.5.3. Schema Update Window Description

The Schema Update window is used to manage the schema update process.

The following table describes the columns in the Schema Update List:

Column	Description
File or Sequence	The file name or sequence name in the Progress dictionary.
Stage	BldErr: Error while building Schema Update List Ready: Ready to be updated Intermediate stages (Not normally seen by user); DelIdx: Delete index ModFld: Modify field NewIdx: New index ModFil: Modify table FixDat: Fix data Compl: Update is complete
Action	Add Update Delete
Flags	D: Deleted fields I: Unique index T: Datatype changes E: Extent changes
Data	No-action Purge only Dump only Load only Transform
Deact?	Deactivate indexes? Yes or No.

#### 4.5.4. Unapplied Changes Report

The Unapplied Changes Report lists each schema object in the workspace that has changed since the last Update Schema process. It is a good idea to run this report before updating the schema as a documentation tool and confirm that the changes about to be made to the Progress database are correct.

Follow these steps to print the report:

1. From the Workspaces Menu, choose Ws-Config. The Workspace Config. Menu appears.
2. Choose Update Schemas. The Schema Update Control Menu appears.
3. Choose Report.

#### 4.5.5. Building the Schema Update List

The first step in updating the physical schema is to build the schema update list. The Build function locates all logical schema objects that have changed since the last time the schema was updated and creates the schema update list.

Because a PFIELD can be assigned to several PFILE objects, a change to a single PFIELD object might affect the update of multiple tables.

After Roundtable generates the list of database tables and sequences that must be updated, it checks schema integrity. The integrity check looks for missing information such as a missing dump-name or primary index. It also checks for missing objects and other problems that would cause the table definition update to fail.



To see which schema objects need to be updated before building the Schema Update List, print the Unapplied Changes Report. For more information on this report, see Section 4.5.4, “Unapplied Changes Report” [4–20] section.



Sometimes tables are shown in the schema update list that do not actually need updating. These extra tables are sometimes included because Roundtable cannot resolve all cross references to changes in the PFILE and PFIELD objects related to the table. Including these tables will not cause any problems in the schema update process.

Follow these steps to build the schema update list:

1. From the Workspaces Menu, choose Ws-Config. The Workspace Config. Menu

appears.

2. Choose Update Schemas. The Schema Update Control Menu appears.
3. Choose Build. Roundtable prompts you for the working directory.
4. Accept the default directory schupd unless large amounts of data are being preserved and it is necessary to specify a different schema update directory to store temporary data. Enter a different directory path if necessary.
5. Press **ENTER** to generate the schema update list.

### 4.5.6. Database Storage Areas

You can choose to assign new tables or indexes to existing storage areas in the physical database by selecting the DB Areas menu option. This menu option is only active when a new table or index has been created.

Use Esc-R to switch the object browse between "edit mode" and "row mode." In edit mode you can individually assign objects to the DB Area specified in the drop down box. However in row mode you may select, using the space bar, a group of objects to assign to a DB Area.

### 4.5.7. Deactivate Flags

You can choose to deactivate the indexes on a table during the update schema process. This is necessary when making changes to the table that will result in non-unique index entries. The Deact? column shows the current status of the deactivate flag for a table.

Follow these steps to change the deactivate flag:

1. Select the table or sequence in the Schema Update list.
2. Choose Edit. The Update Processing Menu appears.
3. Choose **Quit**. The pointer is placed in the Deact? field.
4. If Deact? flag is Yes, type **No**.

**OR**

If the Deact? flag is No, type **Yes**.



You must manually re-index those tables for which the indexes have been deactivated after the schema update process. Use Progress utilities to re-index the database tables affected.

### 4.5.8. Applying a Table as "New"

When a table appears with an "Update" action in the Schema Update List, you can change the action to "New", causing the table to be dropped from the physical database and re-added. This can be useful if a previous "New" table update was skipped, causing the physical workspace database to be out of synchronization with the logical workspace schema.

Follow these steps to set the action of a table in the Schema Update List to "New":

1. Select the appropriate table in the Schema Update List.
2. Choose the "New" menu option. A warning about potential data loss appears:
3. If you wish to proceed, then choose the **Yes** button to proceed with change.

When you apply the update, the table will be dropped from the database and re-added. If you need to preserve data in the workspace database, define the appropriate data procedures before updating the schema. Also, with the table's action set to "New", you will be able to assign storage areas for the table's indices and LOB fields.

### 4.5.9. Skipping a List Item

You can skip the update of one or all of the items on your schema update list. This is useful if the database schema has already been updated in some other manner. When skipping a table, the status of the list item becomes Compl. Once skipped, the changes are assumed to have been made in the physical schema by some other means than the schema update process. Skipped tables and sequences will not show up on a subsequent build.

After skipping the update of one or more tables, choose Update Schema for final processing to occur.

Follow these steps to skip the update of a table in the schema update list:

1. Select the table to skip in the schema update list.
2. Choose Skip. The Skip File Update Option dialog box appears.
3. Select N, C, or A for none, current, or all files, respectively.
4. Press the **GO** key. The stage of a skipped file is changed to Compl.

### 4.5.10. Data Procedures

The update schema process can perform three functions on each file it updates:

1. Run Before Schema Update Data Procedure

2. Update Physical Schema
3. Run After Schema Update Data Procedure

The first and third processes are optional and performed only if specified for the table. These procedures are used to preserve data across data type or extent changes, to a field definition, or when unique indexes are changed.

For example, if you are modifying the extent of a field, the Before Schema Update Data Procedure is used to dump the data from the field and the record ID of each record in the table being updated. This is necessary because Progress forces the deletion of the field when an extent change is performed during the Update Physical Schema step. The After Schema Update Data Procedure is used to read the dump file and import the dumped data back into the table.

Roundtable generates templates for the Before and After Schema Update Data Procedures for each table based on the following options:

- **No-action:** No procedures are generated
- **Purge only:** A purge procedure is generated
- **Dump only:** Only a before update process procedure is generated
- **Load only:** Only an after update process procedure is generated
- **Transform:** Both before and after update processes are generated

Editing the generated procedures must be done to include logic for data transformations. For example, if the data type of a field is changing from Character to Integer, the After Schema Update procedure must provide an algorithm that converts the character values (imported from the dump file) to integer values to assign to the field.

When modifying a unique index, you can use the Before and After Schema Update Data Procedures to ensure that unique records exist in the database after the update. The Before Schema Update Data Procedure can save the record ID of each record in the table. During the schema update the index can be deactivated. Then the After Schema Update Data Procedure can find the records that the Before Schema Update Data Procedure saved and delete or modify those records to meet the requirements of the modified index. You need to manually re-activate the index using the Progress utility after the schema update is complete.

## 4.5.11. Creating Data Procedures

Follow these steps to create and edit data procedures:

1. Select the table in the schema update list.
2. Choose Edit. The Update Processing menu appears.

3. Choose the Processing option appropriate for the table.
4. Choose Before Process (edit) to edit the procedure generated to handle the Purge Only, Dump Only, or Transform option.

**OR**

Choose After Process (edit) to edit the procedure generated to handle the Load Only or Transform options.

5. Choose **Quit** to close the menu.

## 4.5.12. Updating Physical Schema

You must build the schema update list before you can update the physical schema. See Section 4.6.2, “Adding a Release” [4–26].

Follow these steps to update the physical schemas of the application databases managed in the workspace:

1. Choose Update. A status dialog box appears while Roundtable updates the schema displaying the files being updated.
2. The schema update list appears. Each updated file has a status of Compl.

## 4.5.13. Deleting the Schema Update List

After updating the schema, delete the schema update list. Roundtable will not allow using any of the schema objects as long as the list exists.

Follow these steps to delete the schema update list:

1. Choose Delete. The message "Delete this record?" appears.
2. Type **Yes**. Another message asks you to confirm deletion of the update process programs (dump and load programs).
3. Type **Yes**.
4. Choose **Quit** to close the Schema Update Control Menu.

## 4.6. Releases

A workspace is a collection of object versions that define a configuration of the application. Roundtable tracks each change as an event and assigns a sequential event number to it. These events are collectively known as the event history of the workspace.



The actions that create events include:

- The assignment of an existing object version to the workspace. This is commonly done by the workspace import process. The workspace import process quickly moves many object versions from a source workspace into a target workspace according to workspace update rules. Object versions can also be individually assigned to the workspace.
- The deletion of an object version from the workspace configuration. This is commonly done by the workspace import process, but can also be done individually.
- The check-in of an object. This results in a new object version in the workspace.

A release is a record of a specific event number in a workspace. It can be used to re-create the exact contents of the workspace configuration that existed as of that event.

Each release record has a sequential release number that uniquely identifies it in the workspace. Release records also contain a text label that describes the configuration of the system at the time the release record was created. For example, release number 23 from the Test workspace might have a descriptive label like "Beta Release 2.1a".

Release records are used by the workspace import process to control what can be pulled from a workspace. See Section 4.7, “Imports” [4–28].

Release records are used to control the content of deployments created by the system. See Section 4.11, “Deployments” [4–34].

### 4.6.1. Releases Window Description

The Release Window contains a list of each release that has been created in the current workspace. A release is a label identifying the state of the workspace as of the latest event count.

The following table provides the Releases Menu field descriptions:

Field or Menu Item	Description
Release	The release number is generated automatically.
Event#	The latest workspace event number when release was added.
Date	The date the release record was created.
User	The user who created the release.
Note	A user-specified description of the currently selected release.
Add	Choose this menu item to add a new release.
Delete	Choose this menu item to delete the latest release.

### 4.6.2. Adding a Release

Before adding a release, ensure that all schema objects in the workspace have a status of complete. Other types of objects can remain work-in-process.

Checked out objects have a work-in-process status and the special event number 99,999,999. When the object is checked in, a real event number for the new object version is assigned to the event record.

When you create a release record in a workspace that has work-in-process object versions, the configuration represented by the release includes the last completed object version of each work-in-process object. Work-in-process objects are never included in a configuration identified by a release record.

Follow these steps to add a release:

1. From the Workspaces Menu, select the workspace for which you want to add a release.
2. Choose Ws-config. The Workspace Config. Menu appears.
3. Select the Releases Menu.
4. Choose Add. A new release is inserted into the browse table.
5. Enter a note in the Note field. Use this note field to describe the current configuration of the application in the workspace (for example, 4.0a).
6. Press **GO**.
7. Choose **Quit** to close the Releases Menu.

### 4.6.3. Editing a Release

Once a release is created, only the Note field can be changed.

Follow these steps to edit a release:

1. From the Workspaces Menu, select the workspace for which you want to edit a release.
2. Select the Releases Menu.
3. Select the release and choose Edit.
4. Change the text in the Note field as necessary.
5. Press **GO**.
6. Choose **Quit** to close the Releases Menu.

#### 4.6.4. Deleting a Release

Roundtable only allows the deletion of a release record if it is not referenced by a completed deployment and if it is the last release created for the workspace.

Follow these steps to delete a release:

1. From the Workspaces Menu, select the workspace for which you want to delete a release.
2. Choose Ws-config. The Workspace Config. Menu appears.
3. Select the Releases Menu. The Releases Menu appears.
4. Select the Release to delete.
5. Choose Delete. The message "Delete this record?" appears.
6. Type **Yes** and press ENTER.
7. Choose **Quit** to close the Releases Menu.

#### 4.6.5. Finding a Release

If so many releases are created that you cannot find one easily by looking through the selection list, then search for a release by looking up the release number.

Follow these steps to find a release:

1. From the Workspace menu, select the workspace.
2. Choose Ws-config. The Workspace Config. Menu appears.
3. Select the Releases Menu. The Releases Menu appears.
4. Choose Find. The Releases in Workspace dialog box appears.
5. Enter the release number and press **ENTER**. The release is selected.
6. Choose **Quit** to close the Releases Menu.

#### 4.6.6. Release Report

The Release Report provides a detailed list of the changes that have occurred in a workspace between any two releases. Roundtable allows a comparison between specified releases.

Follow these steps to print the report:

1. From the Workspaces Menu, choose Ws-config.

2. Select Releases Menu. The Releases Menu appears.
3. Choose Report. The Release Report dialog box appears.
4. Enter the report options and press the **GO** key.

The following table describes the fields in the Release Report Submission dialog box:

Field	Description
From Release	Type the release number to report from, or type <b>0</b> to report on all releases.
To Release	Type the release number to report to, or type <b>0</b> to include the current workspace contents.
Page Headers	Type <b>Yes</b> to print a report header at the top of each page.
Object Headers	Type <b>Yes</b> to show an object type header at the beginning of each object type.
Release Information	Type <b>Yes</b> to show release information at the top of the report.
Task Summary Information	Type <b>Yes</b> to show summary task information at the top of the report.
Task Detailed Information	Type <b>Yes</b> to show a detailed task information at the top of the report.

Object Information fields:

Field	Description
Details	Type <b>Yes</b> to show detailed description of the latest object version.
Version Information	Type <b>Yes</b> to show the object version information (version and product module).
Show Only Latest Version	Type <b>Yes</b> to show only the latest version of each modified object.
Release Information	Type <b>Yes</b> to show the release number that the object version was first included in.
Task Information	Type <b>Yes</b> to show the task number that the object version was created in.
Show update history	Type <b>Yes</b> to show the update history notes for each object version displayed.

## 4.7. Imports

To synchronize the currently selected workspace configuration with the configurations of one or more source workspaces, use the import process. This process is the primary means of managing the flow of object versions among workspaces. See the Introduction section in Software Management Configuration for an overview of how these workspaces allow the management of various development cycles.

The import process reads the latest release configurations of one or more source workspaces and compares the object versions found in those source workspace configurations with the current configuration of the currently selected (target) workspace. Where the object versions in the source workspaces differ from those in the target workspace, the object versions are added to an import control list with an appropriate action code.

You then review the import control list and manually exclude or include items as necessary. Usually it is not necessary to do more than a quick review of this import control list. The new Sort feature allows users to sort objects listed by Status, Object, Product Module, Module, Object Group, or Task Number.

After reviewing the import control list, you launch the import process. The import process assigns objects from the repository to the target workspace to bring the configuration of the target workspace into agreement with the list of object versions included in the import control list. This import process might also remove objects from the workspace, if necessary.

Follow these steps to perform an import:

1. Build the import control table. See Section 4.7.1, “Building the Import Control Table” [4–29].
2. Edit the contents of the import control list, if necessary. See Section 4.7.5, “Toggling the Import Status” [4–31].
3. Perform the import processing to assign or delete objects in the target workspace. See Section 4.7.6, “Import” [4–31].
4. Delete the import control list. See Section 4.7.8, “Deleting the Import Control Table” [4–32].
5. Perform the update schema process if any schema object versions have changed. See Section 4.5, “Database Schema Updates” [4–17].
6. Perform a selective compile on the workspace. See Section 7.3.5, “Selective Compiles” [7–6].

### **4.7.1. Building the Import Control Table**

Set up the source workspaces prior to building the Import Control Table. See Section 4.3, “Workspace Sources” [4–11].

Once the Import Control Table is built, Roundtable locks the workspace and will not allow others to work in the workspace. If other users are working in the workspace when Roundtable tries to lock it, the build process stops. Once the import process is complete and the Import Control Table is deleted, Roundtable removes the lock. This workspace lock is not a Progress record lock and survives system shutdown and restart.

Follow these steps to build the contents of the import control table:

1. From the Workspaces Menu, select the workspace that you want to perform the import in.
2. Choose Ws-config. The Workspace Config. Menu appears.
3. Select Import Control Menu. The Import Control Table appears.
4. Choose Build. The Import Filter Selection dialog box appears.
5. Selecting the default "none" is recommended, especially if you are working with QA, pre-production, or production workspaces. If you select the Workspace/Release/Module or Task option or the Task option a warning will display. Select OK to continue.
6. If the Import Filter Selection was selected in Step 4, one of the following dialogs will appear.
7. Note that you must select at least one item from each list.
8. Choose **Yes** to continue.
9. Choose the **Yes** button to ensure that only objects with a version number higher than the version number of the object currently in the target workspace are given a status of INC (Include).
10. The list will now be built. You can sort the import list by Status, Object, Product Module, Module, Object Group and Task Number.

### 4.7.2. Import Analysis Report

The Import Analysis Report lists all of the import objects found during the build process along with information on each object currently in the workspace.

From the Import Control Table Menu select Reports → Import Analysis .

### 4.7.3. Version Notes Report

The Version Notes Report lists the version notes for all versions between the current version and the import version for the selected object.

From the Import Control Table Menu select Reports → Import Analysis

#### 4.7.4. Compare Source Report

The Compare Source Report displays the differences in the source code between the current version and the import version for the selected object.

From the Import Control Table Menu select Reports → CompareSource .

#### 4.7.5. Toggling the Import Status

The status of the objects in the Import Control Table can be either INC (Include) or EXC (Exclude). Only object versions with the INC status are processed during the import process. If you wish to INC all of the objects select "include-All" from the menu bar.

Use the **Toggle** button to change the status of any object.

Follow these steps to toggle an import status:

1. Select the object in the Import Control Table.
2. Choose the Toggle menu item. The status changes to either EXC or INC, depending on the previous status.
3. Repeat Steps 1 and 2 for each object you want to change.

#### 4.7.6. Import

The import process assigns or deletes objects in the current workspace that appear in the import control list with a status of INC (Include). The assign process first removes any source code belonging to an existing object version and then exports the source from the repository for the new object version. As each object is successfully assigned, its Done field value is changed to **Yes**.

If the import process is interrupted, it can be restarted. If uncertain of a PCODE object due to an interruption, you can manually assign it to the workspace. This forces the system to update the source code in the workspace from the Roundtable repository. See the "Assign an Object" section in Chapter 5 for more information.

After importing an object, Roundtable sets the object's update status in the workspace to New. This ensures the object is processed in the next update schema or selective compile process.

Follow this step to import the contents of the import control table:

- Choose the Import menu item. Roundtable assigns objects to the workspace.

### 4.7.7. PCODE Data Imports

If you are using Roundtable's data management features, you might want to run the PCODE data imports program for all of the newly imported PCODE objects. See Chapter 3, *Roundtable Administration* [3–1] for more details about the subtype edit program and its import mode.

Before running the PCODE Data Imports program, follow these steps:

1. Perform the import.
2. Update the workspace schema. This should be done before running the PCODE data imports program, because the schema for the data being managed might have changed.

Follow these steps to run the PCODE Data Imports program:

1. From the Workspaces Menu or the Workspace Module Menu, choose Ws-config. The Workspace Config Menu appears.
2. Choose All PCODE Data Imports. The Update Special Processing dialog box appears.
3. Choose **Yes**. A status window appears listing the objects that have been processed.

### 4.7.8. Deleting the Import Control Table

When Roundtable built the import control table, it placed a lock on the workspace to prevent others from working in the workspace during the import process. To remove the lock, you delete the import control table.

Follow these steps to delete the import control table:

1. Choose Delete. The message "Delete contents of Import Table?" appears.
2. Type **Yes** and press **ENTER**.
3. Choose **Quit** to close the Import Control Table Menu.

## 4.8. Workspace Populate Process

Use the workspace populate process to restore or remove source from a workspace directory. A workspace is a configuration of object versions defined by the configuration list stored in the repository. This makes it possible to restore the source of a workspace by extracting the appropriate object version's files from the repository.



There are two common situations where it is necessary to use the populate process. The first is disaster recovery. For example, if a workspace directory was deleted, the populate process can be used to restore its contents. Only the latest completed version of each object is restored. If work-in-process objects existed and had a status of Central, this work would be lost. In fact, the populate process does not even try to restore work-in-process objects to ensure that they are not overwritten in the populate process.

The second situation involves the use of the workspace and workspace module S-code flag. This flag indicates whether source (and other object files as well) should be stored in the workspace. Set this flag to No, then run the populate process to remove source from the workspace. Set this flag to **Yes**, then run the populate process to restore source in the workspace.

A short cut can remove the source from a workspace if no work-in-process objects exist. Change the S-code flag to No, then delete the source manually. This is a quick way to free disk space when a workspace has not been used for some time and you want to keep the configuration of the workspace indefinitely.

Follow these steps to run the workspace populate process:

1. If Roundtable security is enabled, log in a sysop.
2. From the Workspaces Menu, select the workspace that you want to populate.
3. Choose Ws-config. The Workspace Config. Menu appears.
4. Choose Populate Workspace. The Populate Workspace Utility dialog box appears.
5. Type the name of the module to populate or use MATCHES criteria to populate many modules.
6. Press the **GO** key. Roundtable displays the name of each PCODE and DOC object while it writes the file out into the workspace directory.

## 4.9. Build Names Table

This option rebuilds the physical names table for the workspace in the repository. This utility is rarely needed. Run this utility only when you believe that Roundtable has become confused about the physical names of objects in a workspace. The Build Names Table most commonly needs to be run after the source code directory for a module. When you run this utility, ensure that no one is using the workspace in which the utility is run.

Follow these steps to run the build names table utility:

1. If Roundtable security is enabled, log in as sysop.
2. From the Workspaces Menu, choose Ws-config. The Workspace Config. Menu appears.

3. Select the Build Names Table.
4. Select the workspace to Build Names Table, then choose OK to execute.

## 4.10. Schema Xref Build

This option rebuilds the cross-reference information among the schema objects in the repository for the currently selected workspace. This option is rarely required but is occasionally necessary if Roundtable gets confused about which schema objects reference each other. If you run this utility, ensure that no one is using the workspace while the utility runs.

Follow these steps to run the schema xref build utility:

1. If Roundtable security is enabled, log in as sysop.
2. From the Workspaces Menu, select the workspace for which you want to rebuild the schema Xref table.
3. Choose Ws-config. The Workspace Config. Menu appears.
4. Select Schema Xref Build. Roundtable displays a message while it builds the schema Xref table. This process can take several minutes or longer, depending on how complex your workspace schema is.

## 4.11. Deployments

Deployments provide a mechanism for packaging software changes for delivery to remote sites. It is not necessary for these remote sites to be running Roundtable to receive updates, but some Roundtable utility procedures must be delivered with your update. Sites that have the Roundtable system can be updated with information directly from the repository, if desired. Sites that receive updates packaged by Roundtable are called remote sites. Sites having the Roundtable system and receiving repository information are called partner sites. Each site is designated to be of one of the following types:

- R: Runtime License
- Q: Runtime Query License
- D: Development License
- S: Source License
- P: Partner License

Roundtable automatically encrypts source files based on the type of site and the content of the Encrypt field in each object record. Roundtable automatically includes repository information in export format for partner sites.

To create an encrypted-source deployment, xcode must be in your UNIX PATH. The xcode program comes with Progress Software's Toolkit product. If you do not have the Toolkit from Progress Software, you will only be able to create deployments where the deployment's Type field (license type) is set to S for source, or P for partner.

Each workspace can own one or more remote sites. Each of these sites has one or more sequential deployments. Each deployment, except for the first, is packaged as an incremental update of just the changes made in the system since the previous deployment to the site.

Each deployment record is numbered sequentially by site. Each deployment contains a Release number field into which the workspace release number to be deployed is Entered. A release number defines a specific configuration of the workspace.

The content of a deployment is determined by comparing the release number of the current deployment with the release number in the previous deployment. Given these two release numbers, Roundtable can use the event history of the workspace to find the changes that occurred in the workspace between the two releases. These changes are packaged as the incremental update.

Roundtable creates incremental updates for both the source files and database schema content of your application. Roundtable utilities for the update or creation of the application databases and for the compilation of procedures are supplied in source form and can be customized and delivered with your application.

Deployments can be made from any workspace. Usually, you deploy from the last workspace in the quality assurance cycle. Follow these steps to deploy to a remote site:

1. Ensure the correct release is available.
2. Select or add a remote site.
3. Add a deployment for that remote site.
4. Build a schema control table list.
5. Create schema data procedures as required.
6. Make update directories.
7. Package and deliver the contents of the update directories to the remote site.
8. Install the deployment at the remote site.
9. Complete the deployment.

#### **4.11.1. Sites & Deployments Descriptions**

A remote site receives packaged updates of the application system. Each site can have

multiple sequential deployments. Each deployment is a software update consisting of the incremental changes in the software system since the previous deployment to the site. Remote sites and deployments are managed in the Sites & Deployments window shown.

This window contains a browse table that lists each remote site, type, status, and PROGRESS version.

Field or Menu Item	Description
Site	Can be deployed to many different sites from a single workspace. The Site Code names each site. Site Codes are validated to be unique across all workspaces.
Type	Type of deployment: Runtime, Development, Query, Source, or Partner. The type of deployment determines the content of the deployment, including whether or not the source is encrypted.
Status	The status of a site can be active or inactive.
Progress Version 6	Is this deployment for a Progress Version 6 site?. This field defaults to NO. If you choose <b>Yes</b> for this field, then the .df files written out for the deployment will be Progress Version 6 compatible.
Notes	Free form notes for the site.
Compile Parameters	<p>This editor contains one or more compile parameters that should be added to the compile statements for procedures compiled at the remote site.</p> <p>Workspace and module compile parameters are ignored when building a deployment. There is no way to have module by module control of your site's compile parameters.</p> <p>If a PCODE object's compile parameters are set to override the workspace compile parameters, then they also override the site compile parameters. The site compile parameters would be ignored. Otherwise, the PCODE object's compile parameters are used in addition to the site compile parameters.</p>
Select	Choose Select to go to the Deployments Menu.
Add	Adds a new site record.
Delete	Deletes the currently selected site.

### 4.11.2. Deployments Menu

The Deployments Menu contains information about each sequential deployment made to

the remote site.

The following table provides the Deployments Menu field descriptions:

Field or Menu Item	Description
Deploy#	You make one or more sequential deployments to each site. Each of these is identified by a sequential number starting at a value of 1 and incrementing with each deployment by 1.
Status	The Status of the deployment is either Work In Process or Complete.
Rel#	Each deployment is associated with a release number. Release numbers identify a particular state of the workspace. You can deploy any release level.
Directory	The Directory field in the deployments table shows both the deployment root directory and schema update subdirectory separated by a comma.

Field or Menu Item	Description
Full Comp	This field can have a value of <b>Yes</b> or <b>No</b> . It controls whether a full compilation is performed at the remote site when the deployment is received. If it is No then only those routines that need to be compiled because of the newly delivered system components are compiled. This value is shown as a Full Compile toggle box as well.
Root Directory	The root directory to build your deployment in.
Sch Upd Subdir	The name of the subdirectory that will contain the deployment's schema information. This subdirectory will reside under the rtb_dbup subdirectory of your deployment root directory. The name of this subdirectory is important as it represents your 4-digit schema version number, or an 8-digit incremental schema update number. See Section 4.12.3, "Schema Release Rules" [4–50].
Add	Choose Add to create a new deployment.
Edit	Choose Edit to edit the deployment.
Delete	Choose Delete to delete a WIP deployment.
Schema	Choose Schema to generate the schema information for the deployment.
Make	Choose Make to generate the deployment package.
Complete	Choose Complete to change the status of the deployment from WIP to Complete.

### 4.11.2.1. Adding a Remote Site

Follow these steps to add a remote site:

1. From the Workspaces Menu, select the workspace for which you want to add a remote site.
2. Choose Ws-config. The Workspace Config. Menu appears.
3. Select the Deployment Menus. The Remote Sites for Workspace window appears.
4. Choose Add.
5. Enter the new site code, license type, status, PROGRESS Version and site notes.
6. Site codes must be unique throughout the system. The same site code cannot be defined under two different workspaces.
7. Roundtable matches the license type against the Encrypt field in the PCODE object definition to determine whether it should encrypt the PCODE objects being deployed. The Encrypt field defaults to a value of "RQD", which means that if the license type is Run-time, Query, or Development, Roundtable should encrypt the PCODE object.
8. The Status is either A (Active) or I (Inactive).
9. Set the Progress Version 6 flag to **YES** if you are deploying to a Version 6 Progress site. The main difference is in the format and content of the .df files for schema update at the remote site.
10. Choose **Quit** to close the Remote Sites Menu.

### 4.11.2.2. Editing a Remote Site

Once the remote site is created, the site notes, status, and Progress version of the site can be changed.

Follow these steps to edit a remote site:

1. From the Workspace Menu, select the workspace.
2. Choose Ws-config. The Workspace Config. Menu appears.
3. Select the Deployment Menus. The Deployment Menu appears.
4. Select the site to edit from the Site table.
5. Choose Edit.
6. Enter the Status, PROGRESS Version, and Site Notes.
7. Press **GO**.
8. Choose **Quit** to close the Remote Sites Menu.

### 4.11.2.3. Deleting a Remote Site

Once a remote site is no longer in use, it can be deleted.

Follow these steps to delete a remote site:

1. From the Workspaces Menu, select the workspace.
2. Choose Ws-Config. The Workspace Config. Menu appears.
3. Select Deployment Menus. The Deployment Menu appears
4. Choose the site to delete from the Site table.
5. Choose Delete. A message appears asking for confirmation.
6. Type **Yes** to delete the site and press **ENTER**.
7. Choose **Quit** to close the Remote Sites Menu.

### 4.11.2.4. Printing the Workspace Remote Sites Report

The Workspace Remote Sites Report provides site information, including the site notes, Progress version, and status, for each remote site.

Follow these steps to print the report:

1. From the Workspaces Menu, select the workspace.
2. Choose Ws-config. The Workspace Config. Menu appears.
3. Select Deployment Menus. The Deployment Menu appears.
4. Choose Print.

### 4.11.2.5. Adding a Deployment

Once a remote site is created and set up, the software can be deployed. See Section 4.11.2.1, “Adding a Remote Site” [4–38]. When creating a deployment, Roundtable determines what the last update to the remote site was, then determines what needs to be deployed.

After adding a deployment, you can change both the release number and deployment directory. See Section 4.6.3, “Editing a Release” [4–26].

Follow these steps to add a deployment:

1. From the Workspaces Menu, select the workspace.

2. Choose Ws-config. The Workspace Config. Menu appears.
3. Select Deployment Menus. The Deployment Menu appears.
4. Find the site to deploy from and choose Select.
5. Choose Add. Complete the Release#, Directory, and Full Comp fields.
6. Press go to save the deployment.
7. Choose **Quit** to exit the Deployment Menu.

OR

Continue with other steps in the deployment process:

- See Section 4.11.2.6, “Editing a Release Number and Directory” [4–40].
- See Section 4.11.2.8, “Building a Schema Update List” [4–41].
- See Section 4.11.2.9, “Making an Update Directory” [4–42].
- See Section 4.11.2.10, “Updating At a Remote Site” [4–42].
- See Section 4.11.2.12, “Completing a Deployment” [4–46].

#### 4.11.2.6. Editing a Release Number and Directory

After creating a deployment, you can change the release number or deployment directory. The deployment directory is where Roundtable writes the update package for the deployment.

Follow these steps to edit the release number and directory:

1. Find the site from the Remote Sites Menu.
2. Choose Select to go to the Deployment Menus.
3. Select the deployment to change (only change WIP deployments).
4. Choose Edit.
5. Type the release you want to use or press the GET key for a lookup.
6. Change the directory, if necessary, in the Directory field.
7. Press go to complete the changes.
8. Choose **Quit** to close the window.

OR



Continue with other steps in the deployment process:

- See Section 4.11.2.8, “Building a Schema Update List” [4–41].
- See Section 4.11.2.9, “Making an Update Directory” [4–42].
- See Section 4.11.2.10, “Updating At a Remote Site” [4–42].
- See Section 4.11.2.12, “Completing a Deployment” [4–46].

#### 4.11.2.7. Deleting a Work-in-process (WIP) Deployment

Once a deployment is created, it has a work-in-process (WIP) status. Change this status to complete after confirming that the deployment is successful. Roundtable ensures that only WIP deployments can be deleted.

Follow these steps to delete a WIP deployment:

1. Find the site from the Remote Sites Menu.
2. Choose Select to go to the Deployment Menus.
3. Select the deployment you want to delete.
4. Choose Delete. A message appears asking for confirmation.
5. Type **Yes** and press **ENTER** to delete the deployment.
6. Choose **Quit** to close the window.

#### 4.11.2.8. Building a Schema Update List

A schema update list is similar to the one used when updating schema objects. To build the list for regular schema updates, Roundtable examines all of the changes made to schema objects in the system and determines their impact on the physical schema tables in the Progress Data Dictionary. To build the schema update list for a deployment, Roundtable compares the schema objects in the workspace with the last version of the schema objects delivered to the remote site. After Roundtable builds a list of tables changes to be delivered to the remote site, it performs an integrity check on these tables. The integrity check ensures that all PFILE and PDBASE relationships are valid.

Follow these steps to build the schema update list:

1. Select the deployment to build the schema update list for.
2. Choose Schema. The Schema Control Screen appears.
3. Choose Build to generate the schema update list.
4. Choose **Quit** to return to the Deployments Menu.

5. Choose **Quit** to return to the Remote Sites Menu.

**OR**

Continue with other steps in the deployment process:

- See Section 4.11.2.9, “Making an Update Directory” [4–42].
- See Section 4.11.2.10, “Updating At a Remote Site” [4–42].
- See Section 4.11.2.12, “Completing a Deployment” [4–46].

### 4.11.2.9. Making an Update Directory

The make update process writes the contents of the deployment into a directory of your choice. Install support tools are also placed in the directory automatically. These support tools are discussed in the next section.

Follow these steps to create an update directory:

1. Select the deployment to make the update directory for.
2. Choose Make to generate the deployment directory.
3. Choose **Quit** to return to the Remote Sites Menu.

**OR**

Continue with other steps in the deployment process:

See Section 4.11.2.10, “Updating At a Remote Site” [4–42].

See Section 4.11.2.12, “Completing a Deployment” [4–46].



To create an encrypted-source deployment, xcode must be in your UNIX PATH. The xcode program comes with Progress Software’s Toolkit product. If you do not have the Toolkit from Progress Software, you will only be able to create deployments where the deployment’s Type field (license type) is set to S for source, or P for partner.

### 4.11.2.10. Updating At a Remote Site

Roundtable provides a set of Progress-specific utilities necessary for installing the software at a remote site. These utilities do not include a packaging utility to create install

disks or utilities to automate the installation from distribution media onto a computer.

The utilities provided by Roundtable perform:

1. Schema update processing
2. Selective and full compile processing

When Roundtable creates a deployment directory, it also creates three subdirectories to support the install utilities. These are the `rtb_inst`, `rtb_idat`, and `rtb_dbup` directories:

Directory	Directory Description
<code>rtb_inst</code>	Contains the generic utility programs and data files for the installation and compilation of the changes copied to the remote site. This directory should remain on the remote site permanently to support future incremental updates.
Files Name	File Description
<code>compctrl</code>	Text file containing a list of compilable procedures with their compile options.
<code>schupd.p</code>	Schema update procedure, part 1.
<code>schupd2.p</code>	Schema update procedure, subroutine for part 1.
<code>schupdp2.p</code>	Schema update procedure, part 2.
<code>compobj.p</code>	Procedure for compiling a deployed Roundtable object.

Directory	Directory Description
<code>rtb_idat</code>	Contains temporary files that can be deleted from the remote site after the update installation is complete.
File Name	File Description
<code>Objctrl</code>	A list of new or modified objects delivered in the deployment. The <code>_update.w</code> procedure posts these changes in the <code>rtb_instcompctrl</code> file.
<code>Delctrl</code>	A list of files to delete from a previously installed configuration of the system. The <code>_update.w</code> procedure performs this task.
<code>Selcomp</code>	A list of selected objects to compile. The <code>_update.w</code> procedure reads this list and marks corresponding objects in the <code>rtb_instcompctrl</code> file for compilation. Roundtable creates this list of objects requiring compilation automatically.
<code>Objcopy.ful</code> and <code>Objcopy.ful</code>	The <code>_update.w</code> procedure allows you to designate an alternate root directory for the R-code of compiled objects to be placed. The <code>objcopy</code> files contain a list of additional files or directories that

Directory	Directory Description
	should be copied into the root directory. These files are created automatically. The ".ful" file will contain all objects that have "objcopy" set to <b>Yes</b> (for a full compile), the ".sel" file will contain just the objects that have changed (for a selective compile).

Directory	Directory Description
rtb_dbupdbver	Contains the database schema update files required to update the application databases at the site. During the update schema process, the files in this directory are copied into rtb_idat where they are processed. The dbver subdirectory name is usually generated by Roundtable.
File Name	File Description
schctrl	Schema update control file. This file contains a list of each database table to be updated in each database used by the application.
*.df	Files for updating database schema.
*d.p	Programs for pre-update file processing.
*f.p	Programs for post-update file processing.

To provide install support for Roundtable deployments into your application, take a copy of the update.w procedure supplied within the Roundtable scripts directory and modify it to suite your needs. The update.w procedure should become part of your application and be located in the root directory of your application. This will ensure that the rtb\_inst, rtb\_idat, and rtb\_dbup directories will be immediately below the directory where \_update.w is stored. Additionally, \_update.w calls the DB Areas function which allows you to assign new tables and indicies, during the product installation, to existing storage areas in the physical database.

The following files are created at the remote site by the installation control procedure or manually by the user. These files are not automatically created by the installation support procedures supplied by Roundtable.

File	Description
Schema.pf	A Progress parameters file that contains the database connection parameters used during the remote site schema update process. This file must connect all of the databases to be updated at the remote site. See the discussion of the _update.w procedure for more information on how this parameter file is processed. The _update.w procedure has support for creating and maintaining the schema.pf file.
Compile.pf	A Progress parameters file that contains the database connection

File	Description
	parameters to be used during the compile process. The _update.w procedure has support for creating and maintaining the compile.pf file.

During execution of \_update.w, various log and error files are created in a directory called rtb\_ilog. Normally, these files can be ignored.

The following log files are created in the rtb\_ilog directory when the schema update procedures are compiled:

File	Description
schcomp.srt	Created by _update.w during the OS-COMMAND when executing schcomp.p. From this file you can determine if execution of schcomp.p began.
Schcomp.log	Created by schcomp.p. Contains the names of the schema update files being compiled.
Schcomp.err	Created by schcomp.p. Contains any compile errors that occur during the compile of the schema update programs. This file is empty if no compile errors occur.

The following log files are created in the rtb\_ilog directory during the pre-schema and schema update:

File	Description
Schupd.srt	Created during the OS-COMMAND when executing schupd.p. From this you can determine if schupd.p execution began.
Schupd.log	Created by schupd.p. Contains information about what table is being updated in the pre-schema and schema update.
Schupd.err	Created by schupd.p. Contains any errors that occur during pre-schema and schema update.
Schema.log	Created by schupd.p. Contains detailed log information about the pre-schema and schema update.

The following log files are created in the rtb\_ilog directory during the post-schema update:

File	Description
Schupdp2.srt	Created during the OS-COMMAND when executing schupdp2.p. From this you can determine if schupdp2.p execution began.

File	Description
Schupdp2.log	Created by schupdp2.p. Contains information about what table is being processed by the post-schema update.
Schupdp2.err	Created by schupdp2.p. Contains any errors that occur during post-schema update.

The following error file is created in the `rtb_ilog` directory when updating the `compctrl` file:

File	Description
objctrl.err	Created by <code>_update.w</code> when updating the <code>compctrl</code> file. If any errors occur during the creation of the <code>comctrl</code> file, they appear here.

#### 4.11.2.11. The Update Process

Follow these steps to perform a sample update process:

1. Shut down databases if this is not a first-time installation.
2. Back up the databases and application directories if this is not a first-time installation.
3. Copy the installation package into the application directories. This means that the `rtb_inst`, `rtb_idat`, and `rtb_dbup` directories exist in your application's root directory.
4. Change the directory to the applications root directory, and run `_update.w`. Depending on your environment and how you installed Progress, your command might look like this: `pro -p _update.w`.
5. After completing the remote site update process you should complete the deployment. See Section 4.11.2.12, "Completing a Deployment" [4–46].

#### 4.11.2.12. Completing a Deployment

Before you complete a deployment, it is good practice to confirm that the remote site successfully received and installed that deployment. That way, if the remote site update did not work properly, you can modify the workspace and restart the deployment process. Once the deployment is complete, the Status field toggles between WIP and Complete.

Follow these steps to complete a deployment:

1. Select the deployment to complete.
2. Choose Complete.

3. Choose **Quit** to close the window.

A new deployment cannot be created until the last deployment has been completed.

### 4.11.3. Procedure Updates and Compiles on Remote Sites

Procedure updates and compiles are performed by the `_update.w` procedure. This procedure performs maintenance on the `rtb_inst/compctrl` file. The `compctrl` file contains a list of each application procedure that can be compiled by your application. `_update.w` updates the `compctrl` file when a new update package is received by reading the contents of the `rtb_idat/objctrl` file and inserting and deleting items from `compctrl`.

The `_update.w` procedure deletes unused application files by reading the list of files to be deleted from the `rtb_idat/delctrl` file.

Use the R-code directory fill-in in the `_update.w` procedure to specify the destination root directory of the R-code in the application. Make this field blank if your R-code will remain in the same directory as the source. If an `rtb_idat/objcopy` file exists, items specified in it are copied into the R-code destination as well. This is very useful if you have bitmaps and other files that must be in your application R-code directory structure.

## 4.12. Database Schema Updates on Remote Sites

Remote site database schemas are updated by the `_update.w` procedure with calls to the `rtb_inst/schupd.p`, `rtb_inst/schupd2.p`, and `rtb_inst/schupd2p.p` procedures. These procedures read information in a PROGRESS database parameters file. `_update.w` finds and connects to one or more databases that are updated with the schema update package delivered in the `rtb_dbup` directory.

If the databases have not yet been created, then you must create the database manually from the `empty.db` supplied with the server platform's copy of Progress before updating its schema from the client running `_update.w`.

A Progress client running `_update.w` can update the schema of a database connected by a networking protocol. However, you must start the database servers before running the `_update.w` procedure on the client machine. The `schema.pf` file should contain all of the required connection information for the client.

The `_update.w` procedure uses the `schema.pf` file to manage the client connection to the databases when these databases are located on database servers. The creation, startup, and shutdown of the database servers cannot be managed by the Progress client running the `_update.w` procedure. Again, if you are installing in a client/server environment, you must ensure that the database servers are available for connection by the Progress client running the `_update.w` procedure.

Because it is important that the Progress client running `_update.w` has uncontested access to database servers during a schema update, please ensure that the servers are started with a `-n 1` argument to allow only a single client connection.

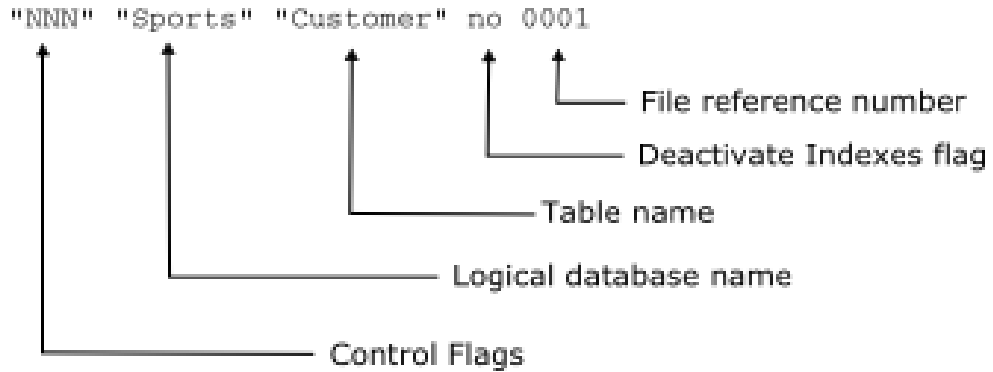
### 4.12.1. Schema Update Process

The schema update process is the most complex process performed during the update at the remote site. This process performs the following actions:

- Runs any Before Schema Update Procedures to dump selected data from the database that would otherwise be destroyed by the schema change.
- Applies the schema updates contained in the `*.df` files delivered with the deployment.
- Runs any After Schema Update Procedures to restore data into the database that was dumped in the first stage of update processing.
- The schema update process reads a control file, called `schctrl`, to manage the three-stage update of each of the connected databases. As the operations specified in the `schctrl` file are completed, the `schctrl` file is modified to record these completed steps. This allows the update schema process to be restarted if it is interrupted.
- The `schctrl` file is a simple text file with one line per database table to be updated. Each line in the `schctrl` file contains the following fields in PROGRESS export format:
- Control flags: Three characters representing the status of the three update stages. Each character is either Y or N, indicating whether the operation has been completed. The stages are Before Schema Update, Schema Update, and After Schema Update.
- Database name: The name of the logical database in which the file being updated is located.
- Table name: The name of the table to update.
- Deactivate indexes?: The option for deactivating the indexes, either **Yes** or **No**.
- File reference number: A four-digit number used to identify the `*.df`, `*d.p`, and `*f.p` files to use during the schema update process. The `*.df` file contains the schema change information. The `*d.p` procedures are run during the first stage, and the `*f.p` procedures are run during the third stage of the update processing.

Here is a sample line from a `schctrl` file:





Because the control flags for the sample line above are "NNN", each of the three processing actions are taken on the file. This means that there will be two procedure files and a schema update file for this table. These files will be named 0001d.p, 0001f.p, and 0001.df. The 0001d.p procedure file is responsible for dumping any data that would otherwise be deleted during the schema update process. The 0001.df file contains Progress data definition language statements describing the schema update to be performed on the table. The 0001f.p procedure is responsible for loading any data back into the database that was dumped by 0001d.p.

#### 4.12.2. Creating and Editing the schema.pf File

The schema.pf file is a Progress parameters file containing the startup parameters necessary to connect to the application databases being updated. In addition to the normal database connection parameters, the schema.pf file must contain a comment line with a release number that indicates the release level of the current database schemas. Here is a sample schema.pf file:

```
#Schema.pf file - used for the update of database schema
#release=0021
-db sports.db -ld sports -l -U "" -P ""
```

Progress considers lines beginning with # in a parameters file to be comments. The \_update.w procedure uses the #release= token to establish the current release level of the database so it can select the correct database update package from the rtb\_dbup directory.

If the schema.pf file is created by the \_update.w procedure, it contains the following default values:

```
#Schema.pf file - used for the update of database schema
#release=empty
```

Enter the database connection parameters manually or choose the **pf Entry** button. The **pf Entry** presents a series of dialog boxes that construct database connection parameters for you and write the parameters into the editor widget containing the parameters file. After creating the database connection parameters, the contents of the editor widget should resemble the following example:

```
#Schema.pf file - used for the update of database schema
#release=empty
-db sports.db -ld sports -l -U "" -P ""
```

The "#release=empty" line indicates that the database has not yet been updated with any schema. As part of the schema update process the "#release=empty" line is replaced with a line containing the release level of the update. For example:

```
#Schema.pf file - used for the update of database schema
#release=0004
-db sports.db -ld sports -l -U "" -P ""
```

### 4.12.3. Schema Release Rules

The release name specified on the #release line in the schema.pf file is derived from the name of the subdirectory in rtb\_dbup that was used to process the schema update. The rules governing the selection of the subdirectory to use in the update are simple:

1. If #release=empty, then find and use the subdirectory in rtb\_dbup that has a name four characters long. If more than one such directory exists, use the one with the highest value. The new #release=value line contains the name of the directory selected.
2. If #release=xxxx, then find a subdirectory in rtb\_dbup that is eight characters long and begins with the value "xxxx". If more than one such subdirectory exists, then select the one with the highest value. The new #release=value line contains the last four characters of the eight-character subdirectory name selected.

By default, the subdirectories created in the rtb\_dbup directory are named using the release number associated with the deployment being made and the previous release. Refer to the following example:

Deploy	Rel	Schema Changes?	rtb_dbup/subdir	Before install #release=	After install #release=
1	12	Yes-full schema	0012	empty	0012
2	13	Yes	00120013	0012	0013
3	14	No		0013	0013
4	15	Yes	00130015	0013	0015

When no schema updates are included in a deployment, no schema update package directory is created, as shown in deployment 3 above. Then the `_update.w` program can determine that no update was delivered and skip the schema update process.

You override the naming of the schema update package directory created during a deployment by specifying it in the deployment directory fill-in in the Sites & Deployments window. Remember that these schema update directories must be either four or eight characters long and have collation values that follow the rules stated above to participate in the update process.

#### 4.12.4. Updating the Database Schemas Dialog Box

When you update the schema of a database at a remote site, the Updating Database Schemas dialog box appears. This dialog box displays each step of the database schema processing.

Each possible step in the processing is audited by this dialog box. The steps include:

1. Scanning the `rtb_dbup` directory for schema update packages. The release number in the `schema.pf` file is used to identify which schema update package should be used in the update process. The contents of the selected schema update directory are copied into `rtb_idat`. This step is skipped if a `rtb_idat/schlock` file exists (see step 3).
2. Expanding the `rtb_idat/schctrl` file for database "copies" identified in the `schema.pf` file. See Section 4.12.5, "Database Copies" [4–52]. This step is skipped if a `rtb_idat/schlock` file exists.
3. Writing a process lock file named `rtb_idat/schlock`.
4. Compiling the schema update subroutine procedures.
5. Running the database dump procedures to preserve the data that would be lost because of the schema updates.
6. Updating the schema for each database.
7. Running the database load procedures to restore dumped data.

8. Removing the process lock file `rtb_idat/schlock`.
9. Removing the temporary files copied from the schema update package directory into `rtb_idat`.

### 4.12.5. Database Copies

In many Progress applications, it is useful to create copies of the application database. For example, each department in a company might have a separate application database but use the same application code. Each of these database copies must have the same cyclical redundancy check (CRC) stamp to use the same compiled application code. The update process can keep the CRC in each of these databases in sync by applying schema updates against each database in exactly the same order.

To update these database copies, the installation process must know about each database copy created at the site. This is done by adding a line in the `schema.pf` file that identifies the database copy. Refer to the following `schema.pf` example:

```
#Schema.pf file - used for the update of database schema
#release=0004
-db sports.db -ld sports -1 -U "" -P ""

#dbcopy=sports sports2 sports2
-db sports2.db -ld sports2 -1 -U "" -P ""
```

This `schema.pf` file contains two database connections and a special `#dbcopy=` line that instructs the schema update process to treat the `sports2` database as if it has the same schema as the `sports` database. When the two databases start out with the same CRC stamp, the install process keeps them CRC compatible by applying schema updates to each in exactly the same order.

The `#dbcopy=` line requires three values.

- First, the logical database name of the database whose schema is duplicated in the database copy.
- Second, the logical name of the database copy.
- Third, the name of a temporary directory where data dumped from the `dbcopy` database should be placed in stage one update schema processing. This prevents data dumped from the first and second databases from being confused.

The schema update process performs an additional step when multiple database copies exist. This additional process expands the content of the `rtb_idat/schctrl` file to include entries for each of the files requiring update in each database copy. An additional line is

added to the top of the `rtb_idat/schctrl` file with the token `EXPANDED` so that the `rtb_idat/schctrl` file never expands twice. This makes it important to register all database copies in the `schema.pf` file before running the update schema process.

If the database copy is created sometime after the first installation of your application, the new copy must be created from a copy of an existing database. There are strategies for accomplishing this:

- Maintain at least one copy of the application database at each site that is never populated with data. This database can be used as the source database when creating a new database.
- Make a copy of an existing database and then purge data from it. This is generally a bad idea because Progress does not shrink the footprint of a database on disk. This can be circumvented by a dump and reload of data for the source database.

#### 4.12.6. Deploying Both a Full and Incremental Schema Update

A single schema update package directory is created when you create a deployment. However, the `_update.w` process can recognize and process more than one database update package directory if they exist. This allows you to assemble a multi-release deployment.

A multi-release deployment is useful if you must send a deployment to sites that might be at different release levels of your software. You create a multi-release deployment by combining the contents of one or more deployments into a single deployment package.

For example, if the latest release of your system is 8.0A and you want to distribute the same package to both existing 7.3C clients and completely new clients, you would need to create a multi-release deployment.

Follow these steps to create a multi-release deployment containing the full source and schema for the latest release level, as well as the schema update information to bring an existing application database from a previous schema release level to the latest schema release level.

1. Create a new site.
2. Create a deployment in your new site using the desired release level. Name the schema update directory with the four-character name of the latest schema release level.
3. Build the schema update table and make the deployment directory.
4. Create a second new site.
5. Create a deployment in the second new site with the release level where you want this package to upgrade existing clients.
6. Complete the deployment described in the previous step. It is not necessary to build

that deployment.

7. Create a second new deployment in the second new site. The schema update directory must have an eight-character name. The first four characters must be the name of the previous schema release level. The last four characters must be the name of the latest schema release level.
8. Build the schema update table and make the deployment directory.
9. Copy the eight-character schema update subdirectory from the second deployment's `rtb_dbup` directory. Copy that directory into the `rtb_dbup` subdirectory of the first deployment.

The first deployment directory is now a multi-release deployment directory. The `rtb_dbup` directory contains a four-character subdirectory named with a complete definition of the latest schema release level. The `rtb_dbup` directory also contains a subdirectory with an eight-character name, which contains the schema update information necessary to update an existing application database from the previous schema release level to the latest schema release level.

It is not necessary to retain the Roundtable site records created in these steps. Also, it is not necessary to keep the second deployment directory on disk. It was only created so that you could copy the incremental schema update directory from it.

---

# Task Management

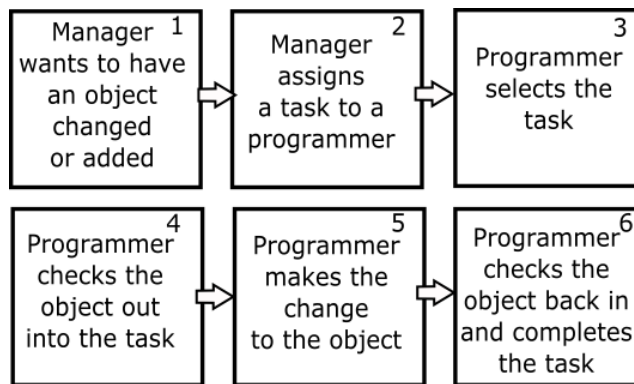
## 5.1. Introduction

After you set up your system and create your products, product modules, and workspaces, you are ready to program. The first step is to create tasks. A task defines work to be done in a workspace.

An object is a component of your Progress application: a program file, a file, a field, a text file, or a part of your database schema definition. Objects are created and modified under the current task. This chapter explains both tasks and objects in depth.

## 5.2. What is Task Management?

Task management is the process of changing or adding objects. The following flow chart illustrates the task management process:



### 5.2.1. Benefits of Task Management

Task management is useful because it:

- Helps programmers track work done in logical units (tasks)
- Facilitates check-in of many objects at once
- Gives managers access to important information about the work being performed in the

workspace

- Allows the programmer to keep track of many concurrent tasks
- Allows programmers and managers to see quickly what others are doing in the workspace

## 5.2.2. How Objects Relate to Task Management

You can add different types of objects to your workspace and use different views to show the objects assigned to your workspace and how they interrelate. You can also see where and how an object is used in your system, and print several object reports.

## 5.3. Task Maintenance

The task management system in Roundtable allows programmers to work together on complex projects by breaking up activities into logical units of work. All modifications to objects in a workspace are completed under a task number. Creating tasks is a simple process, and tasks provide a simple way of organizing many concurrent activities. Each task is specific to a single workspace, manager, and programmer.

After you set up your Roundtable system, you are ready to begin work. If you have many programmers working on one project, control and track this work by dividing it into tasks.


Task maintenance covers adding tasks, editing tasks, deleting tasks, selecting tasks, completing tasks, and printing task reports.

### 5.3.1. The Task Maintenance by Workspace Menu

The following table lists the Task Maintenance by Workspace Menu fields:

Field	Description
Task #	Task numbers are assigned by the system.
Workspace ID	Workspace with which that task is associated. Only one workspace can be associated with each task.
Entered	Date the task was added to the system.
Completed	Date the task was completed.
Compl. By	Userid of the person completing the task.
Manager	Manager's userid, defaults to userid of user adding task or you can select from a list using the Manager combo-box.
Programmer	Programmer's userid, defaults to userid of user adding task or you can select from a list using the Programmer combo-box.
User Ref	User's reference. This is an indexed field which is not used internally by Roundtable.



Field	Description
Directory	<p>A fill-in for the full path to a task directory. If you specify a share status other than Central, you must fill this in. Objects from the workspace, checked out under the task, populate this directory structure. In the task directory, Roundtable creates the portion of the workspace directory hierarchy in which the object is stored. For example, if an object is stored in a subdirectory named .ar in the workspace, then the .ar subdirectory is created in your task directory when you check out the object under the task.</p> <p> A task path should not be a relative path.</p>
Share	Share status for this task. Possible values are: Task, Group, Public, and Central.
Summary	A short description of the task.
Notes	Detailed description of the task.

### 5.3.2. Task Selection By Programmer Menu

This screen displays the status, task number, and summary of each task. Each task has a status of either W (work-in-process) or C (complete).

The following table lists the Task Selection By Programmer Menu fields:

Field	Description
Status	The status is either Work in Progress (WIP) or complete.
Task #	Displays the task number assigned to each task by the system.
Summary	A short description of the task.

The following table lists the Task Selection By Programmer Menu fields:

Field	Description
Task #	Displays the task number assigned to each task by the system.
Workspace ID	Displays the workspace ID code. Only one workspace can be associated with a task.
Entered	Displays the date the task was created.
Completed	Displays the date the task was completed.

Field	Description
Compl. By	Displays the ID of the person who completed the task.
Manager	A fill-in for the ID of the manager assigning the task. Defaults to the userid of the user adding the task.
Programmer	A fill-in for the ID of the programmer responsible for the task. Defaults to the userid of the user adding the task.
User Ref	A fill-in for your own use. This field is indexed but unused by Roundtable. You can build your own reports to get information from the repository quickly using your own key values.
Share	Choose one of the following share statuses for the task: Central: Programmer modifies objects in the workspace directory. Others can see changes immediately. Task: Roundtable checks out objects to a task directory where the programmer modifies them. The changes are not seen by others. The last completed version of the checked out object remains in the workspace directory for others to use. Group: Roundtable checks out objects to a task directory where the programmer modifies them. The changes are not seen by others until the programmer executes the Update Group/Public Source option, which copies the modified objects into a group directory. Other programmers can choose to see these changes by attaching one of their tasks to the group. The last completed version of the objects remains in the workspace directory. Public: Roundtable checks out objects to a task directory where the programmer modifies them. The changes are not seen by others until the programmer executes the Update Group/Public Source option, which copies the changes into the workspace directories.
Directory	A field for the full path to a task directory. If you specify a share status other than Central, you must complete this field. Objects from the workspace, checked out under the task, populate this directory structure. In the task directory, Roundtable creates the portion of the workspace directory hierarchy in which the object is stored. For example, if an object is stored in a subdirectory named .ar in the workspace, then the .ar subdirectory is created in your task directory when you check out the object under the task. The task directory must be a fully qualified directory name, beginning with the (/) character.
Summary	Short summary description of the task.
Notes	Detailed description of the task.

### 5.3.2.1. Share Status

The Roundtable share status feature allows multiple programmers to work in the same workspace, isolating each programmer's work so that other programmers are not affected. Depending on the share status, the other programmers might see objects that are currently being worked on. The share status determines the location of the source files modified under the task.

When running or compiling your application, Roundtable automatically modifies the Progress PROPATH so that directories are always searched in this order:

1. Task directory (if one is defined for the current task)
2. Group directories (if the current task belongs to any groups)
3. Workspace directory

You can change the share status of an object or task at any time. Roundtable moves the objects to and from the workspace directory as required. It is possible to have some objects in a task with a different share status than that of the task. However, if a task has no task directory assigned, then all objects in that task must have a share status of Central. As objects are checked out under a task they are given the share status specified for the task by default.

1. **Central:** Programmer modifies objects in the workspace directory. Others can see changes made immediately. No task directory is necessary if all objects under a task have this status.
2. **Task:** Roundtable checks out objects to a task directory where the programmer modifies them. The changes are not seen by others. The last completed version of the checked-out object remains in the workspace directory for others to use.
3. **Group:** Roundtable checks out objects to a task directory where the programmer modifies them. The changes are not seen by others until the programmer executes the Update Group/Public Source option, which copies the modified objects into a group directory. Other programmers can choose to execute the changes by attaching one of their tasks to the group. The last completed version of the objects remain in the workspace directory.
4. **Public:** Roundtable checks out objects to a task directory where the programmer modifies them. The changes are not viewed or executed by others until the programmer executes the Update Group/Public Source option, which copies the changes into the workspace directories.

If you specify a share status other than Central, you must supply a full path to a task directory. Objects from the workspace, checked-out under the task, populate this directory structure. In the task directory, Roundtable creates the portion of the workspace directory hierarchy in which the object is stored. For example, if an object is stored in a subdirectory named .ar in the workspace, then the .ar subdirectory is created in your task directory when you check out the object under the task.

You can use the same task directory for one or more tasks. This makes all changes in each task sharing that directory visible to each other without having to define groups. Tasks belonging to different programmers should never share the same task directory.

### 5.3.2.2. Adding a Task

In your role of manager, you distribute work by assigning tasks to programmers. Each task is specific to one workspace, one manager, and one programmer. Each task has a summary that is descriptive of the entire task. You can **ENTER** more detailed information in the note field.

You can add a task by selecting either Tasks or Workspaces from the Main Menu.

To Add a Task from the Main Menu, follow these steps:

1. From the Main Menu, choose Tasks. The Task Selection By Programmer menu appears.
2. Choose Toggle. The Task Maintenance By Programmer menu appears.
3. Choose Add and complete the fields.
4. Press the **GO** key to complete the task entry.
5. Choose **Quit** to exit this screen.

A Tasks menu item is also available on the following three menus:

1. Workspaces Menu
2. Workspace Objects Menu
3. Workspace Modules Menu

Follow these steps to add a task from any of these three menus:

1. Choose Tasks. The Task Selection By Workspace Menu appears.
2. Choose Toggle. The Task Maintenance By Workspace Menu appears.
3. Choose Add and complete the task fields.
4. Press the **GO** key to complete the task entry.
5. Choose Select to select your new task or Quit to exit without selecting this task.

### 5.3.2.3. Editing a Task

Although you can never change the task number, you can change the Summary, Manager,

Programmer, Share Status, and User Ref fields, and the notes for a task you manage.

You can edit tasks from the following Workspaces menus:

- Workspaces Menu
- Workspace Objects Menu
- Workspace Modules Menu

Follow these steps to edit a task from any of the three menus:

1. Choose Tasks. The Task Selection By Workspace Menu appears.
2. Choose Toggle. The Task Maintenance By Programmer menu appears.
3. Choose Edit and edit the fields.
4. Press the **GO** key to complete the task entry.
5. Choose **Quit** to exit this screen.

#### 5.3.2.4. Deleting a Task

You cannot delete a task if objects are checked out or have been checked in under the task. If you change your mind about the task right after you add it, you can delete it.

You can delete tasks from the following Workspaces menus:

- Workspaces Menu
- Workspace Objects Menu
- Workspace Modules Menu

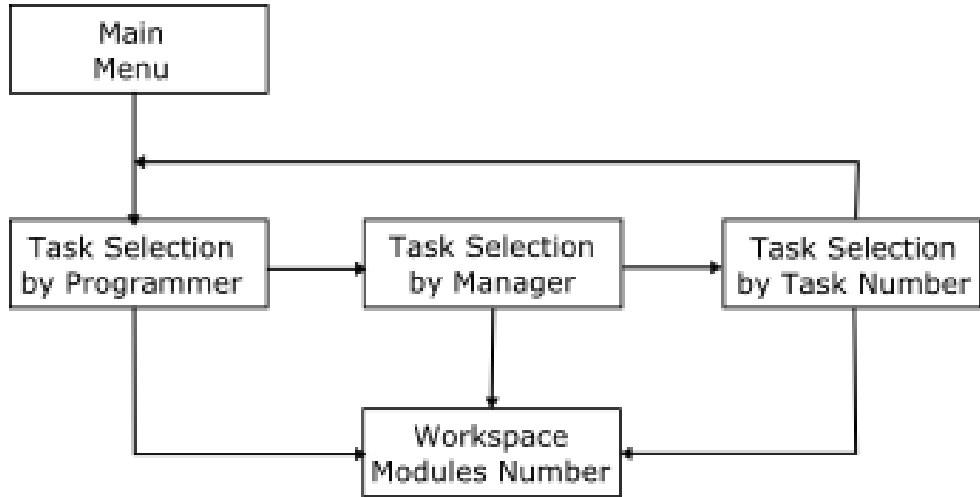
Follow these steps to edit a task from any of the three menus:

1. Choose Tasks. The Task Selection By Workspace Menu appears.
2. Choose Toggle. The Task Maintenance By Workspace Menu appears.
3. Choose Delete. The message, "Delete this record" appears.
4. Type Yes to delete.
5. Press the **GO** key to complete the deletion.
6. Choose **Quit** to exit this screen.

#### 5.3.2.5. Selecting a Task

When programmers want to work on a task, they generally choose the Tasks option from the Main Menu to go to a Task Selection Menu. Then they move to the task they want to work on and choose the Select option to go to the Workspace Modules Menu for the workspace associated with the selected task.

There are three task selection menus available from the Main Menu. Each provides the user with different selection criteria as shown in the following figure.



The contents of a Task Selection screen are sorted by task status so that Work in Progress (WIP) tasks are shown at the top of the table.

For each of the Task Selection Menus there is also a Task Maintenance Menu which shows the full information of a task rather than a table of tasks. Choose the Toggle option from a Task Selection Menu to go to a Task Maintenance Menu. There is a Select option on the Task Maintenance screen as well.

A Task Selection Menu and a Task Maintenance Menu are also available from the Workspaces Menu, Workspace Modules Menu, and Workspace Objects Menu. These task menus only show tasks for the currently selected workspace.

Here are the various places that a Task Selection Menu can be found:

- Main Menu#Tasks
- Main Menu#Workspaces#Tasks
- Main Menu#Workspaces#Module Selection#Tasks
- Main Menu#Workspaces#Module Selection#Object Selection for Module#Task

Follow these steps to select a task:

1. From a Task Selection or Task Maintenance Menu, find the task that you want to select.
2. Choose Select. Roundtable selects the task and returns to the previous menu.



Roundtable displays the currently selected task number in the status bar near the bottom of your screen.

### 5.3.2.6. Completing a Task

After you have made all the necessary changes, you can complete the task. Completing the task gives all objects in the task a C (complete) status. Before completing the task, ensure that:

- All objects have a share status of Central
- The schema changes are updated to the databases
- All objects compile correctly

If Roundtable finds an object with a share status other than Central, use the Objects in Task tool to change the share status to Central. See the Changing the Share Status of a Task section. If Roundtable finds an object that does not compile, you must fix the problem before completing the task.

1. From the Task Selection menu, choose Toggle. The Task Maintenance Menu appears.
2. Find the task that you want to complete by using the Find option, or by paging through the task with the down arrow key.
3. Choose Compl. Messages appear while Roundtable compiles and completes objects in the task.



If Roundtable cannot compile any of the objects, then Roundtable displays the compile errors. For each program that fails to compile, Roundtable only displays the compile errors for five seconds. Press the Spacebar to keep the error messages on the screen longer. Once Roundtable has finished compiling the rest of the programs in the task, the following error message dialog box appears.

4. Type Yes and press the **GO** key to view the error list and determine which objects would not compile. Type No and press the **GO** key if you do not need to view the

error list.

### 5.3.2.7. Task Report

The Task Report provides information on each of the object versions created under the selected task. This report groups the objects by object type and provides the following information for each object:

- Object name
- Description of the object
- Version number
- Current status
- Subtype (for PCODE objects)

For PFIELD objects, the report also lists the type, extent, number of decimals, label, location, format, initial value, and other field information.

Follow these steps to print the task report:

1. From the Task Selection Menu, find the table that you want to report on.
2. Choose Report. The Task Reports Menu appears.
3. Select the Task Report.
4. Select the desired report options:
  - Select Show task notations? to print the task notations.
  - Select Show Objects in Task? to print all the object versions created by this task.
  - Select Object Details? to print details about the object.
  - Select Show Update History? to print the programmer's notes.
5. Press the **GO** key.

### 5.3.2.8. Object Orphans in Task Report

The Object Orphans in Task Report gives you the name and number of any orphan records relating to objects under the task. Follow these steps to print the Object Orphans in Task Report:

1. From the Task Selection Menu, select the task that you want to report on.



2. Choose Report. The Task Reports Menu appears.
3. Select the Object Orphans in Menu Report.

## 5.4. Task Activities

Now that you know the basic procedures for working with tasks, you can explore more ways to work with tasks, including:

- Listing Objects in a task
- Finding a task
- Clearing a task number
- Building a `runtask.p` procedure

### 5.4.1. Checking Out an Object

You cannot check out an object that already has a W (work in process) status because it is already checked out by another Roundtable user.

Follow these steps to check out an object:

1. Select the Task that you want to check out with the object.
2. From the Workspace Objects Menu, find the object that you want to check out.
3. Choose Select. The Object Menu for the object's type appears (PCODE, DOC, PFIELD PFILE, or PDBASE).
4. Choose Ver. The Versions Menu appears.
5. Select the version level to increment by choosing Version, Revision, or Patch. Determine the version level according to your company's policy. A patch might be used for a minor change, a revision for a major change, and a version for a complete overhaul. For example, if the version number is 010000 and you choose Revision, the version number is incremented to 010100.
6. If the source of the OS file in the workspace directory differs from the source in the repository, when an object is checked out, a warning dialog appears. You may select one of the two files to work with or select the compare option.
7. This resolves the discrepancy, or if there is none, and the object is now checked out into your task.
8. Roundtable returns you to the Object Menu. Note that the Version field for the object has changed, and its status is now W (Work in Process).

### 5.4.2. Creating an Object Variant

Roundtable allows you to create more than one variation of the same object called a variant. Variants have the same name and type but belong to different product modules. See the Object Variants section.

Follow these steps to create an object variant. When you create an object variant, it will be Version 1.0.0, with a Work In Process (WIP) status. The original object cannot be WIP when you are creating a variant of it.

1. From the Workspace Objects Menu, choose Add. The Add or Assign Object dialog box appears.
2. Type New in the New/Assign field, Enter the object type for the object that you are creating a custom variant for, and press the **GO** key. The Adding New Object dialog box appears.
3. Enter the name of the product module that your new object variant will be assigned to. This must be a different product module than the one that the original object is assigned to. Enter the Object Group and Level.
4. Enter the object name. This must be the same as the original object's name. Press the **GO** key.
5. Enter yes in this dialog box. The object variant is created and displayed, and you are returned to the Workspace Objects menu. You can see in the object details frame that your new object variant is Version 1.0.0, and its status is W for Work In Process.
6. Choose Select. The object type's menu appears. You can see that this object's product module is the one that you specified in step three. Your new custom variant is now a part of your workspace. The original object variant is no longer assigned to the workspace.

### 5.4.3. Checking in an Object

You cannot check in an object until it has an update status of Current. A PCODE object's update status is changed to Current when the object has a share status of Central and a successful compile with xref has been performed. A schema object's update status is changed to Current after a successful schema update process. The check in process changes the object's status to C (complete) and permanently stores the contents of the object version in the repository.

Follow these steps to check in an object:

1. Select the task that you want to check in with the object.
2. From the Workspace Objects Menu, find the object that you want to check in.
3. Choose Select. The Object Menu for the object's type appears (PCODE, DOC,

PFIELD, PFILE, or PDBASE).

4. Choose Ver. The Versions Menu appears.
5. (Optional) Choose Full-source. Normally, this option is not used. This menu item toggles the Full Src? Field between Yes and No. Toggle to Yes to store the full source code of the object. Toggle to No to just store the changes (delta) to the source since the last version. This option is not available for schema objects or for PCODE and DOC objects that track binary files, because these objects are not stored as incremental differences.
6. Choose Complete. The object version is stored permanently in the repository and the Status of the object is changed to C (complete).

#### 5.4.4. Listing Objects in a Task

Each time an object is checked out from the workspace into a task, it is assigned a new version number. The Object Version Created Under Task Menu displays a table that lists all the objects assigned to a task. The table displays the object name, object type, version number, and status for each object created under the specified task.

From this screen, you can also:

- Change the share status of one of the object versions.
- Change the share status of all of the object versions.
- Promote all objects with a share status of Group from the task directory to the group directories for each group with which the task is associated.
- Promote all objects with a share status of Public from the task directory to the workspace directory.

Follow these steps to display the object versions created under a task:

1. From a Task Selection Menu, choose Obj.

OR

From a Task Maintenance Menu, choose Objects. The Object Version Created Under Task Menu appears.

This screen contains all the object versions created under this task. For each object, the screen lists the object name, type, version number, and update status. The status column displays both the share status of the object and its current update status. The update status of the object is either complete, modified, new, or current. If the object is complete, it has been checked in. If it is modified, it needs to be compiled or updated. If it is current, no changes have been made since the last compile or update.

If it is new, its share status has just changed or it was recently added to the task.

2. Choose **Quit** to return to the previous screen.

### 5.4.5. Changing the Share Status of an Object

Follow these steps to change the share status of an object.



If you want to change the share status of an object in the task to something other than Central, the task must have a task directory defined.

1. From either the Task Selection Menu or the Task Maintenance Menu, find the task form which the object was checked out.
2. Choose Objects. The Object Version Created Under Task Menu appears.
3. Select the object for which to change the share status.
4. Choose Task, Group, Public, or Central. Roundtable changes the share status of the object.
5. Choose **Quit** to close the screen.

### 5.4.6. Changing the Share Status of All Objects in a Task

Follow these steps to change the share status of all objects in a task:

1. From the Object Versions Created Under Task Menu, choose All. The Change Status of All Objects dialog box appears.
2. Complete the New Share Status and Old Share Status fields and press the **GO** key.
3. Choose **Quit** to exit the screen.

### 5.4.7. Promoting Task Objects

When an object has a share status of Group or Public, the work performed on the object is visible only in the task directory until it is promoted. The promotion process copies the source files of the object to group and/or workspace directories as necessary. You choose when this promotion occurs. This intermittent update of the shared group directories or workspace directories can be very helpful when you have not finished working on a given object, but it is necessary for others to use or review that work.

1. From the Object Version Created Under Tasks Menu, choose Update. The Update from task directory to group and workspace directories dialog box appears.
2. Complete the fields and press the **GO** key.
3. Choose **Quit** to return to the previous screen.

### 5.4.8. Finding a Task

If you know a task number, you can use the Find option on the Task Selection Menu or the Task Maintenance Menu to find the task quickly.

1. From the Task Selection By Workspace Menu, choose Find. The **ENTER**-task num field appears.
2. Enter the task number and press the **GO** key. If the task is found, Roundtable selects the task in the table.
3. Choose **Quit** to return to the previous screen.

### 5.4.9. Selecting No Current Task

To browse a workspace without a current task, choose Zero from the Task Selection Menu or the Task Maintenance Menu.

### 5.4.10. Building a Runtask.p Procedure

If you need to work with your task outside of Roundtable, use the runtask.p program to set up your task environment. Roundtable creates the procedure in the task directory. Runtask.p generates the path to all of your task directories and workspace sources. You must have a task selected to build a runtask.p procedure.

For example, you can use the runtask.p procedure to test your program while running your own main menu system. To do this, incorporate runtask.p into your own main menu program or run it before running your own main menu program.

Follow these steps to build a runtask.p procedure.



You must have a task selected to perform this procedure.

1. From the Task Maintenance Menu, choose Build.



Roundtable builds the `runtask.p` program. Roundtable only takes a second to perform this function and there is no visual indication when it is done.

2. Choose **Quit** to return to the previous screen.

## 5.5. Task Groups Maintenance

Task groups allow the changes in one or more tasks to be seen at the same time. For example, use task groups when two programmers are working on different tasks but must see each other's work.

Task groups are useful only with tasks that do not have a Central share status because work done under a task with a Central share status is visible to all users of the workspace. Remember that Roundtable sets the `PROPATH` so that code is seen first in any local task directory, then in any group directories, and finally in the workspace directory.

Task group directories must reside on the same drive as the workspace and use the same drive mappings as the workspace to which they belong.

A task group is assigned a group directory, which is never accessed directly by a programmer. Instead, the task promotion process copies the source from the task directories into the group directory at the request of the programmer owning the task.

A task assignment associates a task with a group. A task can be associated with up to nine task groups. All of these task groups are updated when the task promotion process is executed.

### 5.5.1. Task Groups Window Description

There are two menus for maintaining task groups. The first menu is the Task Groups Menu.

The following table describes the fields in the Task Groups window:

Field	Description
Task Group	You supply a task group code when you add a new task group.
Directory	This is the path to the directory under which code from tasks belonging to the task group will be placed when promoted by the owners of the tasks.
Add	Adds a new task group.
Edit	Edits the task group directory.

Field	Description
Delete	Deletes the task group.
Find	Finds a task group.
Quit	Returns to the previous screen.

The second menu for maintaining task groups is the Task Group Assignments menu.

The following table lists the fields in the Task Assignments To Group:

Field or Menu Item	Description
Task#	The task number of a task associated with the group.
Summary	The summary description of the task.
Add	Adds a new task to the task group.
Delete	Deletes a task from the task group.
Find	Finds a task already assigned to the task group.
Quit	Returns to the previous screen.

### 5.5.2. Adding a Task Group

Follow these steps to add a new task group:

1. If the Roundtable security is enabled, login as Sysop.
2. From either the Workspaces Menu or the Workspace Modules Menu, choose Ws-config. The Workspace Config. Menu appears.
3. Choose Task Groups. The Task Groups Menu appears.
4. Choose Add.
5. Complete the fields and press the **GO** key.
6. Choose **Quit** to return to the previous screen.

### 5.5.3. Editing a Task Group Directory

When you edit a task group directory, the files in the old directory are moved to the new directory.

Follow these steps to edit a task group directory:

1. If the Roundtable security is enabled, login as Sysop.
2. From either the Workspaces Menu or the Workspace Modules Menu, choose Ws-config. The Workspace Config. Menu appears.
3. Choose Task Groups. The Task Groups Menu appears.
4. Select the task group and choose Edit.
5. Edit the fields and press the **GO** key.
6. Choose **Quit** to return to the previous screen.

#### 5.5.4. Deleting a Task Group

Before you can delete a task group, you must delete each task group assignment that it owns. Deleting task assignments does not affect the associated tasks in any way.

Follow these steps to delete a task group:

1. If the Roundtable security is enabled, login as Sysop.
2. From either the Workspaces Menu or the Workspace Modules Menu, choose Ws-config. The Workspace Config. Menu appears.
3. Choose Tasks Group. The Task Groups Menu appears.
4. Select the task group and choose Delete. The message, "Delete this record" appears.
5. Type Yes and press the **GO** key.
6. Choose **Quit** to return to the previous screen.

#### 5.5.5. Adding a Task Group Assignment

Follow these steps to add a task group assignment:

1. If the Roundtable security is enabled, login as Sysop.
2. From either the Workspaces Menu or the Workspace Modules Menu, choose Ws-config. The Workspace Config. Menu appears.
3. Choose Task Groups. The Task Groups Menu appears.
4. Select the Task Group and choose Select. The Task Group Assignments Menu appears.
5. Choose Add and complete the fields.
6. Press the **GO** key to complete the Task Group Assignment.
7. Choose **Quit** to return to the previous screen.



### 5.5.6. Deleting a Task Group Assignment

Follow these steps to delete a task group assignment:

1. If the Roundtable security is enabled, login as Sysop.
2. From either the Workspaces Menu or the Workspace Modules Menu, choose Ws-config. The Workspace Config. Menu appears.
3. Choose Task Groups. The Task Groups Menu appears.
4. Select the Task Group and choose Select. The Task Group Assignments Menu appears.
5. Select the Task and choose Delete. The message, "Delete this record" appears.
6. Type Yes and press the **GO** key.
7. Choose **Quit** to return to the previous screen.

## 5.6. Task Notations

Roundtable provides a simple screen in source form (rtb/p/rtb0298.p) for maintaining additional task information. The table maintained by this screen is called `rtb_tnot` (Task Notes). Any number of `rtb_tnot` records can be related to a given task record. This table includes a number of free-form fields, and fields for tracking such things as estimated and actual hours.

Rtb0298.p is intended to be modified to fit your own task information needs. How you use the fields in the `rtb_tnot` table is entirely up to you.

Roundtable does not use the `rtb_tnot` table except in these three places:

- Rtb0298.p - Task Notes maintenance screen.
- The Task report. It shows task notations for the tasks being reported on.
- When a WIP task is deleted, any `rtb_tnot` records associated with it are deleted.

The Task Notations screen is accessed by the Info menu-item from the Task Selection or Task Maintenance window.

---

---

# Objects

## 6.1. Introduction

All workspaces are comprised of collections of Roundtable objects. These objects are of one of the following types:

- PCODE: A collection of up to nine files that define a component of the software application
- PFIELD: A database schema definition for a field
- PFILE: A database schema definition for a table
- PDBASE: A database schema definition for a database
- DOC: A special type used for documentation objects

## 6.2. The Repository

Objects are stored in the Roundtable repository. The Roundtable repository can store many different versions of the same object. When you add a new object, you create the first version of that object, which is numbered 01.00.00. This version code has three parts: Version Level, Revision Level, and Patch Level. All objects are created in a workspace with an initial status of work-in-process (WIP). When you create an object, you assign it a Name, Type, and Product Module. The Product Module, Type, Name, and Version Code constitute the unique key of the object in the repository.

An object does not become a permanent part of the repository until you check it in. Checking it in changes the object's status from WIP to Complete. It is not possible to delete an object from the Roundtable repository once it has been completed. However, you can remove the object from the workspace. Remember that a workspace is a collection of objects that represents a configuration of your software application. Deleting an object removes the object version from this configuration.

One last point about deleting object versions: if you delete the first version of an object while it still has a WIP status it does not become a permanent object in the repository. So, if you inadvertently create an object, you can delete it.

You can assign any object version available in the repository to any workspace. Since the object already exists, such assignments are not considered part of your current task.

### 6.2.1. Object Versioning and Tasks

Before you can create or modify any object you must have a current task . It is not unusual to create or modify multiple objects under the same task. The task number, under which the object version was created, is stored in the object. This provides traceability between the object version and the task. For information on using tasks, see Section 5.2, “What is Task Management?” [5–1].

When you want to modify an existing object, you must check out the object under a task. This creates a new version of the object and gives the object a WIP status. While you have the object checked out, no other programmer, in the current workspace, can check it out. If the object is subsequently checked out in some other workspace, Roundtable warns the user that an object orphan condition exists.

When you check out the object, you choose whether to increment the Version Level, Revision Level, or Patch Level of the version code. The new object version becomes a permanent resident of the repository, and the object’s status changes to complete.

### 6.3. Workspace Objects Menu

This menu allows the programmer to Browse, Add, Edit, and Delete object assignments in a workspace module.

The Object Selection for Module screen includes the following information:

Field	Description
Object Group	Contains a code used to group objects together within the module. The field can be changed at any time.
Level	Contains a numeric code used to group objects within an object group. This field can be changed at any time.
Type	The object type: DOC, PCODE, PDBASE, PFILE, or PFIELD.
Description	Object name/short description. For multi-part objects the object name is always the name of the first object part.

#### 6.3.1. Object Details Frame

The contents of this screen changes as the user scrolls through the objects in the top screen. The information displayed in the screen varies depending on the object type displayed. For instance, if the object currently displayed is a PFIELD, an abbreviated field definition is shown.

The Workspace Objects Menu includes the following menu items:

Menu Item	Description
Select	Selects the object to work on and Goes to the appropriate menu based on the object type.
Add	Adds/Assigns the object to the workspace.
Edit	Edits the Object Group and Level of the current object.
Del	Deletes (or de-assigns) the current object from the workspace. This option does not actually delete the object version unless the object version is WIP. Previous object versions of the object are not affected. If a previous version of this object was assigned in the workspace, this option asks if you would like to revert to it. This would cause the previously assigned version to be reassigned to the workspace. Note that this option warns you if the object is used by other objects in the system. For schema objects like PFIELD and PFILE it might not be possible to completely delete the object assignment from the workspace if the object is used by another object.
Find	Finds an object by group or by name.
Tasks	Goes to Task Selection By Workspace Menu.
Compile	Selective Compile.
Run	Runs the current object if it is a runnable program. If not, it looks for the first (by Level) object in the current object's Object Group that is runnable.
More	<p>The More Menu includes these options:</p> <p>Grab This option allows you to quickly get a field lists and assignment lists from the dictionary. You may select one or more field names from a file to include in a text file that can be "read" into the editor while modifying a PCODE object. The fields in the text file can take a list or assignment form as shown below:</p> <p>List form:</p> <p>Buffer1.fieldname[9]</p> <p>Assignment form:</p> <p>Buffer1.fieldname[9]=Buffer2.fieldname[9]</p> <p>Note that the utility will ask for Buffer1 and Buffer2 and that all files from connected databases are available to the Grab option (except the Roundtable repository).</p>

Menu Item	Description
	<p><b>O/S Level</b> - Shells out to the operating system.</p> <p><b>Run any Program</b> - Will run program Entered at prompt.</p> <p><b>Dictionary</b> - See Chapter 4, <i>Workspaces</i> [4–1].</p> <p><b>Informal Xrefs</b> - See Chapter 4, <i>Workspaces</i> [4–1].</p> <p><b>Source Compare Report</b> - See Chapter 6.</p> <p><b>Print Source</b> - Prints source code for selected object.</p> <p><b>Versions in Product Module Report</b> - See Chapter 6.</p> <p><b>History versions</b> - Browses the object's update history.</p> <p><b>Build</b> - See Chapter 4, <i>Workspaces</i> [4–1].</p> <p><b>Copy</b> - See Chapter 4, <i>Workspaces</i> [4–1].</p> <p><b>Modify using sed</b> - See Chapter 4, <i>Workspaces</i> [4–1].</p> <p><b>Test Version Integrity</b> - Checks if source in the workspace is different from that registered in repository.</p> <p><b>Update Public/Groups</b> - Updates the source in the workspace directory for the current task's objects that have a share status of Group or Public.</p>
goapp	Runs the program specified in the current workspace module's Lead In Program.
Quit	Returns to the Workspace Modules Menu.

## 6.4. PCODE Objects

You can use PCODE objects to store almost any kind of data in the Roundtable repository by defining custom code subtypes that tell Roundtable how to process the data when it is stored into and retrieved from the repository.

PCODE objects are maintained in the PROGRESS Code Object Menu. You can run the procedure editor from Roundtable to edit a PCODE object.

PCODE objects store program or include files and, if necessary, other text files such as shell scripts. PCODE objects can also store binary data like bitmaps. The PCODE

definition records the:

- Compile time parameters of programs
- Deployment options
- Documentation Entered by the programmer about the purpose of the object

You must assign a subtype to each PCODE you create. The subtype defines the type of source the PCODE object represents. Subtypes allow you to track up to nine related text files as a single object. For more information on subtypes, see Section 3.9, “Subtypes” [3–21].

### 6.4.1. PROGRESS Code Object Menu Description

The PROGRESS Code Object Menu contains information about how Roundtable processes the object in the workspace configuration.

The PROGRESS Code Object Screen contains the following fields:

Field	Description
Sub Type	Defines the type of source represented by the PCODE object. Note that when a subtype is specified as not being a "program," the Runnable, Compiles, Save, Attr Space, and Compile Parameters fields cannot be edited.
Desc	A description of what the PCODE object does in the system.
Task#	No changes can be made to objects in a workspace unless the programmer selects a Task under which to work. The task# is assigned to the object version when the version is created. Many object versions can be created under the same task number.
Event	The Workspace Event History number for the latest action taken on the selected Object and the first Release in which the event appeared. If the object has a WIP status then the event field contains the message, "Work in Process."
Pmodule	Product module codes are assigned to an object when it is first created.
Version	Version code of the PCODE object. A version code has three, two-digit parts: level, revision, and patch. Use the ### keys to scroll through each version of the PCODE object that is assigned to the workspace.
Status	The status of the object version is either WIP (Work-In-Process) or Complete. Changes can only be made to objects that have a WIP status. A WIP status is assigned to object versions when they are created from the Version Menu. The status field also displays

Field	Description
	the status of the source associated with the object as CURRENT, ARCHIVED, NEW, or MODIFIED. In addition, the status field also shows the current share status value. The share status value can be Task, Group, Public, or Central.
Encrypt	Designates which type of remote sites receive encrypted code. The type of remote site is determined when creating deployments. See Section 4.11, "Deployments" [4–34]. Each character in the field represents a type of remote site. The default, "RQD", stands for Runtime, Query, and Development Progress products. This value indicates that any site running these products would receive encrypted source. A value of "S" in this field indicates that the object should be encrypted even if the site is designated to receive Source. Two special types are included: - B indicates a Binary file - ! indicates that the object should be excluded from deployments The (!) value is normally used with DOC objects and is potentially dangerous on PCODE objects. A value of "B" in this field indicates that the file is a binary file.
Runnable	Yes indicates that the PCODE object is a procedure that you can run directly from Roundtable. A procedure that expects parameters to be passed cannot be run from Roundtable.
Attr Space	Yes in this field indicates that the object version should be compiled with ATTRSPACE.
Compiles?	Type Yes to indicate whether this procedure compiles.
Compile Parameters	Additional PROGRESS COMPILE syntax should be used when compiling this object. To allow you to update this field, Roundtable presents a dialog box after you choose the Notes menu item.
Save	Yes indicates that the compile process should produce .r code. However, if the module specification for the Rcode value is No then no .r code is produced. Remember that the Rcode value in the module might be the ? value which indicates that the Rcode default value for the workspace is used. If the value of the Save: field is No then .r code is not produced in any circumstance.
Override	Sets the Override field to Yes if the Compile Parameters for this object are to be used for compiling the object instead of the compile parameters for the object's workspace and workspace module. Otherwise, the object's compile parameters are used in addition to the compile parameters for the workspace and workspace module.



The PROGRESS Code Object Menu contains the following menu items:

Menu Item	Description
Edit	Edits the parts of the PCODE object. Depending on the subtype assigned to the object, one or more test files can be included under a single PCODE object. If only one text file is specified by the subtype, this option takes you directly into your editor on that file. If more than one text file is involved, this option takes you to a source edit menu screen.  Note that if you use the PROGRESS procedure Editor, the <b>Select All</b> button Parts option is available. This option allows calls to the Procedure editor with each part loaded into a separate buffer.
Find	Jumps to another object in the system.
Compile	Compiles the program.
Run	Runs the program. If the PCODE object selected is not run directly from Roundtable, this option looks for the first PCODE object in the same object group that can be run. If none of the PCODE objects in the group are runnable, consider using <b>GOapp</b> .
Xref	Goes to the Xrefs Menu.
Where	Goes to Where Used Menu.
More	Additional utilities menu.
Notes	Edits compile options and programmer documentation.
Ver	Goes to the Version Control Menu.
Print	Prints text associated with object.
<b>GOapp</b>	Runs the workspace application from module entry point. (See the Workspace Modules Menu for instruction on how to set module entry points).
Quit	Returns to the previous menu.

### 6.4.2. Adding a PCODE Object

You must select a task before adding a PCODE object.

Follow these steps to add a new PCODE object:

1. From the Main Menu, select Workspaces. The Workspaces Menu appears.
2. Select the Workspace ID and choose Select. The Workspace Module Menu appears.

3. Select the Module and choose Select. The Workspace Objects Menu appears.
4. Choose Add. The Add or Assign Object dialog box appears.
5. Type New in the New/Assign field and PCODE in the Type field and press the **GO** key. The Adding New PCODE Object dialog box appears.
6. Enter the subtype in the Sub Type field and enter PCODE in the Object field.



Press the GET key to view and select the PCODE object subtypes.

7. Press the **GO** key to complete your edits. Roundtable inserts a row for the new PCODE object, and returns you to the Workspace Objects Menu.

### 6.4.3. Object Name Aliasing

Within a workspace, two objects cannot have the same name, even if they are in different subdirectories. If you must have two objects with the same name in a workspace, use the object name aliasing feature.

Roundtable has reserved the @ character for the object name aliasing feature. Use the @ character in the first position in an object name. When you enter an object name using the @ character, Roundtable prompts you for filenames for each of the subtype part files associated with that object. For example, if you have a post.p program in both the ap and ar subdirectories, use object name aliasing to name the second object @post.p. Roundtable prompts you for the relative path and part (file) name. In this example, your path is "ar" and your part name is "post.p."

Generally, you should not use this feature unless your existing Progress application has duplicate filenames. Try to assign a unique name to each object in your application.

Object name aliasing is only available for PCODE object types. It cannot be used with schema objects or DOC objects.

### 6.4.4. Editing a PCODE Object's Configuration Fields

You can only edit an object's configuration information if you have it checked out under your current task.

Follow these steps to edit a PCODE Object:

1. From the Workspace Objects Menu, select a PCODE object.
2. Choose Select. The PROGRESS Code Object Menu appears.
3. Choose Notes.

4. Edit the fields and press the **GO** key. The Object Compile Parameters dialog box appears.
5. Enter the any additional PROGRESS Compile syntax that should be used when compiling this object and press the **GO** key.
6. Choose **Quit** to close the current screen.

### 6.4.5. Editing a PCODE Object

To edit a PCODE object, its status must be Work in Process and its task must be the currently selected task.

Follow these steps to Edit a PCODE Object:

1. From the Workspace Objects Menu, find the PCODE Object and choose Select. The PROGRESS Code Object Menu appears.
2. Choose Edit. The editor you defined in the RTBSETUP File is launched. (Refer to the RTBSETUP File Options regarding the Editor and View options.)



If neither Editor nor View is defined, the PROGRESS Procedure Editor appears.

- Select a part to edit by using the up and down arrow keys, and press **ENTER**. Alternatively, you can quickly edit a part by pressing the key that corresponds to the part name. For example, to edit part "P," press the P key.

The following functions are also available from the Source View Menu:

Key	Description
<b>F3 (Ctrl-T)</b>	This function is only available if you are using the PROGRESS Procedure Editor as your editor. If you defined the EDITOR and VIEW options in your RTBSETUP file, then this function is not available. Selects all parts and loads them into the PROGRESS Procedure Editor. This allows you to edit several files simultaneously.
<b>F5 (Ctrl-G)</b>	Finds Object. Jumps to another object.

Key	Description
<b>F6 (Ctrl-P)</b>	Goes to the Xrefs Menu screen.
<b>F7 (Ctrl-R)</b>	Goes to the Where Used Menu.
<b>F8 (Ctrl-Z)</b>	Compiles without Xref.
<b>F9 (Ctrl-N)</b>	Compiles with Xref.
<b>F10 (Ctrl-D)</b>	Runs Program.
<b>F11 (Ctrl-B)</b>	Goes to the <b>GO</b> app option.
<b>F12 (Ctrl-A)</b>	Goes to the More Menu.

Your application can re-map these function keys without confusing Roundtable because Roundtable restores the mapping it requires whenever it is run or returned to after running your application.

## 6.5. DOC Objects

A DOC object is a simple text or binary file object managed by Roundtable. DOC objects are always stored in a subdirectory called DOC off of the subdirectory associated with the workspace module the object is assigned to. If any of these restrictions is a problem, create a subtype for PCODE objects that behave exactly the way you want them to.

The editor used for a DOC object is determined by the DOCEDIT and DOCVIEW options in the RTBSETUP file. If no DOCEDIT or DOCVIEW is defined, then the PROGRESS Procedure Editor is used.

The following table provides the DOC objects fields and descriptions:

Field	Description
Desc	Short, one-line description of object.
Task	No changes can be made to objects in a workspace unless the programmer selects a Task under which to work. The task# is assigned to the object version when the version is created. Many object versions can be created under the same task number.
Event	The Workspace Event History number for the latest action taken

Field	Description
	on the selected Object and the first Release in which the event appeared. If the object has a WIP status then the event field contains the message, "Work in Process."
Pmodule	Product module codes are assigned to an object when it is first created.
Version	Version code of the PCODE object. A version code has three, two-digit parts: level, revision, and patch. Use the ### keys to scroll through each version of the PCODE object that is assigned to the workspace.
Status	The status of the object version is either WIP (Work-In-Process) or Complete. Changes can only be made to objects that have a WIP status. A WIP status is assigned to object versions when they are created from the Version Menu. The status field also displays the status of the source associated with the object as CURRENT, ARCHIVED, NEW, or MODIFIED. In addition, the status field also shows the current share status value. The share status value can be Task, Group, Public, or Central.
Encrypt	Each character in this field indicates a remote site type for which this object version should be encrypted. For instance, if a remote site is defined as a "Query" site, then a Q in this field indicates that the object version should be encrypted when deployed to that site. A value of ! in this field indicates that the object should never be deployed to a remote site. A value of B in this field means that this file is a binary file.

### 6.5.1. Adding a DOC Object

You must have a WIP task selected to add a DOC object.

Follow these steps to add a DOC object:

1. From the Workspace Objects Menu, choose Add. The Add or Assign dialog box appears.
2. Type New in the New/Assign field and DOC in the Type field and press the **GO** key. The Adding New DOC Object dialog box appears.
3. Enter the subtype in the Subtype field and enter DOC in the Object field. Press the **GO** key. Roundtable adds a row for the new DOC object and returns you to the Workspace Objects Menu.
4. Choose Select. The Document Object Menu appears.

5. Choose Edit to edit the document.

OR

Choose Notes to change the version notes for the document.

6. Choose **Quit** to close the current screen.

## 6.6. Schema Objects

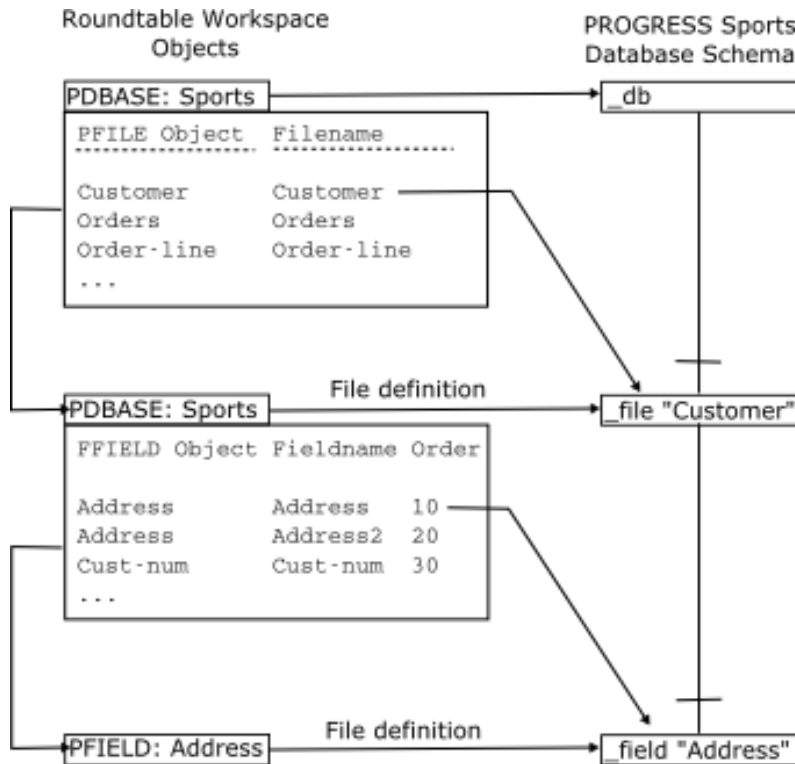
Roundtable provides a complete schema management system that includes full version control. In addition, field and file domains have been implemented to increase the consistency of your overall database schema and reduce the amount of maintenance required for some types of database schema changes.

Roundtable provides version control on the database schema by maintaining the database schema information in three object types:

- PDBASE: Database information and file assignments
- PFILE: Table definition and field assignments
- PFIELD: Field definitions

PFILE and PFIELD objects are reusable definitions, often called domains. A database is defined by a PDBASE object to which one or more PFILE objects are assigned. Each of these PFILE object assignments defines a table in the database. Each PFILE object contains information about the table and one or more PFIELD assignments. Each PFIELD object assignment defines a field in the table.

Refer to the following diagram that shows how Roundtable objects are translated into the \_(schema) tables of the Progress database:



Roundtable manages the Progress database schema by taking the database, table, and field definitions in the configuration specified by a workspace and updating the Progress schema tables in the application database associated with the workspace so that it conforms to the schema definition contained in the Roundtable objects. Roundtable allows any number of databases to be managed in any number of workspaces up to the limit on database connects allowed by Progress itself.

### 6.6.1. Domains

Both table and field domains are supported by Roundtable.

In a domain is you create a single definition that is used in a number of different places. For example, you can define a single PFIELD object called Address and use this field definition in a number of different tables, or even many times in the same table. Each use of the field definition is a field assignment. This assignment establishes a field in the table with name and order specified in the assignment. This can be advantageous when you need to change some attribute of the field definition. You can make the change in a single place, and Roundtable makes sure that all of the tables that use the field definition are

updated properly. This ensures consistency in your application and reduces the time spent on database maintenance.

Table domains are similar to field domains except that you are assigning a table definition (PFILE object) to two or more databases. Any change to the table definition is updated in each database by Roundtable. There is a limitation in the current release of Roundtable that precludes the assignment of a PFILE to two or more tables in the same PDBASE. It is possible to assign the same PFILE to two or more tables if these tables belong to different PDBASE objects.

## 6.6.2. Schema Object Versions

Schema objects are versioned like other objects in the Roundtable system. However, some differences do exist. Schema objects, PDBASE, PFILE, and PFIELD, are not stored as incremental versions. A complete copy of the data structure defining the schema object is stored in the repository for each version of the object. This does not lead to excessive growth in the database because creating a new object version of a PFIELD does not necessarily require new object versions of the PFILES to which it may be assigned.

In addition, Roundtable maintains internal buffers for checked out schema objects to track the last updated content of the schema objects. In this way, it can track the changes you make to schema objects while they are WIP. You do not have to check in (complete) your schema objects with each schema update process. Instead, you can make changes and update the schema into your application database in an iterative manner. You only check in the schema objects when you have finalized the database schema content.

It is usually a good idea to make your schema changes in a separate task from that used for creating and modifying programs. It is necessary to complete all schema objects before a release is created in a workspace.

## 6.7. PDBASE Objects

PDBASE (database) objects record a list of PFILE assignments that link a PFILE with the PDBASE object. When a PDBASE object is assigned to a workspace, additional information is required to allow connection of the physical database. This connection information is part of the assignment of the object to the workspace. It is not part of the PDBASE object. It is possible to change this database connection information without checking out the PDBASE object.

There are two things to be aware of when creating a PDBASE object:

- The name of a PDBASE object is important, because Roundtable uses the PDBASE object name as the logical database name. Roundtable requires this for cross-referencing purposes.
- Unless you specify -U and -P connection parameters for the database, Roundtable uses the same user ID and password for the workspace database as you used for logging into



Roundtable. To avoid this, use -U "" -P "" as part of your workspace database's connection parameters.

### 6.7.1. PROGRESS Database Object Menu

The PDBASE object screen contains version information about the PDBASE object as well as connection parameters to the physical PROGRESS database.

The PROGRESS Database Object Menu screen contains the following fields:

Field	Description
Task#	No changes can be made to objects in a workspace unless the programmer has selected a Task to work under. The task# is assigned to the object version when the version is created. Many object versions can be created under the same task number.
Event	The Workspace Event History number for the latest action taken on the selected Object and the first Release in which the event appeared. If the object has a WIP status then the event field contains the message, "Work in Process."
Pmodule	Product module codes are assigned to an object when it is first created.
Version	Version code of the PDBASE object. A version code has three, two-digit parts: level, revision, and patch. Use the ARROW keys to scroll through each version of the PDBASE object that is assigned to the workspace.
Status	The status of the object version can be Work in Process (WIP) or complete. Changes can only be made to objects that have a WIP status. A WIP status is assigned to object versions when they are created from the Version Menu. The status field also shows the status of the source associated with the object as CURRENT, ARCHIVED, NEW, or MODIFIED. In addition, the status field also shows the current share status value. The share status value is Task, Group, Public, or Central.
DataServer	Indicates whether or not the database is a DataServer schema holder for a non-Progress database.
Db Type	The type of database (PROGRESS, ORACLE, ODBC, etc.).
Physical Name	Physical name of database. This field can contain a relative path to the database from the root directory or a fully qualified path starting with a (/) character if the database is not located in the workspace directory structure. Note that if the physical name field is left blank, Roundtable does not attempt to connect the database when the workspace is connected.

Field	Description
Connect Order	A field that controls the order in which Roundtable connects to multiple databases.
Connection Parameters	A set of fields for the Progress connection parameters. When connecting to an application database, Roundtable supplies the user ID and password you used to log into Roundtable. To avoid this, you specify a user ID and password using the -U and -P options. It is possible to pass a blank user ID and password with a connection parameter specification of -U " " -P " ".

The PROGRESS Database Object Menu contains the following options:

Menu Item	Description
Select	Goes to Database File Assignments Menu.
Find	Jumps to another object in the workspace.
Aliases	Displays a table where you can <b>ENTER</b> one or more database aliases that are created when the database is connected. Note that all databases defined in the workspace are connected first and then aliases are created. If a database is already connected with a logical name that conflicts with an alias, the alias is ignored.
#seq	Displays the Database Sequences Assignment Menu.
Edit	Edits workspace-specific parameters and notes. The notes section can only be edited if the object has a WIP status. Workspace-specific parameters are not subject to version control and can be edited at any time.
Ver	Goes to Versions Menu.
Load	Loads PROGRESS Schema from connected database. This option allows the user to Load the schema of an existing database into Roundtable. PFILE and PFIELD objects are created for each of the file and field definitions found in the PROGRESS database.
Preview	Previews unapplied changes to the object.
Reports	Provides access to the PDBASE Object, Database Definition, and the Versions in Product Module Report, which are described in the Reports Review section. The PDBASE Reports Menu also provides access to the Integrity Check Report, which compares Roundtable's schema definition with the schema definition in the Physical PROGRESS Database. This report is very useful for finding out if someone has modified the PROGRESS database schema manually.
Where	Goes to the Where Used Menu.

Menu Item	Description
History	Views update history of object versions.
Quit	Returns to previous menu.

### 6.7.2. Adding a PDBASE Object

Before you add a PDBASE object, the database must exist in the system and be available for connection. A connect is attempted during the PDBASE object add process to verify the database connection parameters you enter. You must also have a WIP task selected.

Follow these steps to Add a PDBASE object:

1. From the Workspace Objects Menu, choose Add. The Add or Assign dialog box appears.
2. Type New in the New/Assign field and PDBASE in the Type field and press the **GO** key. The Adding New PDBASE Object dialog box appears.
3. Enter the product module, object group, and level for the new PDBASE object and press the **GO** key.
4. Enter the name of the new PDBASE object and press the **GO** key. The object name that you enter for the new PDBASE object will be used by Roundtable when it connects to it as the logical database name.



Do not Enter the .db database extension in the object name field.

Roundtable displays the following dialog box with the object name that you Entered:

1. Choose Yes if the object name that you Entered is the correct logical name for the database. The Workspace Objects Menu appears.
2. Choose Select. The PROGRESS Database Objects Menu appears.
3. Choose Edit to edit the version notes and connection parameters for the new PDBASE object.
4. Choose **Quit** to return to the previous screen.

### 6.7.3. Editing the PROGRESS Database Object

Follow these steps to edit the PDBASE Object:

1. From the Workspace Objects Menu, select the PDBASE Object.
2. Choose Select. The PROGRESS Database Object Menu appears.
3. Choose Edit and edit the fields.
4. Press the **GO** key to complete your edit.

#### 6.7.4. PDBASE for DataServer Database

In order to utilize DataServer schemas with Roundtable, PDBASE objects must be defined in specific ways. First, there must be a PDBASE object defined for each database with its schema managed by Roundtable. This is the case, even if one physical Progress database contains schemas for more than one database (such as both its own native schema and the schema of foreign database).

Second, the name of the PDBASE object must be identical to the name of the database used to manage the schema. If the PDBASE object corresponds to a foreign database, the name of the PDBASE object must be the same as the name of the foreign database as it was when it was connected to populate the DataServer schema holder database (stored in the `_Db-name` field of the corresponding `_Db` record). The physical database specified for these objects, however, will be the name of the Progress database that serves as the schema holder for the foreign database. If the PDBASE corresponds to a native Progress schema, the PDBASE name will serve as the logical name for that database (unless an `-ld` value is supplied in the connection parameters for that PDBASE object).

Consider the following sample PDBASE object definitions:



PDBASE definition for ORACLE DataServer schema holder and database. Notice that the physical name specified is for the Progress schema holder. A connection to the ORACLE database is made using the additional connection parameters supplied in the definition (See the Progress Oracle DataServer Guide for appropriate connection parameters).

Property	Value	Description
Object Name	ORCL	Name of ORACLE database.
Physical Name	C:/rtb/project/devel/orahold.db	Pathname of Progress schema holder.
Connection Parameters	-ld orahold  -RO  -db ORCL -dt ORACLE	Logical name for schema holder.  Connection parameters for ORACLE database.

Property	Value	Description
	-U user_1@ORACLE.SERVICE.NAME  -P passwd	



PDBASE definition for ODBC DataServer schema holder and database. Notice that the physical name specified is for the Progress schema holder. A connection to the ODBC database is made using the additional connection parameters supplied in the definition (See the Progress ODBC DataServer Guide for appropriate connection parameters, and DSN setup).

Property	Value	Description
Object Name	MS-Demo	Name of ODBC database.
Physical Name	C:/rtb/demo/devel/demohold.db	Pathname of Progress schema holder.
Connection Parameters	-ld demohold  -RO  -db MS-Demo -dt ODBC	Logical name for schema holder.  Connection parameters for ODBC database.

PDBASE definitions for Progress a database used for both native schema and as a DataServer schema holder for an ORACLE database. The first PDBASE definition is for the native schema, and the second for the ORACLE database. Note the read-only (-RO) parameter in the second definition. Without this, you will get an error message when Roundtable attempts a second connection to a single database.

Property	Value	Description
Object Name	Prodemo	Name of Progress database.
Physical Name	/usr/rtb/demo/devel/prodemo.db	Pathname of Progress database.
Connection Parameters		Only as required per installation.
Property	Value	Description
Object Name	Orademo	Name of ORACLE database.
Physical Name	/usr.rtb/demo/devel/prodemo.db	Same pathname as before.
Connection		

Property	Value	Description
Parameters	-RO  -db orademo -dt ORACLE  -U user_1@ORACLE.SERVICE.NAME  -P passwd	Required.  Connection parameters for ORACLE database.

### 6.7.5. DataServer Schema Change Restrictions

Since Progress DataServer schema holders are designed to be only recipients of foreign schema (using utilities provided in the Progress Data Administration tool), Roundtable also limits one-way update of DataServer schema. This allows you to load in the schema changes from a Progress schema holder for impact analysis and selective compiling. The Load Schema utility finds any change to the physical schema holder and reads it into Roundtable (versioning objects if necessary).

Schema for a DataServer database can only be added and/or updated in Roundtable using Roundtable's Load Schema feature (see Section 7.6, "Load Schema" [7–12]). Schema assigned to a PDBASE object that corresponds to a DataServer database cannot be manually checked out, nor can a user directly modify the properties of the objects.

As with loading native Progress schema using the Load Schema feature, schema objects for DataServer schema will be checked out automatically as required. Even though schema objects assigned to a PDBASE object corresponding to a DataServer database remain WIP until they are checked in, users will not be able to change any of the properties (local table name, data type, etc.) for these objects.



Roundtable does not support the use of a DataServer for the Roundtable repository.

### 6.7.6. Adding a Database Alias

The database aliases defined for the database are created for you when Roundtable connects to the database. Otherwise, code that expects these aliases to be available cannot compile. Roundtable does not provide automatic alias support on deployment. Instead, your deployed application connects to the appropriate databases and creates the required aliases.

Follow these steps to add an alias:

1. From the PROGRESS Database Object Menu, select the PDBASE object to which you want to add an alias.
2. Choose Select. The PROGRESS Database Object Menu appears.
3. Choose Aliases. The Database Aliases screen appears.
4. Choose Add.
5. Type the Alias Name in the Alias field.
6. Press the **GO** key to complete your entry.

### 6.7.7. Deleting an Alias

Follow these steps to delete an alias:

1. From the Database Aliases Menu screen, select the alias name you want to delete.
2. Choose Delete. The Question, "Delete this record" appears.
3. Type Yes to delete.
4. Press the **GO** key to delete the alias.

### 6.7.8. Adding a Database Sequence

Follow these steps to add a database sequence:

1. From PROGRESS Database Object Menu, select the PDBASE Object.
2. Choose #seq. The Database Sequences Assignment screen appears.
3. Enter these values for the new sequence.
  - **Name:** Enter the name.
  - **Cycle:** Check here to have Progress start over assigning numbers once the maximum number is reached.
  - **Increment:** The amount by which to increment each consecutive number.
  - **Initial:** The first number to use when assigning numbers.
  - **Maximum:** The highest number to assign before stopping or starting over.
  - **Minimum:** The lowest number that Progress should use when assigning numbers.
4. Press go to complete your new sequence entry.
5. Choose **Quit** to return to the previous menu.

### 6.7.9. Edit a Database Sequence

Follow these steps to delete a database sequence:

1. From the Workspace Objects Menu, select the PDBASE Object and choose select. The PROGRESS Database Object Menu appears.
2. Choose #seq. The Database Sequences Assignment screen appears.
3. Choose Edit. Edit the Sequence.
4. Press go to complete the edit.

### 6.7.10. Delete a Database Sequence

Follow these steps to delete a database sequence:

1. From the Workspace Objects Menu, select the PDBASE Object and choose select. The PROGRESS Database Object Menu appears.
2. Choose #seq. The Database Sequences Assignment screen appears.
3. Choose Delete. The question "Delete this record" appears.
4. Type Yes and press the **GO** key.
5. Choose **Quit** to return to the previous menu.

### 6.7.11. Load PROGRESS Schema

The Load Progress Schema utility loads the schema of an existing database into Roundtable. The utility creates PFILE and PFIELD objects for each of the file and field definitions found in the Progress database. It can load changes made to the Progress database schema, so that the logical schema object definitions in Roundtable are brought in sync with the physical Progress database schema.

See Section 7.6, "Load Schema" [7–12] for a complete description of this utility.

### 6.7.12. Managing Database Tables

Databases consist of one or more tables defined by PFILE objects. You can add tables to your database by assigning existing PFILE objects or by creating a PFILE object while in PDBASE expand mode. You can delete tables from a database by removing the PFILE assignment. When you delete a table from the database, the PFILE object remains. The PFILE object still exists in the workspace and the repository.

### 6.7.13. Adding a Table



You can add a PFILE object without immediately assigning it to a PDBASE object. However, you usually add tables to your database by creating a new PFILE object while in PDBASE expand mode. When you add a new PFILE object in this mode, it is assumed you want to assign it to the PDBASE, so the assignment is created automatically. The PDBASE object must be WIP under your current task to add a table to it.

Follow these steps to add a new table:

1. From the Workspace Objects Menu, find the PDBASE object and choose Select. The PROGRESS Database Object Menu appears.
2. Choose Select. The Database File Assignments Menu appears.
3. Type the new PFILE object name. The message, "PFILE Object does not exist in workspace. Add a new object?" appears.
4. Choose Yes. The Adding New PFILE Object dialog box appears.
5. Complete the fields and press the **GO** key.
6. Enter the local filename and press the **GO** key. The local file name (physical name of the table in the database) is normally the same as the name of the PFILE object.
7. Choose **Quit** to return to the previous menu.

### 6.7.14. Table Assignment

The PDBASE object must be WIP under your current task to add a table assignment to it.

Follow these steps to add tables to your database by assigning an existing PFILE object to the database:

1. From the Database File Assignments Menu, choose Add.
2. Enter the name of an existing PFILE Object.
3. Enter the local filename and press the **GO** key. The local file name (physical name of the table in the database) is normally the same as the name of the PFILE object.

### 6.7.15. Deleting a Table Assignment

The PDBASE object must be WIP under your current task to delete a table assignment from it. You can delete a table from your database by removing the PFILE assignment that defines the table. Deleting the PFILE assignment does not delete the PFILE object.

Follow these steps to delete a table:

1. From the Database File Assignments Menu, select the file object.
2. Choose Delete. The message, "Delete this record" appears.
3. Type Yes and press the **GO** key.
4. Choose **Quit** to return to the previous screen.

### 6.7.16. PDBASE Object Report

The PDBASE Object Report reports the definition of a PDBASE object. It does not report on the definitions of the PFILE assigned to the PDBASE. If you want a complete schema definition, use the Database Definition Report instead. See Section 6.7.18, "Database Definition Report" [6–24]. Follow these steps to print the report:

1. From the PROGRESS Database Object Menu, choose Reports. The More Menu appears.
2. Select the PDBASE Object Report.
3. Complete the print fields and press **GO**.

### 6.7.17. Changing Local Table Name

The local table name is the name of the table seen in the PROGRESS database. When you assign a PFILE object to a PDBASE object to create a table, you specify the local table name. The name of the PFILE object is supplied by default but does not have to be the same as the local table name. The PDBASE object must be WIP under your current task to change a local table name.

To change the local table name of a PFILE assigned to a PDBASE follow these steps:

1. From the Database File Assignments Menu, choose Edit.
2. Edit the Filename field. This is the name that is used for the table in the physical PROGRESS database. It can be different from the PFILE object name.
3. Press the **GO** key to complete the edit.

### 6.7.18. Database Definition Report

The Database Definition Report provides the full definition of the database from the contents of the Roundtable schema.

Follow these steps to print the report:

1. From the PROGRESS Database Object Menu, choose Reports. The More Menu appears.
2. Select the Database Definition Report. The Database Definition Report dialog box appears.
3. Complete the report fields and press the **GO** key. The Print Destination dialog box appears.
4. Complete the print fields and press the **GO** key.

### 6.7.19. Integrity Check Report

The Integrity Check Report compares the Roundtable schema definition with the schema definition in the physical Progress database. This report is useful for finding out if someone has modified the Progress database schema manually.

Follow these steps to print the report:

1. From the PROGRESS Database Object Menu, choose Reports. The More Menu appears.
2. Select the Integrity Check Report.
3. Complete the print fields and press the **GO** key.

### 6.7.20. Index Usage Report

The Index Usage Report shows where the selected indices are used in the application, including an unused index summary. To run the Index Usage Report:

1. From the Progress Database Object Menu, choose Rpts, and then choose Index Usage Report. The Index Usage Report dialog box appears.
2. Select the tables and/or indices for the report. If you select multiple tables, all indices in the tables will appear on the report. Selecting the Unused index summary toggle box shows all unused indices in the system.
3. Choose OK after making your choices.

## 6.8. PFILE Objects

A PFILE object defines the contents of a Progress table. It contains:

- Schema information related to the table as a whole
- Index and table header definitions for the table

- A list of PFIELD assignments that include local field names, field order, and mandatory status

Define the fields of a table through assignment of existing PFIELD objects or by creating new PFIELD objects while in the Workspace Objects Menu. In addition to the contents of the PFIELD definition, table fields are further defined by a local field name, a field order, and mandatory status values. The table field name and the PFIELD object name are usually (but do not have to be) the same.

Roundtable shows the content of a PFILE object definition where you perform the following activities to manage a PFILE object:

- Assign PFIELD objects to the PFILE
- Define indexes and index fields for the PFILE
- Define file triggers

The PFILE reports available are:

- PFILE Object Report, which provides the definition of a PFILE object
- File Definition Report, which provides the full PFILE definition

### 6.8.1. PROGRESS File Object Menu

The PROGRESS File Object screen contains a PFILE object definition.

The File Object screen contains the following fields:

Field	Description
Task#	No changes can be made to objects in a workspace unless the programmer has selected a Task to work under. The task# is assigned to the object version when the version is created. Many object versions can be created under the same task number.
Event	The Workspace Event History number for the latest action taken on the selected Object and the first Release in which the event appeared. If the object has a WIP status then the event field contains the message, "Work in Process."
Pmodule	Product module codes are assigned to an object when it is first created.
Version	Version code of the PFILE object. A version code has three, two-digit parts: level, revision, and patch. Use the ARROW keys to scroll through each version of the PFILE object that is assigned to the workspace.

Field	Description
Status	The status of the object version can be Work in Process (WIP) or complete. Changes can only be made to objects that have a WIP status. A WIP status is assigned to object versions when they are created from the Version Menu. The status field also shows the status of the source associated with the object as CURRENT, ARCHIVED, NEW, or MODIFIED. In addition, the status field also shows the current share status value. The share status value may be on of Task, Group, Public, or Central.
Hidden	Progress table _hidden attribute.
Frozen	Progress File _frozen attribute. Roundtable can update files that have the Frozen attribute set. Use this attribute to prevent others from making changes to the file using the Progress Data Dictionary. Roundtable is not affected by the _frozen flag attributes when it updates the schema.
File-label	Table label and its string attribute.
Dump name	A fill-in for the dump name of the table.
Description	Description of the table's use in the system.
ValExp	Record deletion validation expression.
Msg	Validation message and its string attribute.

### 6.8.2. Adding a PFILE Object

You must have a task selected to add a PFILE object.

Follow these steps to add a PFILE object:

1. From the Workspace Objects Menu, choose Add. The Add or Assign dialog box appears.
2. Type New in the New/Assign field and PFILE in the Type field and press the **GO** key. The Adding New PFILE Object dialog box appears.
3. Complete the PMODULE, Object Group, and Level fields and press the **GO** key.
4. Type the name of the PFILE object and press the **GO** key.

### 6.8.3. Editing the PFILE Object

The PFILE object must be WIP under your current task to edit it.

Follow these steps to edit the PFILE Object:

1. From the Workspace Objects Menu, select the PFILE Object and choose Select. The PROGRESS File Object Menu appears.
2. Choose Edit.
3. Edit the fields and press the **GO** key.

### 6.8.4. Field Assignments

A table is comprised of one or more fields. Define these fields by assigning PFIELD objects to the PFILE object that defines the table. If the PFILE object has a work-in-process (WIP) status, you can create, edit, or delete PFIELD object assignments to change the field definitions of the table.

The following table lists the Field Assignment screen fields and descriptions:

Field	Description
Name	Contains either the local field name of the PFIELD object assigned to the PFILE object, the side label, or the column label of the field. The default is "fieldname" on Entering the screen. All three values are always shown for the current row in the table, with two of these values in the detail area below the table.
Ord	The Field ordering value can be edited in this screen.
Mand	The Mandatory field flag can be edited in this screen.
Data-type	The data type field is specified in the PFIELD object definition.
Ext	The extent of the field is specified in the PFIELD object definition.
Dec	The number of decimal positions is specified in the PFIELD object definition.
Case	The case sensitivity is specified in the PFIELD object definition.
Idx	Indicates whether the field is used in an index in the PFILE object.
Object	Name of PFIELD object.
ID	When fields are created in a PFILE, each is given an assignment number (ID). If the field-name is changed, Roundtable uses the ID of the field, which does not change, to compare the old field definition to the new definition. You can change the PFIELD assigned to a field definition without the ID changing. This is the basis for data preservation on field extent and data type changes.
Sort	Sort order for the display. The Order option on the Filed Assignment Menu allows the Fields of File screen to be sorted by one of: Fieldname, Order, Index, Reverseorder, Mandatory, or

Field	Description
	Data type.
Trigger	Contains the name of the procedure that contains the trigger to fire on assignment of data to the field. Note that this is the filename of the procedure and not an object name. This filename must contain a relative path from the workspace root directory. Do not use fully qualified path names here!

The Field Assignment Menu contains the following menu items:

Menu Item	Description
Select	Goes to the PROGRESS Field Object Menu.
Add	Adds a new field.
Edit	Edits the current field: PFIELD object, Order, Mandatory, Fieldname, and Trigger. Object must have WIP status. Certain edits are restricted if the field participates in an index.
Delete	Deletes the current field. Object must have WIP status. You can not delete field assignments that participate in an index.
Reorder	Reorders the field order numbers in increments of 10.
Order	Changes the display sort order to one of (Fieldname, Order, Index, Reverseorder, Mandatory, Data type).
Find	Finds the first field that begins with user entry.
Name	Toggles name displayed in table among (Field-name, Label, Col-label).
Index	Goes to the Index Menu.
Quit	Returns to the previous menu.

### 6.8.5. Adding a Field to a Table

You can add new fields to a table by creating new PFIELD objects while in the Field Assignments Menu screen. When you create a PFIELD object, a field assignment is created automatically. The PFILE object must have a work-in-process (WIP) status for new fields to be added.

Follow these steps to add a new field to a table:

1. From the Workspace Objects Menu, select the PFILE object that you want to add a field to that has a WIP status.

2. Choose Select. The PROGRESS File Object Menu appears.
3. Choose Select. The Field Assignment Menu appears.
4. Choose Add. The Assign PFIELD Object dialog box appears.
5. Enter a new PFIELD object name and press **GO**.
6. Type Yes in the New Object? field and press the **GO** key. The Adding New PFIELD Object dialog box appears.
7. Enter the product module, object group, and level, and press the **GO** key. The Field Parameters dialog box appears.
8. Enter the data type and extent and press the **GO** key. The Assign PFIELD Object dialog box reappears.
9. Complete the remaining fields and press the **GO** key.

### 6.8.6. Assigning a Field

You can add fields to the table by assigning an existing PFIELD definition. The PFILE object must have a work-in-process (WIP) status for new fields to be assigned.

Follow these steps to assign an existing PFIELD object:

1. From the Workspace Objects Menu, select the PFILE object that you want to assign a field to that has a WIP status.
2. Choose Select. The PROGRESS File Object Menu appears.
3. Choose Select. The Field Assignment Menu appears.
4. Choose Add. The Assign PFIELD Object dialog box appears.
5. Enter an existing PFIELD object name and press the **GO** key.
6. Complete the remaining fields and press the **GO** key.

### 6.8.7. Editing a Field Assignment

The PFILE object must be WIP under your current task to edit field assignments.

Follow these steps to edit a PFIELD assignment:

1. From the Field Assignment Menu, select the field assignment you want to edit.
2. Choose Edit. The Assign PFIELD Object dialog box appears.
3. Edit the PFIELD object field if necessary, and press the **GO** key. Only change this field if you want to change which PFIELD object is being used for the field



assignment.

### 6.8.8. Deleting a Field Assignment

When you delete a PFIELD assignment from a PFILE object, you are not deleting the PFIELD object from the workspace, only its assignment to the PFILE. The PFILE object must be WIP under your current task to delete field assignments.

Follow these steps to delete a PFIELD assignment:

1. From the Workspace Objects Menu, select the PFILE object that you want to delete the field assignment from that has a WIP status.
2. Choose Delete. A message verifying the delete appears.
3. Type Yes and press the **GO** key.
4. Complete the remaining fields and press the **GO** key.

### 6.8.9. Indexes

The Index Menu allows you to add and edit the indices of the PFILE object.

The Index Menu screen contains the following fields:

Field	Description
PFILE Object	The name of the PFILE object.
Description	Description in the PFILE object header.
Status	Status of PFILE object is either WIP or Complete.

Also see Section 4.5.7, “Deactivate Flags” [4–21].

### 6.8.10. Select Index Frame

This frame contains a list of the indices of the PFILE object. Use the ARROW keys to scroll through the indices. Note that as you select each index, the Index Details Frame and # Field-Name Frame contents change. Also, note that the primary index in the Select Index Frame is identified by an asterisk (\*).

The # Field-Name Frame contains the following fields:

Field	Description
#	Contains the sequence number of the index component.

Field	Description
+-	+ means ascending, - means descending
Field-name	Field name assigned to the index.
Type	Data type of field assigned to the index.

### 6.8.11. Index Menu

The following table lists the Index Menu options:

Field	Description
Add	Adds an Index.
Edit	Goes to the Index Components Menu.
Delete	Deletes the Index.
Rename	Renames the Index.
Make-Primary	Makes the selected index the primary index.
Fields	Goes to the Field Assignment Menu.
Where	Where Used (INDEX) Menu.
Quit	Returns to the PROGRESS File Objects Menu.

### 6.8.12. Index Components Menu

The Index Fields Frame contains the following information:

Field	Description
#	Index field order.
Field-name	Field name assigned to the index. When editing this field, press the GET key for a list of available field names.
Asc	Ascending Flag.
Abr	Abbreviate Flag.

The Index Components Menu includes these menu items:

Menu Item	Description
Add	Adds an index component.
Edit	Edits an index component.

Menu Item	Description
Change	Changes an index description.
Delete	Deletes an index component.
Abbreviate	Sets Abbreviate flag on last index component.
Word	Toggles word index flag. Note that a word index can only have a single field, which must be a character data type.
Quit	Returns to the Index Menu.

### 6.8.13. Adding an Index

The PFILE object must be WIP under your current task to add an index.

Follow these steps to add an index:

1. From the Workspace Objects Menu, select the PFILE where you want to add an index.
2. Choose Select. The PROGRESS File Object Menu appears.
3. Choose Index. The Index Menu appears.
4. Choose Add. The Add New Index dialog box appears.
5. Complete the fields and press the **GO** key. The Index Components Menu appears.
6. Choose Add to add components to your new index.

### 6.8.14. Deleting an Index

The PFILE object must be WIP under your current task to delete an index.

Follow these steps to delete an index:

1. From the Index Menu, select the Index you want to delete.
2. Choose Delete. The message, "Delete this record" appears.
3. Type Yes and press the **GO** key.

### 6.8.15. Renaming an Index

The PFILE object must be WIP under your current task to rename an index.

Follow these steps to rename an index:

1. From the Index Menu, select the Index you want to rename.
2. Choose Rename.
3. Type the new index name and press the **GO** key.

### 6.8.16. Editing an Index

The PFILE object must be WIP under your current task to edit an index.

Follow these steps to edit an index:

1. From the Index Menu, select the index you want to change.
2. Choose Make-primary if you want to make the current index the primary index for the table.
3. Choose Edit. The Index Components Menu appears.
4. Choose Add if you want to add a new index component.
5. Select the index component and choose Edit if you want to edit an existing index component.
6. Choose Change if you want to change the text description of the index.
7. Select the index component and choose Delete if you want to delete the index component.
8. To toggle whether or not an index is abbreviated, choose aBreviate. Roundtable toggles the Abbreviate flag between Yes and No. An index can be abbreviated only if its last component is a character field.
9. To toggle whether or not an index is a word index, choose Word. Roundtable toggles the Word Index flag between Yes and No. An index is a word index only if it contains a single, character field component.

### 6.8.17. Adding Table Triggers

The following screen allows you to maintain event triggers to be defined as part of the PFILE object. Note that Roundtable does not support CRCs on triggers because this would make deployment of schema information difficult or impossible in many situations. See the PROGRESS documentation for more information on CRC checks on triggers.

Follow these steps to add a table trigger:

1. From the Workspace Objects Menu, select the PFILE Object that you want to change. The PFILE Object must be Work in Process under your current task.

2. Choose Select. The PROGRESS File Object Menu appears.
3. Choose Trig. The File Triggers Table Menu appears.
4. Choose Add.
5. Complete the fields and Press **GO**.

### 6.8.18. Deleting a Trigger

Follow these steps to delete a table trigger:

1. From the Workspace Objects Menu, select the PFILE Object. The PFILE Object must be Work in Process under your current task.
2. Choose Select. The PROGRESS File Object Menu appears.
3. Choose Trig. The File Triggers Table Menu appears.
4. Select the event and choose Delete. The message "Delete this record" appears.
5. Type Yes to delete and press the **GO** key.
6. Choose **Quit** to return to the previous screen.

### 6.8.19. PFILE Object Report

The PFILE Object Report prints the definition of a selected PFILE object.

Follow these steps to print the report:

1. From the Workspace Objects Menu, select the PFILE Object.
2. Choose Select. The PROGRESS File Object menu appears.
3. Choose Report. The PFILE Reports Menu appears.
4. Select the PFILE Object Report.
5. Complete the print destination fields and press **GO**.

### 6.8.20. Table Definition Report

The Table Definition Report prints a full file definition from the contents of Roundtable schema.

Follow these steps to print the report:

1. From the Workspace Objects Menu, select the PFILE Object.

2. Choose Select. The PROGRESS File Object menu appears.
3. Choose Reports. The PFILE Reports Menu appears.
4. Select the File Definition Report. The File Definition Report dialog box appears.
5. Choose Provide Brief List to print a brief list of all the fields in the file or choose Provide Full Detail to print the full details of each field in the file.



The Brief List and Full Detail options are not mutually exclusive. The report produces both sections if both options are selected.

6. The print dialog box appears. Complete the report dialog box fields and press the **GO** key.
7. Complete the print destination fields and press the **GO** key.

## 6.9. PFIELD Objects

PFIELD definitions in Roundtable are similar to the field definitions found in the Progress Data Dictionary. However, unlike Progress field definitions, PFIELD objects can be used in one or more field assignments in PFILE table definitions.

For example, you can define a single PFIELD object called Address and assign this definition to one or more PFILE table fields. To change the length of the field, you change only the PFIELD definition. Roundtable updates the table's field definition wherever a field is assigned to a table.

### 6.9.1. Field Object Menu Description

The PROGRESS Field Object Menu contains the PFIELD definition. You can change most of the fields in this window, except the data type or extent of the field. To change the data type or extent of a field, see Section 6.9.4, "Changing the Data Type or Extent of a Field" [6–38]. You can change the decimal field only if the field has a decimal data type.

The PROGRESS Field Object Menu fields include:

Field	Description
Label	Default field label and string attribute.
Type	Data type: Character, Integer, Decimal, Date, Recid, Logical, or Raw.
Col-label	Default column label and string attribute.
Extent	Extent of field.
Format	Default format phrase of field and sting attribute.

<b>Field</b>	<b>Description</b>
Decimals	Decimal places if data type is decimal.
Initial	Initial value of field and string attribute.
View-as	View-as widget phrase.
Case Sensitive	Yes/No.
Val	Field validation expression.
Mesg	Validation message and string attribute.
Help	Field help message and string attribute.
Desc	Description of field and string attribute.

The PROGRESS Field Object Menu items include:

<b>Menu Item</b>	<b>Description</b>
Edit	Edits the field definition. Object must have WIP status.
Ver	Goes to the Version Control Menu.
Preview	Reviews changes to the field definition since its last schema update.
Reports	Provides field reports including the PFIELD object and Versions in Product Module Report.
Where	Goes to the Where Used Menu for the current PFIELD object.
History	Views the editing history of each version of the object.
Change	Allows you to change the data type and/or extent of a field. You are not allowed to change the extent of a field that is used in an index.
Quit	Returns to the previous menu.

### 6.9.2. Adding a PFIELD Object

You must have a task selected to add a PFIELD object.

Follow these steps to add a PFIELD object:

1. From the Workspace Objects Menu, choose Add. The Add or Assign Object dialog box appears.
2. Type New in the New/Assign field and PFIELD in the Type field and press the **GO** key. The Adding New PFIELD Object dialog box appears.

3. Enter the product module, object group, and level fields, and press the **GO** key.
4. Enter the new PFIELD object name and press the **GO** key. The Field Parameters dialog box appears.
5. Enter the data type and extent, and press the **GO** key. Roundtable adds a row for your new PFIELD object, and returns you to the Workspace Objects Menu.

### 6.9.3. Editing a PFIELD Object

The PFIELD object must be WIP under your current task to edit a PFIELD object.

Follow these steps to edit a PFIELD object:

1. From Workspace Objects Menu, select the PFIELD you want to edit.
2. Choose Select. The PROGRESS Field Object Menu appears.
3. Choose Edit.
4. Edit the current record and press the **GO** key.

### 6.9.4. Changing the Data Type or Extent of a Field

You can only change the data type or extent of a field using the Change Data type or Extent function on the PROGRESS Field Object Menu. The PFIELD must have a work-in-process (WIP) status to be modified.



Changing the data type or extent of a field might cause loss of data in your application database. Roundtable does provide a mechanism for transforming the data from one type to another during the schema update process, but you often have to write your own data conversion routines. See Section 4.5, “Database Schema Updates” [4–17].

If you have used a field in an index, you cannot change the extent of the field.

Follow these steps to change the data type or extent of a field:

1. From the Workspace Objects Menu, select a PFIELD object that has a WIP status.
2. Choose Select. The PROGRESS Field Object Menu appears.
3. Select Change. The Change data type and/or extent of field object dialog box appears.
4. Complete the fields and press the **GO** key.



### 6.9.5. PFIELD Object Report

The PFIELD Object Report prints the field definition information for a selected field.

Follow these steps to print the report:

1. From the PROGRESS Field Object Menu, select Reports. The PFIELD Reports Menu appears.
2. Select the PFIELD Object Report.

### 6.10. Assigning an Object

The assign process allows you to extract any completed version of any object from the Roundtable repository and assign it to the current workspace. It becomes available in the workspace module that is associated with the product module to which the object belongs. If the object already exists in the workspace, the current object version is overwritten by the new object version. The object's status is set to NEW to mark the object for compilation or update schema processing, if necessary.

Follow these steps to assign an object version to the current workspace:

1. From the Workspace Modules Menu, find the module that you want to assign the object to and choose Select. The Workspace Objects Menu appears.
2. Choose Add. The Add or Assign Object dialog box appears.
3. Type **Assign**, Enter the Object type, and press the **GO** key. The Assigning Object Type dialog box appears.
4. Complete the fields and press the **GO** key. Roundtable assigns the object to the workspace.

### 6.11. Object Reports

There are a number of reports you can access to provide information on the objects managed by Roundtable. The available reports include:

- **Versions in Product Module Report:** Shows the history of the development of object versions for the currently selected object.
- **Versions in Workspace Report:** Shows the version notes for all versions of an object in any product module of the current workspace.
- **Cross-reference Reports**
- **Xref Report:** Shows what other objects the selected object uses.
- **Program Usage Report:** Provides a diagram of the call sequences to a selected

program.

- **Call Diagram Report:** Provides a diagram of the calls between the programs in your system.
- **Unused Object Report:** Shows the registered objects in your system that are not being used by other objects.
- **External Objects Report:** Shows the objects that are referenced by objects in the workspace, but are not defined in the workspace as objects.
- **Where Used Report:** Shows where a selected object is used in the system.
- **Index Usage Report:** Provides a list of all programs that use a specified table index.

### 6.11.1. Versions in Product Module Report

The Versions in Product Module Report provides a history of the development of object versions for the currently selected object. This report helps to identify and integrate changes made to an object in different workspaces.

The report provides:

- Description of the object
- Status of the object
- Subtype (for PCODE objects)
- Details of the object

Use the Versions in Product Module Report dialog box to print a report about the versions of the selected object.

The Versions in Product Module Report Dialog Box fields:

Field	Description
Product Module	A field where you type the product module containing the object.
Object type	A field where you Enter the type of object.
Object	A field where you Enter the name of the object.
Version to Start On	A field where you Enter the starting version.
Version to Stop On	A field where you Enter the ending version.
Maximum Versions to Show	Prevents excessive numbers of versions from being printed when you are not sure how many versions exist across the version range specified.
Show Orphans?	Orphans exist when versions are created in multiple workspaces.
Show object	Prints object descriptions in the report.

Field	Description
description?	
Show version notes?	Prints object's version notes.

This report is available from the More option on any of the object menus. It provides a history of the development of object versions for the currently selected object. This report is helpful in identifying and integrating changes to an object made in different workspaces.

Follow these steps to print the Versions in Product Module Report:

1. From an object menu, choose More. The More Menu appears.
2. Choose Versions in Product Module Report. The Versions in Product Module Report dialog box appears.
3. Complete the fields and press the **GO** key.

### 6.11.2. Versions in Workspace Report

The Versions in Workspace Report will display the version notes for all versions of an object in any product module of the current workspace. This report shows the development of an object across one or more product modules.

The report provides:

- Object
- Versions of the object
- Product Module
- History Notes

Follow these steps to run the Versions in Workspace Report:

1. Position cursor on object.
2. Workspace Object Menu#More#Versions in Workspace Report.
3. The Report dialog box appears.
4. Enter preferences and press the **GO** key.

### 6.11.3. Program Usage Report

The Program Usage Report provides a diagram of the call sequences to a given program. This report is used to:

- Find the call sequences that lead to a program or include a file's usage
- Find all of the programs that are passed through to reach the specified program
- Ensure that shared buffers and variables are set in the programs that lead to the specified program

Follow these steps to print the Program Usage Report:

1. From the PROGRESS Code Object Menu, select Xref. The Xrefs Menu appears.
2. Choose Reports. The Xref Reports Menu appears.
3. Select the Program Usage Report. The Program Usage Report dialog box appears.
4. Complete the report fields:
  - a. Enter the name of the programmer include file for which to run the report in the Lead In Program field.
  - b. Enter the maximum number of levels to include in the report in the Max Levels field.
  - c. Enter the number of white spaces to use as indents for each level in the White Space field.
  - d. Activate the Extra line by selecting it to double space the report.
5. Press the **GO** key.

#### 6.11.4. Call Diagram Report

The Call Diagram Report provides a diagram of the calls among the programs in your system.

Follow these steps to print the Call Diagram Report:

1. From the PROGRESS Code Object Menu, select Xref. The Xrefs Menu appears.
2. Choose Reports. The Xref Reports Menu appears.
3. Select the Call Diagram Report. The Call Diagram Report dialog box appears.
4. Complete the report fields:
  - a. Enter the name of the program or include file for which to run the report in the Lead In Program field.

- b. Enter the maximum number of levels to include in the report in the Max Levels field.
  - c. Enter the number of white spaces to use as indents for each level in the White space field.
  - d. Active the Extra line by selecting it to double space the report.
5. Press the **GO** key.

### 6.11.5. Unused Object Report

The Unused Object Report provides a list of objects that are not referenced by any object in the workspace. Use this report to track down code that is no longer used in the system.

Some objects that are no longer used in, but are assigned to, the workspace appear on this report. For example, a menu.p program might appear on this report, if it is the root program for Entering the system and is not referenced by other objects in the system.

Any objects that are only called by RUN VALUE() statements also appear in the report.

To print the Unused Object Report:

1. From the PROGRESS Code Object Menu, select Xref. The Xrefs Menu appears.
2. Choose Reports. The Xref Reports Menu appears.
3. Select the Unused Object Report. The Print Destination dialog box appears.
4. Complete the print destination fields and press the **GO** key.

### 6.11.6. External Objects Report

The External Objects Report identifies objects that are referenced in the workspace, but are not defined in the workspace as objects.

To print the External Objects Report:

1. From the PROGRESS Code Object Menu, select Xref. The Xrefs Menu appears.
2. Choose Reports. The Xref Reports Menu appears.
3. Select the External Objects Report. The Print Destination dialog box appears.
4. Complete the print destination fields and press the **GO** key.

### 6.11.7. Where Used Report

The Where Used Report shows where a selected object is used in the system. For example, a Name field might be used in a number of different tables, including an invoice table, a mailing label table, etc.

Follow these steps to print the Where Used Report:

1. From the Object Menu (PCODE, DOC, PDBASE, PFILE, or PFIELD), choose Where. The Where Used Menu appears.
2. Choose Print. The Print Destination dialog box appears.
3. Complete the print destination fields and press the **GO** key.

### 6.11.8. Repository Check Report

The Repository Check Report is used to identify problems in the Roundtable repository database. This process can evaluate four different areas:

- Check repository records against object version records
- Check links between repository records
- Check for empty object versions
- Check for missing full source object versions

Follow these steps to print the Repository Check Report:

1. Choose Reports from the main menu and select Repository Check Report from the menu. The Repository Check dialog box appears.
2. Select the processes you would like to be reported and press the **GO** key.

### 6.11.9. Xrefs Menu

The Xref report prints the cross-references for the selected object. For example, to see all the fields used in a PFILE object, print the Xref report (this report is printed by selecting 'Print' from the menu bar) for that PFILE.

The Xrefs Screen is available from the PCODE object menu. It allows you to browse through a table of objects accessed by the compiled procedure.

The Cross Reference screen includes the following fields:

Field	Description
Type	

Field	Description
	<p>This field indicates the type of cross reference:</p> <p> <b>PROG</b> Program  <b>FILE</b> File  <b>INDEX</b> Index  <b>FIELD</b> Field  <b>INCLD</b> Include file used  <b>SVAR</b> Shared variable  <b>SWF</b> Shared Workfile  <b>SFRM</b> Shared Frame  <b>WFLD</b> Shared Workfile  <b>FieldPART</b> Object part included  <b>_FILE</b> Schema_file  <b>_FIELD</b> Schema_field  <b>_INDEX</b> Schema_index  <b>XINCLD</b> External include  <b>FUNC</b> Function Defined  <b>EVENT</b> Published, Subscribed, or Unsubscribed event    <b>INTPROC</b> Internal Procedure    <b>CLASS</b> Class created    <b>CONSTRUCTOR</b> Constructor defined    <b>DESTRUCTOR</b> Destructor defined    <b>METHOD</b> Method invoked or defined    <b>DATA-MEMBER</b> Data-member accessed or defined    <b>XINCLD</b> type is a little unusual in that it shows a reference to an include file that is not registered as part of the Roundtable system. This type of include should not normally exist if a workspace has an entire copy of an application in it. </p>
Reference	Name of the object accessed followed by a short description of the object.
Actions	<p>The types of actions performed on the referenced object by a compiled procedure:</p> <p> <b>A</b> Accessed    <b>C</b> Created    <b>D</b> Deleted    <b>E</b> Executed    <b>I</b> Included    <b>N</b> New </p>

Field	Description
	P Published R Referenced S Searched U Updated X External Reference  When an object contains a whole-index reference, the W action is appended to the A(ccessed) action (for the table) and the S(earch) action (for the table's primary index). This action alerts developers to possible performance issues.

The Xrefs Menu contains the following menu items:

Menu Item	Description
Select	Goes to selected object.
Compile	Compiles the current object with Xref and listing options.
Find	Finds an Xref record.
Listing	Views extended listing file created by PROGRESS compile.
Xlisting	Views Xref listing created by PROGRESS compile.
Where-used	Goes to the Where Used Menu.
Reports	Xref Reports Menu provides access to the Program Usage, Call Diagram, Unused Object, and External Objects Reports, which are all described in the "Reports Review" section.
Print	Prints the Xrefs Report.
Quit	Returns to the previous menu.

#### 6.11.10. Informal Xrefs Menu

The Informal Objects Xrefs Menu is accessed from various More menus. It allows the user to specify the type and name of an informal object. The program then provides a list of objects in the system that reference the informal object.

These objects are "informal" objects because they are not explicitly defined and tracked in Roundtable as objects.



Once you select the Informal Xrefs Menu, a box appears.

1. Enter the Ref-type and press **ENTER**. The pointer appears in the Object field.
2. Enter the informal Object name and press **ENTER**. The Informal Objects Xrefs Menu appears.

### 6.11.10.1. Cross Reference Frame

See field definitions in the Xrefs Menu section.

### 6.11.10.2. Referenced Object Details

This screen contains a description of the object selected in the top screen. Currently the only object type that can be tracked as referencing informal objects is PROG so the description frame contains status and descriptive information about the associated PCODE object version of the program.

The Informal Objects Xrefs Menu contains the following menu items:

Menu Item	Description
Select	Selects object for edit.
Change	Asks the user for a new informal object type and name.
Print	Prints the Xref Table.
Quit	Returns to the previous menu.

### 6.11.11. Where Used Menu

The Where Used Report shows where a selected object is used in the system. For example, a Name field might be used in a number of different tables, including an invoice table, a mailing label table, etc.

1. Access the Where Used Menu? from the PROGRESS Code Object Menu.
2. Choose Where. The following screen appears.

The Where Used Frame contains the following fields:

Field	Description
Ref Type	The type of cross reference:

Field	Description
	<p>PROG Program FILE File INDEX Index FIELD Field INCLD Include file used SVAR Shared variable SWF Shared Workfile SFRM Shared Frame WFLD Shared Workfile Field PART Object part included _FILE Schema_file FIELD Schema_field INDEX Schema_index XINCLD External include FUNC Function Defined EVENT Published, Subscribed, or Unsubscribed event</p> <p>INTPROC Internal Procedure</p> <p>CLASS Class created</p> <p>CONSTRUCTOR Constructor defined</p> <p>DESTRUCTOR Destructor defined</p> <p>METHOD Method invoked or defined</p> <p>DATA-MEMBER Data-member accessed or defined</p>
Src Type	<p>The type of source reference:</p> <p>DBASE Database</p> <p>FILE File</p> <p>PROG Program</p> <p>CLASS Class</p> <p>INTERFACE Interface</p>
Object	Object name/description
Actions	<p>The types of actions performed on the referenced object by a compiled procedure:</p> <p>A Accessed</p> <p>C Created</p> <p>D Deleted</p> <p>E Executed</p>

Field	Description
	<p>I Included</p> <p>N New</p> <p>P Published</p> <p>R Referenced</p> <p>S Searched</p> <p>U Updated</p> <p>X External Reference</p> <p>When an object contains a whole-index reference, the W action is appended to the A(ccessed) action (for the table) and the S(earch) action (for the table's primary index).</p>
Referenced Object Details	Contains information about the object using the current object.

The Where Used Menu contains the following menu items:

Menu Item	Description
Select	Selects object for edit.
Print	Prints the Where Used Table.
Quit	Returns to the previous menu.

Note that Where Used Menus are available for PCODE, PFIELD, PFILE, and PFILE/INDEX objects.

### 6.11.12. History Menu

Shows all changes made to selected object in current workspace.

1. Access the History Menu from the PROGRESS Code Object More Menu
2. Choose History.

Field	Description
Edit	Edit Version Note
Find	Find a version
Quite	Return to previous menu

---

## Tools

### 7.1. Introduction

This chapter documents a number of useful tools provided in the Roundtable environment. These include:

- Global Change Finder: Searches for and selects objects for check out into a current task
- Compiling Tools: Compile rules and utilities available in the system
- Module Load: Registers objects into Roundtable quickly
- Source Compare Report: Compares different versions of file-based objects quickly
- AppServer Interface: When developing distributed applications with Roundtable, a workspace may contain some client-side code objects and some server-side code objects. Roundtable provides a means to copy objects to an AppServer partition.

### 7.2. Global Change Finder

The Global Change Finder compares the source files in the current workspace with those stored in the repository and reports on objects where differences are found. This utility is useful when mass changes have been made to a system at the OS level. If you have a task selected, you can have the Global Change Finder create new versions of objects where changes are found.

The Global Change Finder only checks the objects out. It does not check in the new versions. This gives you the opportunity to review and change (or even cancel) any new versions created by the Global Change Finder before you check them in.

The Global Change Finder will also find if any source files have been deleted. If they have been deleted, Roundtable gives you the opportunity to delete the corresponding object record from the workspace.

#### 7.2.1. Global Change Finder Dialog Box

Follow these steps to use the Global Change Finder:

1. Select a task if you want to check out objects where changes to the OS files were made.

2. From the Workspace Config. Menu, choose Global Change Finder. The Global Change Finder dialog box appears.
3. Enter matches criteria for the modules to be scanned. Enter an asterisk (\*) to scan all modules. Press the **GO** key to start the scan. Roundtable lists the names of the objects as it scans them and displays whether or not it found changes in each object.
4. If a task is not selected, Roundtable displays the count of objects found with changes. If a task is selected, and changes were found, the Make New Versions dialog box appears.
5. Enter the New Pmod (product module). Do not change this field if you just want to create a new version of the object. Select a different product module if you want this to be a custom variant of the object.
6. Enter the new version level (Version, Revision, or Patch) into the Bump field. Type N for No Change if you do not want to check the object out.
7. Type Yes in the Full Source field if you want the full source of the version to be checked into the repository rather than just the differences (deltas) from the previous version. This field has no affect until the object is checked in.
8. Enter the History Note to be used for the new version.
9. Type Yes into the Skip Prompts field if you want Roundtable to use your entries from this object version for the rest of the objects that it found with changes. Roundtable processes the remainder of the objects with changes found, without stopping to prompt you.



Roundtable does not create custom variants for any objects that are checked out after the Skip Prompts option is used. Do not use Skip Prompts if you want to create custom variants for any of the object changes found.

10. Press the **GO** key. If you Entered V, R, or P in the Bump field, Roundtable checks the object out. If you selected a different product module, you are prompted for further details about the new custom variant. If you Entered yes to Skip Prompts and N in the Bump field, then none of the remaining objects with differences are prompted and they are not checked out. If you Entered yes to Skip Prompts, and Entered V, R, or P in the Bump field, Roundtable checks out the remaining objects with differences without prompting any further. Roundtable uses the same version bump level and the same version History Notes when checking out the remaining objects.

## 7.3. Compiling Tools

Roundtable provides sophisticated tools for recompilation of procedure objects in your

workspace. In this section, you learn how to:

- Select a workspace
- Compile a workspace or object

Roundtable provides three compile options:

- Compile and save
- Compile with Xrefs
- Selective compile

All the compile options follow the same set of rules and run the same compile process. The compile process performs different actions based on:

- Current task
- Share status of an object
- Full compile flag
- Xref flag

The compile rules are discussed below. These rules are especially important to understand if you are using the share status option. When you use a share status other than Central, two copies of the same object exist, one in the workspace and one in the task directory. The object in the workspace is the previously completed version, and the object in the task directory is the work-in-process version. For more information on the share status option, see Section 5.3.2.1, “Share Status” [5–4].

### **7.3.1. Compiling Rules**

All compile options in Roundtable follow the same set of rules and call the same compilation process. The compile actions on an object are controlled by the following values:

- Currently selected task
- Share status of the object
- R-code and s-code flags specified for the workspace or workspace module
- Full compile flag
- Xref flag



R-code and Xref listings are now generated for objects in a task directory.

When the share status of an object is not central, two copies of the objects source exist. One copy is stored in the workspace directory structure. The other is stored in the task directory structure.

The following compile rules are affected when the share status of an object is not central:

- Compilation of source in a task directory occurs only when the task owning the directory is selected. In the case of the selective compile, the task number is specified in the selective compile dialog. Compilation of source in a task directory is always done with the PROPATH precedence of task directory, group directory, and then workspace directory.
- Compilation of source in workspace directories is always done with the PROPATH including only workspace directories. Sources in task and group directories are never included in the compilation of a source file in a workspace directory.
- R-code is never generated for the source compiled in group directories.
- R-code can be generated for source compiled in a workspace directory based on the value of the R-code flag in the workspace and workspace module specifications and the Save toggle in the Object Config folder. If the Save toggle is No, then R-code is not generated. If the workspace module R-code specification is No, then R-code is not generated. If the workspace R-code specification is No, then R-code is not generated.
- Xref records are only generated on compiles of source in the workspace directory. Compilation of source in task or group directories does not generate cross-reference data.
- When a current task is selected and an object belonging to the task is selected for compile, both the source in a task directory and workspace directory will be compiled. The source in the task directory is compiled because the object belongs to the currently selected task. The source in the workspace module is compiled to refresh the Xref information using the central object.

### 7.3.2. Compile Object Without Xref

Using the F8 or CTRL-Z keys performs a compile but does not produce Xrefs. This option is faster than a compile with Xref but does not change the update status of the object to Current. An object must have a status of Current before it can be checked in. A compiled .r file is not produced for objects that have a share status of other than Central. Use the Compile Object Without Xref to check the syntax of any procedure and update the .r file of the procedure if the object has a share status of Central.

1. From the Workspace Modules Menu or the PROGRESS Code Object Menu, select



the object.

2. Press the F8 key to compile without cross references. If there are any compile errors, Roundtable displays them.

### 7.3.3. Compile with Xref

Use the F9 key or CTRL-N to compile with cross references. This function performs a compile and produces Xrefs records if the object has a share status of central. A compile with Xref changes the update status of an object from Modified to Complete if the compile is successful.

1. From the Workspace Modules Menu or the PROGRESS Code Object Menu, select the object.
2. Press the F9 key to compile with cross references. If there are any compile errors, Roundtable displays them.

### 7.3.4. Selective Compile Specifications Screen Description

The Selective Compile Specifications screen lets you specify the task number, module, object group, object type, and object names to compile. For all fields except the task number, you can Enter an asterisk (\*) to signify all (such as all object types, all object names, etc.). The selective compile process compiles objects selected by the criteria and any object that is affected by a selected object. For example, if an include file is found because of your selection criteria and the include file has changed, then any programs that use the include file are recompiled.

The following table lists the Selective Compile Specifications screen fields:

Field	Description
Task Number	A fill-in for the task number for the compile. If you enter a task number, Roundtable restricts its search for objects to those included in the specified task. If you enter a value of 0, the search is not restricted to any task.
Module	A fill-in for the module for the compile or an asterisk (*). If you enter a specific module, Roundtable restricts its search for objects to that module.
Object Group	A fill-in for the group to compile or an asterisk (*). If you enter a specific group, Roundtable restricts its search for objects to that group.
Object Type	A fill-in for the specific type of object or an asterisk (*). Roundtable restricts its search for object to objects of the

Field	Description
	specified type.
Object	A fill-in for the object name or a match expression using an asterisk (*) placeholder (wildcard). For example, you could specify a value of "ar*.p" to restrict Roundtable to searching for objects that begin with "ar" and end with ".p".
Full Compile	Yes in this field indicates that each object compiled is cross referenced.
Force Compile	Yes in this field indicates that all compilable objects matching the selection criteria are recompiled.
Listing	Yes in this field indicates that each object compilation produces a compile listing. This is the Listing parameter in the Progress COMPILE statement. Roundtable saves the listing file as a list/*.l file. An Xref listing is also generated if a full compile was performed.
Show Meter	Yes in this field displays a percentage completion meter during compilation.

### 7.3.5. Selective Compiles

A selective compile gives you a number of options. You can Enter a specific task to compile or you can **ENTER** 0 to disregard the task number. If you specify a task number and a PFIELD or PCODE, Roundtable compiles all affected PCODEs (disregarding task number). If you **ENTER** 0 for the task number, Roundtable compiles:

- Objects in the current task that have changed
- Objects outside the current task that are referenced by objects within the task that have changed
- Objects within the current task that are referenced by objects outside the task that have changed

Follow these steps to selectively compile objects in your workspace:

1. From the Workspace Objects Menu or the PROGRESS Code Object Menu, choose Compile. The Selective Compile Specifications screen appears.
2. Enter your selective compile options in the fields. These options include the task number, module, object group, object type, and object name. For each of these fields, except the task number, you can Enter an asterisk (\*) to indicate all.
3. Press the **GO** key to compile. If you selected Yes for Show Meter, the Compiles in PROGRESS dialog box is displayed during the compile.

4. Press the SPACE BAR to return to the Workspace Modules Menu.

## 7.4. Module Load

Module Load is a utility program that loads large numbers of PCODE objects into a workspace managed by Roundtable. To load the objects, Roundtable scans a directory specified by a workspace module for files that have not already been defined as PCODE objects in your workspace. By browsing through this table, you can choose the files needed to create PCODE objects.

Prior to using Module Load, you must define your subtypes and products. In addition, you must create the workspace in which the PCODE objects exist.

You can customize Module Load to some extent. This is done by modifying the programs `rtb/rtb/p/rtb_scan.p` and `rtb/rtb/p/rtb0282.p`. `rtb` is the directory in which you installed Roundtable. These programs are largely self explanatory. If you modify these programs, be sure to delete the existing `.r` file. You should also keep a copy of your modified program in your home directory so that future updates to Roundtable do not overwrite your copy.

### 7.4.1. Running Module Load

When you first load a system into Roundtable, you can use the Module Load utility to load a large number of PCODE objects quickly. The Module Load utility works by scanning a workspace module for files and then presenting these files to the user in a table. The user can then move through this table adding these source ode files into the Roundtable repository. Parts of the Module Load utility can be modified to make loading even large systems a quick process.

Before using Module Load, you need to:

1. Define subtypes and products. See Section 3.6, “Defining a Configuration Hierarchy” [3–8], Section 3.9, “Subtypes” [3–21], and Section 3.7, “Products” [3–13].
2. Create the workspace in which the PCODE objects exist. See Section 4.2.3, “Adding a Workspace” [4–3].
3. Assign workspace sources. See Section 4.3, “Workspace Sources” [4–11] section.
4. Create and select a task.

Follow these steps to run the Module Load Utility:

1. From the Workspace Modules Menu, select the module that you want to load.
2. Choose Load. The Roundtable Module Load dialog box appears.

3. Choose Yes to run the Module Load to scan your source for comments. Choose No to run the Module Load without scanning your source for comments. Choose Cancel to quit.
4. Enter the file specification to load.

The Module Load Menu appears.

The following table lists the Module Load Menu fields:

Field	Description
Select	An asterisk (*) depicts whether the object has been selected for loading.
Status	New indicates that the PCODE object has not been created. Done indicates that a PCODE object was created for the source file.
Filename	The name of the source file.
Sub Type	The name of the subtype to assign to the new PCODE object The subtype can be changed through the Properties dialog box.
Pmodule	Product module that the PCODE object will be assigned to. The Pmodule can be changed through the Properties dialog box.
Object Group	Used for sorting objects within a module. The Object Group can be changed through the Properties dialog box.
Level	Used for sorting objects within a module. The level can be changed through the Properties screen.

The following table lists the Module Load Menu items:

Menu Item	Description
Toggle	Selects and deselects the file for loading.
Select	Allows you to select many files in the list using a MATCHES criteria.
Deselect	Allows you to quickly deselect many files in the list using a MATCHES criteria.
View	Allows you to view the Source file using the editor defined by the VIEW option in your RTBSETUP file. If the VIEW option is not defined in your RTBSETUP file, then the PROGRESS Procedure Editor is used.
Properties	Allows you to change the properties of selected files. If you have selected more than one file, some of the properties (such as Object Description) are not available for changing. Any changes

Menu Item	Description
	that you make will affect all of the selected files. Object properties that you can change are: product module, code subtype, object group, object group level, summary object description, and detailed object description.
Load	Initiates the load of all selected files. Only selected files with code subtypes and product modules defined can be loaded. Files that have been loaded are indicated by a status of Done.
Unload	Deletes the Roundtable object records created by the load process. Select Unload if you change your mind about some files that you loaded. Only files with a status of Done can be unloaded.
Find	Allows you to locate a file by its filename.
Quit	Returns to the previous screen.

1. Enter the matching criteria for filenames to select.
2. To change the properties of one or more selected objects, choose Properties. The Roundtable Code Properties for Module screen appears.

The following table lists the Properties dialog box fields:

Field	Description
Sub Type	Edits the name of the subtype to assign to the new PCODE object
Group	Edits the group name used for sorting objects within a module.
Product Module	Edits the Product module that the PCODE object is assigned to.
Level	Edits the level field used for sorting objects within a module.
Summary Description	Provides a short description of the PCODE object. This is used often in reports.
Detailed Description	Provides an additional description of the PCODE object.

1. Enter the Subtype, Group, Product Module, Level, Summary Description and Detailed Description. Some of these fields will not be editable if you have selected more than one file. Any changes that you make in the Properties dialog box will affect all of the selected files.
2. Choose the Load menu item to create the PCODE object for the selected files. This initiates the load of all selected files. Only selected files with code subtypes and product modules can be loaded. Files that have been loaded are indicated by a status of Done.

### 7.4.2. Test Version Integrity

Use the test version integrity utility to determine if a file in the workspace directory has been modified outside of Roundtable. This tool performs a CRC check on the contents of the object in the operating system directory and then compares it with the CRC generated when the object was stored in the repository. If these CRC values do not match, then the object has been modified outside of Roundtable. WIP objects cannot be checked in this manner.

Follow these steps to check the version integrity of a PCODE Object:

1. From the Workspace Objects Menu or the PROGRESS Code Object Menu, select the PCODE Object.
2. Choose More. The More Menu appears.
3. Select Test Version Integrity. The Test Version Integrity dialog box appears:

If differences are found between the repository and the file in the workspace directory, Yes is displayed in the Differences field.

4. Press the SPACE BAR to return to the Workspace Objects Menu.

### 7.5. Source Compare Report

The Source Compare Report is available from the More menu on the PROGRESS Code Objects Menu. It provides a compare facility for any two similar PCODE objects. When the report is selected, you are prompted for the following parameters.

The following table lists the Source Compare Report options:

Option	Description
Pmodule	Product module of the object.
Object Type	Allows PCODE or DOC. The second object must have the same object type as the first object.
Object	Compares objects having different names as long as the subtypes of the objects have the same number and types of parts.
Version	Compares version codes of objects. Roundtable pulls the specified version of the object from the repository and places it in temporary files for the comparison.
Object Details	Prints object descriptions in the report.
Show Update History	Prints object versions' update history in the report.
O/S File Compare	Command with options to use for source compare. The filenames are supplied as arguments after any options. The Unix diff command varies from system to system. Refer to your Unix

Option	Description
	manual for output options available. Most often, use the -D and -e options with diff.

### 7.5.1. Comparing Files with the Source Compare Report

Follow these steps to compare PCODE objects:

1. From the Workspace Objects Menu or the PROGRESS Code Object Menu, select a PCODE object.
2. Choose More. The More Menu appears.
3. Select the Source Compare Report. The Object Version Comparison Report dialog box appears.
4. Complete the fields for both objects and press the **GO** key. The Print Destination dialog box appears. Complete the print fields and press the **GO** key.

### 7.5.2. Partner Site Load

The Partner Site Load function is an interface for loading workspace information deployed as part of a partner site deployment. Partner site deployments are used to move information among different Roundtable Repositories.

Follow these steps to access the Partner Site Load option:

1. From the Main Menu, choose 3. Partner Site Load. The Import Remote Workspace Utility dialog box appears.
2. Enter the Workspace ID and press the **GO** key. The Description and PROPATH for the Workspace dialog box appears.

Note that you must know the Workspace ID in advance. It is not possible for Roundtable to find this information or for you to change it after the load process is started.

3. If the workspace has been loaded before, then the description and path information appears. If this is the first time information for the workspace has been loaded, you must specify both a description and a path.
4. Enter a description of the workspace and press **ENTER**.
5. Enter the full path to the workspace directory and press **ENTER**.
6. Enter any additional path components and press the **GO** key. Roundtable loads the Partner Site information.

If the workspace has not been loaded before, you have to find each PDBASE object managed by the workspace and specify the connection parameters for the workspace's databases.



If cross reference information is being maintained for the workspace, you must do a full workspace compile to ensure that it is up to date after the load process.

## 7.6. Load Schema

The Load Progress Schema utility loads the schema of an existing database into Roundtable. This utility creates PFILE and PFIELD objects for each of the file and field definitions found in the Progress database.

Load Schema can load changes made to the Progress database schema. If changes are made to the physical database schema by some tool other than Roundtable (such as the Progress Data Dictionary), then you can use Load Schema to bring Roundtable's logical schema object definitions in sync with the physical Progress database schema.

### 7.6.1. How to Run the Load Schema Function

Follow these steps to run the Load Schema function:

1. Make sure that there are no WIP schema objects in the workspace. The Load Schema process actually performs this check for you before it starts.
2. The Load Schema process locks the workspace while it runs. Nobody can modify objects in the workspace while you run Load Schema.
3. Create and select a task.
4. Find and select the PDBASE object schema you want to load.
5. Choose the Load option on the Database Object Menu.
6. Press the **GO** key to complete this dialog.

The following table lists the Load Schema dialog box fields:

Field	Description
Object Prefix	Normally, this field should be left blank. An object-naming prefix is required when you are loading a database into Roundtable that has files of the same name as those in a database previously loaded. The prefix allows Roundtable to create object names unique to the new database.



Field	Description
Update Version Level	Do you want revised schema objects to be checked out with Version, Revision, or Patch?
Product Module	Normally, this should be left <No Change>. Roundtable creates new versions of schema objects in the product modules they already belong to. If the schema object does not already exist, it will be checked out into the same product module as its parent schema object. For example, new PFILE objects are created in the same product module that the PDBASE object belongs to. If you select a specific product module, then new schema objects will belong to that product module.

1. Roundtable displays the following status window while it scans the schema of your workspace database.
2. If sequences, tables, or fields are dropped and more are added, you will be prompted if any of them are renamed.

You must indicate if any sequences, tables, or fields are renamed. Otherwise, when schema updates are done in the target workspaces or target deployment sites, the tables will be dropped and re-added, and their data lost.

If you choose to rename any tables or fields, then only the "local" table or field name is changed in the Roundtable schema records. The name of the Roundtable object does not change. This can cause some minor confusion, because the PFILE or PFIELD object name will not match the physical name of the corresponding table or field. For example, if you rename a table from "cust" to "customer", you will have a PFILE object named "cust", but it will be represented as a table with the name "customer" in the physical database schema. This is not a problem for sequences, because sequences are represented as attributes of the PDBASE object; there is no such thing as a "sequence" object in Roundtable.

If data loss in target workspaces or deployment sites is not an issue, you can ignore this rename option. The original PFILE or PFIELD will be dropped, and a new one added. The object name remains the same for the physical table or field name.

Follow these steps to select a sequence, table, or field for renaming:

- a. Select the renamed item from the first selection list.
- b. Press TAB to go to the second selection list.
- c. Select the new item name from the second selection list and press SPACE. The new item name appears in the Delete or Rename to: column of the first selection list.

- d. Repeat these steps for any other renamed items.
  - e. Press the **GO** key to complete this dialog.
3. If there is an object naming conflict, you will be prompted to provide a new object name. For example, if your workspace already has a PFILE object named "customer", and your new schema re-adds a "customer" table, you must either provide an alternate name for the PFILE object, or specify a different product module that it can be added to.
  4. If any tables or fields are dropped, then the following dialog box appears:  
  
Normally, you should choose Yes to delete any unused object definition records from your workspace. Choose No to re-assign the schema objects back into one of your workspace databases.
  5. If the schema load is interrupted for any reason, it should be safe to restart it. However, you might have to complete any WIP schema objects first.
  6. Once the Load Schema is done, complete your task by checking in all of the new or modified schema objects.

### 7.6.2. How to Check for a Successful Load Schema Completion

Once the Load Schema is done and your task is completed, check the following:

- The Database Integrity Check Report should indicate no differences.
- The Workspace Schema Update should not find any schema that requires updating.

### 7.6.3. Load Schema and Object Domains

Load Schema only reads one physical database, and it makes changes to logical object definition as appropriate. Table and field domains present potential problems. However, with a little care, these problems can be avoided entirely. If a PFIELD object definition is used more than once within a database, then Load Schema sets the attributes of the PFIELD object based on the last physical field definition that it looks at. This means that you are not guaranteed that your field domains will all be in sync by just running the Load Schema process.

If you use field domains and changed your schema outside of Roundtable, make sure you run the Database Integrity Check Report after running Load Schema. You will have to manually re-synchronize any fields that fell out of step.

To avoid this problem, do not make changes to field domains outside of Roundtable's schema management.

If a PFILE or PFIELD object definition is used in more than one database, Roundtable and your schema might fall out of step if you make changes to one of those databases outside of Roundtable's schema management.

The following steps might cause this problem:

1. Use a PFIELD or PFILE object definition in more than one of your workspace databases. (Call the databases "A" and "B").
2. Use a tool other than Roundtable (for example, the Progress Data Dictionary) to change the related object's definition in just one of the databases (database "A").
3. Run Load Schema against database "A".

Database "A" and Roundtable are in sync, but database "B" and Roundtable are now out of sync.

If you use field or table domains, and have changed your schema outside of Roundtable, be sure to run the Database Integrity Check Report on all of your databases after running Load Schema. You will have to manually re-synchronize any fields or tables that fell out of step. Depending on the number of schema objects that are out of sync, it might be easier to use the Progress Data Administration tools to dump a .df between the two databases, and then apply the necessary parts of the .df file to the database to bring it back in step.

To avoid this problem, do not make changes to field domains or table domains outside of Roundtable's schema management.

## 7.6.4. Unload Schema Utility

The unload schema utility removes the selected PDBASE object from a workspace and all PFILE and PFIELD objects that are associated with it. The process is cancelled if any PFILE or PFIELD objects are used in another PDBASE object. The unload schema utility is accessed via the Unload option of the Progress Database Object Menu for a selected PDBASE object.



The remove schema utility does not support the use of schema domains.

## 7.6.5. Deploying Roundtable Objects to an AppServer

When developing distributed applications with Roundtable, a workspace may contain some client-side code objects and some server-side code objects. Although Roundtable allows workspaces to be contained on remote systems using UNC paths and mapped network drives, developers may wish to place server-side objects on an AppServer partition that is not accessible through the workspace path for testing or local deployment

purposes.

Sites that require access to AppServer partitions from Roundtable that do not have a means of accessing a remote directory through the workspace path need a way to move objects from Roundtable workspace to an AppServer partition.

This release of Roundtable provides a means for copying objects (both source and/or r-code) from a workspace to an AppServer partition using AppServer itself, rather than using FTP, NFS or other file transfer methods. Copying an object to an AppServer partition depends upon Roundtable connecting to an AppServer broker (defined using the Progress AppServer Partition Deployment Tool). Then sending the object to a server-side program to place the object on the AppServer partition.

If you can access your AppServer partition through the workspace path (i.e. the AppServer partition directory is the same as the workspace path, or a subdirectory under the workspace path as defined by a workspace module and/or code subtype), there is no need to move objects from a Roundtable workspace to a remote location. These instructions are intended for sites where the AppServer partition is not accessible through the workspace path.

Enabling Roundtable to copy objects to an AppServer partition using Roundtable's Appserver Interface Tool requires three setup steps:

1. Setup (an) Appserver(s) according to Progress documentation
2. Enter the AppServer configuration parameters into the Progress AppServer Partition Deployment Tool.
3. Use FTP, NFS, Sneaker-Net, or other file transfer method to copy the Roundtable program located in `rtb/appsrvr/rtb_aswrt.p` under the Roundtable installation directory to the AppServer partition(s). The `rtb_aswrt.p` program may be placed anywhere on the AppServer partition, so long as it is in the App-Service's `PROPATH`. Compile the `rtb_aswrt.p` program on the remote system.

Follow these instructions to move objects from a workspace to an AppServer partition, once the setup has been done:

1. Select AppServer Interface from the Workspace Objects More menu.
2. Select the AppService, Enter the remote path (root) of the target partition (using the **Remote Path** button) , and choose a Workspace Module that contains objects to move. From the objects displayed in the browser, select the objects to move to the AppServer partition.



The entire path of the remote object will then consist of the: - remote path - subdirectories indicated by Workspace Modules and/or code subtypes - object name

3. Check the r-code box if only run time code is to be moved.
4. Check the Dependencies toggle-box if you want Roundtable to move referenced files with an object (Roundtable will perform a single level Xref on the object and move the include files and programs that the object calls with the object). This option is not available when moving r-code.
5. Choose the **View** button (enabled when Dependencies is checked) to view dependent objects that will be moved with the selected object.
6. Choose the **Move** button to move the selected objects to the specified AppServer partition.
7. Choose the **Done** button to close AppServer Interface window.

---

---

## Interfaces

### A.1. Introduction

This appendix describes the entry points into the Roundtable environment that you can use to customize your environment.

- Environment Setup on Execution: rtbrun.p [A-1]
- Edit Program: rtb/p/rtb\_open.p
- Managing Application Data with the Edit Program
- Naming Program: rtb/p/rtb\_bnam.p [A-2]
- Build Program [A-4]
- Report Source Files Locations [A-5]
- Task Record Access Using the Ref-num Field [A-6]
- Roundtable Propath Management
- Roundtable Interfaces with the ADE
- Hooks and Application Programming Interfaces
- External Session Compiling

### A.2. Environment Setup on Execution: rtbrun.p

When you run a program in your application, Roundtable looks for a program named rtbrun.p in the root directory of the workspace. This program, if it exists, can perform setup activities for the application. For example, the setup might include creating global variables. Here is a sample rtbrun.p program:

```
/* EXAMPLE OF A "RTBRUN.P" PROCEDURE */  
  
DEFINE INPUT PARAMETERS Prunfile AS CHARACTER NO-UNDO.  
DEFINE VARIABLE Mi AS INTEGER NO-UNDO.  
  
/* place your global definitions here */  
  
/* make sure the program (Prunfile) is available */
```

```

IF SEARCH(Prunfile) = ? THEN DO:
  Mi = INDEX(Prunfile, ".p").
  IF Mi = 0 THEN Mi = INDEX(Prunfile, ".p").

  IF Mi <> 0 THEN DO:
    Prunfile = SUBSTRING(Prunfile, 1, Mi - 1) + ".r".
    IF SEARCH(Prunfile) = ? THEN RETURN.
  END.
ELSE
  RETURN.
END.
RUN VALUE(Prunfile).

```

You are responsible for creating and maintaining this program. It is usually a good idea to make this procedure a part of your application code. If you do not want to include it in your deployments, just place an exclamation point (!) in the Encrypt field of the object. Remember that the exclamation point instructs the system to never deploy the object.

### A.3. Naming Program: `rtb/p/rtb_bnam.p`

A specialized naming program can be associated with a subtype. The sample program `rtb_bnam.p` takes the naming code defined in the module definition and appends a four-digit number to it.

For example, if the module naming code was `ord` then the default names for objects created in the module would be `ord0001.p`, `ord0002.p`, etc. Naming programs provide only default names; the user always gets a chance to modify the object name. The `rtb_bnam.p` program works by finding the object with the highest value, then increments that value by one. This is helpful if you want to leave gaps in the number scheme. For example, if you manually named an object `ord0100.p`, the next object name supplied by the naming program would be `ord0101.p`.

It is likely that the source shipped with the system is more current than that shown here, so you may want to print it out to see a more recent version of this procedure.

```

/* rtb_bname.p -- Generate new object name using
   module counter */

DEFINE INPUT PARAMETER Ppmod AS CHARACTER NO-UNDO.
DEFINE INPUT PARAMETER Pobj-type AS CHARACTER NO-UNDO.
DEFINE OUTPUT PARAMETER Pobject AS CHARACTER NO-UNDO.
DEFINE OUTPUT PARAMETER Perror AS CHARACTER NO-UNDO.

DEFINE VARIABLE Mnum AS INTEGER NO-UNDO.
DEFINE VARIABLE Mi AS INTEGER NO-UNDO.
DEFINE VARIABLE Mlast-object AS CHARACTER NO-UNDO.

```



```

DEFINE VARIABLE Mno-count AS LOGICAL NO-UNDO.
DEFINE VARIABLE Mname AS CHARACTER NO-UNDO.

{rtb/g/rtbglobl.i NEW}

Perror = "Error! rtb_bnam: Unknown error".

FIND rtb.rtb_pmod
WHERE rtb.rtb_pmod.pmod = Ppmod
NO-LOCK.

FIND FIRST rtb.rtb_moddef
WHERE rtb.rtb_moddef.module = rtb.rtb_pmod.module
NO-LOCK.

FIND FIRST rtb.rtb_ver
WHERE rtb.rtb_ver.obj-type = Pobj-type
AND rtb.rtb_ver.object BEGINS rtb.rtb_moddef.naming-code
NO-LOCK NO-ERROR.

Mnum = 0.
Mlast-object = "".

OBJECT-PASS:
DO WHILE AVAILABLE rtb.rtb_ver:

Mlast-object = rtb.rtb_ver.object.

Mname = rtb.rtb_ver.object.

RUN rtb/p/rtb_snam.p /* strip directory and extension */
(INPUT-OUTPUT Mname).

Mno-count = FALSE.
DO Mi = 4 TO LENGTH(Mname):
IF NOT CAN-DO("0,1,2,3,4,5,6,7,8,9", SUBSTRING(Mname,Mi,1))
THEN DO:
Mno-count = TRUE.
LEAVE.
END.
END.

IF NOT Mno-count THEN Mnum =
    MAX( INTEGER( SUBSTRING( Mname, 4 ) ), Mnum ).

FIND NEXT rtb.rtb_ver
WHERE rtb.rtb_ver.obj-type = Pobj-type
AND rtb.rtb_ver.object BEGINS rtb.rtb_moddef.naming-code
AND rtb.rtb_ver.object <> Mlast-object
NO-LOCK NO-ERROR.
END.

```

```
Mnum = Mnum + 1.

Pobject = SUBSTRING(rtb.rtb_moddef.naming-code,1,3) +
  STRING(Mnum,"9999").

IF Mnum > 9999 THEN Perror = "Error! Counter limit
  exceeded in rtb_bnam".
ELSE
Perror = "".
```

## A.4. Build Program

Optionally, a specialized build program can be associated with a subtype. A build program creates the source file parts of an object being created for the first time. A simple build program can be commands that copy a template specified in a code subtype into the new object part files. If you are ambitious, you could tie a build program to your own program generator. Because a build program needs to know quite a bit about the workspace, task, and specific object you are working with, Roundtable provides the following shared variables:

```
/* --- Define shared --- */
DEFINE SHARED VARIABLE Urtb-object AS CHARACTER NO-UNDO.
DEFINE SHARED VARIABLE Urtb-part AS CHARACTER EXTENT 10 \
  NO-UNDO.
DEFINE SHARED VARIABLE Urtb-part-desc AS CHARACTER EXTENT \
  10 NO-UNDO.
DEFINE SHARED VARIABLE Urtb-path AS CHARACTER EXTENT 10 \
  NO-UNDO.
DEFINE SHARED VARIABLE Urtb-name AS CHARACTER EXTENT 10 \
  NO-UNDO.
DEFINE SHARED VARIABLE Urtb-tmpl AS CHARACTER EXTENT 10 \
  NO-UNDO.
DEFINE SHARED VARIABLE Urtb-num-parts AS INTEGER NO-UNDO.
DEFINE SHARED VARIABLE Urtb-sub-type AS CHARACTER NO-UNDO.
DEFINE SHARED VARIABLE Urtb-userid AS CHARACTER NO-UNDO.
DEFINE SHARED VARIABLE Urtb-task-num AS INTEGER NO-UNDO.
DEFINE SHARED VARIABLE Urtb-task-desc AS CHARACTER NO-UNDO.
DEFINE SHARED VARIABLE Urtb-propath AS CHARACTER NO-UNDO.
DEFINE SHARED VARIABLE Urtb-ws-propath AS CHARACTER NO-UNDO.
DEFINE SHARED VARIABLE Urtb-mod-dir AS CHARACTER NO-UNDO.
```

Variable	Description
part-num	Part numbers 1-9.
Urtb-object	The name of the object without an extension.

Variable	Description
Urtb-part[part-num]	The one-character part identifier defined in the subtypes window.
Urtb-part-desc[part-num]	The textual description of the part defined in the subtypes window.
Urtb-path[part-num]	The full path to the object part file in the workspace.
Urtb-name[part-num]	The name of the part file without a path.
Urtb-tmpl[part-num]	The path to the template file specified in the subtypes window.
Urtb-num-parts	The number of parts. The minimum number is 1.
Variable	Description
Urtb-sub-type	The subtype code.
Urtb-userid	The current user ID.
Urtb-task-num	The number of the currently selected task.
Urtb-task-desc	The description of the currently selected task.
Urtb-propath	The PROPATH for Roundtable.
Urtb-ws-propath	The PROPATH for the currently selected workspace.
Urtb-mod-dir	The full path to the currently selected workspace module directory.

## A.5. Report Source Files Locations

All of the reports in the Roundtable system are provided in source form for your convenience. If you decide to modify a report, copy it to a new directory with a new name so that your modifications are not overwritten by future updates of Roundtable.

File Location	Report
rtb/p/rtb_0603.p	Call Diagram Report
rtb/p/rtb_1101.p	Database Definition Report
rtb/p/rtb_0404.p	Workspace Event History Report
rtb/p/rtb_0601.p	External Objects Report
rtb/p/rtb_1201.p	Table Definition Report
rtb/p/rtb_1007.p	Object Orphans in Task Report
rtb/prtb_1100.p	PDBASE Object Report
rtb/p/rtb_1300.p	PFIELD Object Report
rtb/p/rtb_1200.p	PFILE Object Report

File Location	Report
rtb/p/rtb_0200.p	Product Module Report
rtb/p/rtb_0300.p	Product Report
rtb/p/rtb_0604.p	Program Usage Report
rtb/p/rtb_0405.p	Release Report
rtb/p/rtb_0500.p	Task Report
rtb/p/rtb_0401.p	Uncommitted Changes Report
rtb/p/rtb_0602.p	Unused Objects Report
rtb/p/rtb_1000.p	Version Ancestry Report
rtb/p/rtb_0406.p	Workspace Changes Report
rtb/p/rtb_0400.p	Workspace Report

## A.6. Task Record Access Using the Ref-num Field

Tasks are a significant part of managing work with the Roundtable system. You might want to retrieve information on tasks in the Roundtable system using a key field value that is meaningful to you. A field called user-task-ref is included in the task record and indexed by rtb\_task07. Since the Roundtable repository is a Progress database, you can use standard Progress 4GL commands to access the task records in the database.

Be careful to find records NO-LOCK in your reporting routines to avoid competing with Roundtable for locks on these important records.

## A.7. Hooks and Application Programming Interfaces

If you would like to be able to write your own programs that interact with Roundtable, the following section will describe some of the options available.

### A.7.1. Hooks

There is a program called rtb\_event.p, which can be found in the directory where you installed Roundtable. It provides hooks into Roundtable, similar to the way that adecomm/\_adeevent.p provides hooks into Progress's ADE. .

By modifying rtb\_evnt.p, you can intercept or filter various events that occur in a Roundtable session. Roundtable calls this routine when certain processing occurs. Below you will see a list of parameters and supported events, please see documentation inside the rtb\_evnt.p program code for more information.

### Input Parameters

p_product	Currently, this is always "Roundtable".
p_event	The name of the event (eg. "Close", "Open", "Before-Save")
p_context	A unique context string that can be used to compare the object, or context, of an event. (eg. This might be a RECID or HANDLE of the object being closed or opened).



This context string will be unique for a given class of events in a given product, but not necessarily across products or events. That is, an event for a given window will always have the same context id, even if the name of the window changes.

p_other	Additional information passed about an event. (eg. a save event normally passes the File Name for the save).
---------	--

## Output Parameters:

p_ok	On a case by case basis, this procedure allows you to cancel various Roundtable events.
------	---

Here are the currently supported events:

1. Create new object
2. Compile object
3. Complete object
4. Check out object
5. Create deployment
6. Create release
7. Perform schema update
8. Create task (Before, during, and after)
9. Complete task
10. Create deployment site
11. Import into a workspace
12. Changing workspaces
13. Receipt of object on AppServer partition (AppServer only)
14. Assigning an object to a workspace

15. Moving an object(s) to web server using the Web Interface tool
16. Creating a custom variant (moving object to a new pmod)
17. Partner Site deployment and load

### A.7.2. Application Programming Interfaces (API)

Roundtable comes with a program called `rtb/p/rtb_api.p` which provides you with a single point of entry for doing API calls into Roundtable.

`rtb/p/rtb_api.p` is meant to be run persistently or used as an example to build your own. It contains internal procedures that call API wrappers. Only objects of type PCODE are supported with these API calls. However, please note that you still must pass the object type parameter as PCODE.

The `rtb/x/rtb_apiinit.p` file provides you with an example of how to call `rtb/p/rtb_api.p` and the setup required. This accepts the setup information (global variables, `PROPATH`) as input parameters and instantiates the Roundtable procedure persistently.

`rtb_apiinit.p` accepts three parameters:

1. Workspace to work with
2. UserId
3. Path to Roundtable directory

Please see detailed documentation inside the `rtb_api.p` and `rtb_apiinit.p` programs. `rtb_api.p` contains a description of input parameters and required coding (before calling certain API procedures).

### A.8. External Session Compiling

Some applications require the use of specific startup parameters in order to compile properly. In these situations, the application's requirements conflict with Roundtable. For example, they may use fonts that Roundtable cannot run with, or a keyword forget list that conflicts with Roundtable.

External Session Compiling allows you to start a separate Progress session that Roundtable will use for compiling your application. When this functionality is enabled, Roundtable populates a table in a separate "compile" Database rather than compiling the programs in its own session. The second Progress session is continuously checking the separate database for work to be done, compiling programs and returning Xref information to Roundtable as needed.

The Roundtable External Session Compiling is implemented using a multi-user database shared by the Roundtable session and an external compilation session. The external

compile session consists of four (4) procedures:

1. `rtb_xcyn.p` - Runs within Roundtable session. Returns logical yes or no to indicate whether or not the object currently selected for compilation should be compiled in an external session.
2. `rtb_xcdb.p` - Runs within Roundtable session. Creates a record in the compile database, thus submitting an object for compilation by the external compile session. The record contains program pathname, compile parameters, and listing file pathnames.
3. `rtb_xcck.p` - Runs within Roundtable session. Checks compile database for completion of compile for a specified object (session ID, workspace, and object). Will timeout with an error message when wait time exceeds specified value. Ends when object has been compiled or timeout value is exceeded.
4. `rtb_xcom.p` - Runs in external compile session. Continuously checks compile database for newly submitted records (objects to be compiled) and compiles them using specified compilation program and compile parameters. Requires a connection to the compile database and any databases required by the object(s) being compiled.

### **A.8.1. Implementing External Session Compiling**

The compile directory of the Roundtable installation directory contains two schema definition (.df) files and the four procedures necessary to support external compilation. The procedures that are run by Roundtable (`rtb_xcyn.p`, `rtb_xcdb.p`, and `rtb_xcck.p`) must remain in this directory. Follow these steps to implement external session compiling:

1. Modify `rtb_xcyn.p` as required for your installation. For example, you can modify `rtb_xcyn.p` to externally compile all objects, all objects in specific modules, or a specific condition that you require.
2. Set the compile program (`Mcomp-prog`) in `rtb_xcom.p` as appropriate for your installation.
3. Copy and modify `rtb_xcom.p` for each unique workspace application database configuration required for your installation (i.e. If each workspace connects to a unique set of databases, you need a unique copy of `rtb_xcom.p` that connects to the specific workspace databases and only compiles objects for that specific workspace).
4. Create a new empty database to be used as the compile database and load the schema definitions provided in `rtbcomp.df` and the sequence definition supplied in `_seqdefs.df`.
5. Start a server for the database created in step 4.
6. For each unique workspace application database configuration, open a client session connected to the server started in step 5, and to any application databases needed to compile objects for that configuration.
7. Run the appropriate copy of `rtb_xcom.p` (created in step 3) in each of the client sessions opened in step 6.

8. When starting a Roundtable session that will use external compilation, connect the session to the server started in step 5.



Databases defined as Roundtable PDBASE objects should be started in multi-user servers to allow both the Roundtable session and the external compile session to connect to those databases. Be sure that the startup parameters in PDBASE objects are appropriate for proper database server connection.

The external compile sessions must be able to access source and listing files using the same pathnames as the Roundtable sessions (e.g. If the source managed by Roundtable is located in `/product1/devel/src/proc.p`, the external compile session must be able to read the source file using `"/product1/devel/src/proc.p")`.

Procedures `rtb_xcyn.p`, `rtb_xcck.p`, `rtb_xcom.p`, and `rtb_xcdb.p` are provided as source code and may be modified, with some latitude, paying attention to preserve input and output parameters (since Roundtable expects them). See the comments in each of the procedures.



# Glossary

---

Alias	An alternate name for a database.
Aliased Object	A PCODE object that has an object name that is different from its physical operating system filename. Object aliases are normally used when two files in different directories have the same filename.
Assigning an Object	To add to a workspace an object that has already been created.
Central Object Repository	Roundtable keeps objects in the Central Object Repository (COR). The COR stores all of the objects and their definitions in the system.
Changes Report	A list of changes in a workspace across a range of events. Provides testers and/or customers with extensive information about what has changed in a workspace between any two events.
Check In	To copy a checked out version of an object into the Central Object Repository. Checking in an object completes the object and allows others to see the changes and use the new version.
Check Out	To create a new version of an object by copying the object and assigning it to a new version number. The new version is used for a specific task and can only be used by one programmer.
Completing a Task	An activity processed once the work is finished in the task. Completing the task gives all the objects in the task a C (complete) status.
Configuration	A map of the contents of your application at a given time.
Configuration Hierarchy	A map of the contents of your application that usually corresponds closely to the architecture of your system.
Configuration Level	Application software systems under configuration control are mapped into a configuration hierarchy. The

	configuration level refers to the depth of the hierarchical tree where an object is found.
CRC	See Cyclical redundancy check.
Cross Reference	A cross reference (Xref) is generated for many kinds of relationships among objects in a workspace. These cross references are generated when a full compile is performed on a compilable object. Database schema objects are also cross referenced automatically by the system.
CRC (Cyclical Redundancy Check)	This is a calculated value that is generally unique for any given piece of data. Roundtable calculates a CRC value for each file loaded into the repository. When required, this CRC value can be compared with a CRC calculation on a file at the operating system level to determine if the repository file and OS file are the same.
Data Type	The Progress data type determines what kind of data can be stored in a variable or database field.
Deployment	Roundtable manages the process of packaging a release of an application system from a workspace for delivery to a remote site. The process of packaging the release is called the deployment process, and the update package created is called a deployment.
Development Workspace	A workspace used to create, modify, and delete new objects and data. Since this workspace is unstable, it is only suitable for limited, informal testing, usually done by the programmers.
DOC	An object that normally contains documentation and is always stored in a type directory called doc off of the objects' workspace module directory. You can also use the PCODE object to store documentation objects. Check out a DOC object to modify the document it contains.
Domains	Occur when you create a single definition that is used in a number of different places. For example, you might define a single PFIELD object called Address and use this field definition in a number of different tables, or even many times in the same table.

Event	Each completed change that results in a change to the contents of a workspace is recorded as a sequential event in the workspace event history.
Event History Report	The event history of a workspace sorted from newest to oldest between two events. Traces the history of individual objects or groups of objects belonging to a product module.
Field Assignment	A PFIELD object contains attributes that describe a field (column) and are referenced by field assignments in PFILE objects. The collection of attributes in the PFIELD object and PFILE field assignment completely define a field in a table. A field assignment contains the product module and name of the PFIELD object, local field name, field order, mandatory status, and triggers of the field.
Force Compile	A compile that occurs even if Roundtable does not think the object needs compiling.
Full Compile	A compile with cross references. Leave blank to just compile and save.
Group Access	You define a collection of security privileges and name them with a group access code. You assign one or more group access codes to one or more users in a workspace. Security privileges accumulate based on each group access code assigned to a user in a workspace.
Group Access Code	A code that identifies a set of security privileges that can be enjoyed within the Roundtable environment. You create access groups and then specify what security privileges belong to the group. You can then associate one or more access groups with one or more users by workspace.
Group Code	User-defined values entered when the user defines objects. The group code is used to sort objects in the object browse table.
Group Directories	Roundtable checks out objects to a task directory where the programmer modifies them. The changes are not seen by others until the programmer executes the Update Group/Public Source option, which copies the modified objects into a group directory. Other

	programmers might choose to see the changes by attaching one of their tasks to the group. The last completed version of the objects remains in the workspace directory.
Listings	Check to print the fully expanded code (with the include files). This is like the Listing parameter in PROGRESS. Roundtable saves the listing file as *.lis file.
Module Load	A Roundtable utility used to search for objects in the operating system directory associated with a selected workspace module. The user is able to quickly create new objects for the new source code.
Object	Any field, table, database, document, or piece of code that you want Roundtable to manage. It is the lowest level of configuration management.
Object Group	User-defined values entered when the user defines objects. The group code is used to sort the objects in the object browse table.
Object Types	All workspaces are comprised of collections of Roundtable objects. These objects are of one of the following types: PCODE: A collection of optimized files whose purpose is specified by assigned subtype PFIELD: A database schema definition for a field PFILE: A database schema definition for a table PDBASE: A database schema definition for a database DOC: A special type used for documentation objects
Object Variants	Variations of the same object in a system.
Object Version	Created each time you check in an object. Object versions are stored in the repository. You can see a list of all versions of an object by selecting the object in a workspace and then choosing View. Object versions are permanent. They are never deleted from the repository.
Orphan	When the same object is checked out in two workspaces at the same time, an object version orphan is created. The term indicates that the changes made in one of the object versions will be lost in some future promotion process among workspaces. For instance, suppose you have a development and test workspace,

that you check out the same object in both workspaces, and then you check in both objects. If you then import the object from the development workspace into the test workspace, the object version imported from development replaces the object version created in test. The changes made in test are lost. Roundtable warns you of object orphan conditions wherever possible.

**Partner Site** A Roundtable installation that receives repository information from another Roundtable installation. You perform a partner site deployment at the sending site and a partner site load at the receiving site to move repository information among sites. Each site must have a different site number.

**PCODE** An object that encapsulates up to nine files that together define a system component. These system components can contain textual or binary data, and you assign a subtype code to each PCODE object to describe these files. These files are called parts and their attributes are specified in the subtype definition. Check out a PCODE object when you need to modify the system component it encapsulates.

**PDBASE** An object that contains the topmost components of a database schema definition. PDBASE (database) objects record a list of table assignments that relate PFILE objects with the PDBASE object. Check out a PDBASE object when you need to add, delete, or change the name of a database table or sequence.

**PFIELD** An object that contains attributes that describe a field (column) and are referenced by field assignments in PFILE objects. The collection of attributes in the PFIELD object and PFILE field assignment completely define a field in a table. A field assignment contains the product module and name of the PFIELD object, local field name, field order, mandatory status, and triggers of the field. The PFIELD object contains the remainder of attributes that define a field in a table.

**PFILE** An object that contains attributes that describe a table. PFILE objects are referenced by table assignments in PDBASE objects. The collection of attributes in the PFILE object and PDBASE table assignment

	completely define a database table. Check out a PFILE object when you need to add, delete, or change a PFILE assignment or an index definition.
Pre-production Workspace	After testing, objects and data are imported from the testing workspace into the pre-production workspace. Since this workspace should function with few or no major problems, you can release it to sophisticated users for Beta testing and evaluations. Normally you would require approval to make changes in this workspace for load testing of the application.
Product	Roundtable provides three logical levels of configuration hierarchy: product, product module, and object. Products own one or more product modules, and product modules own one or more objects. This hierarchy provides a logical view of an application system and is useful for implementing promotion and deployment strategies for the application system.
Product Module	Roundtable provides three logical levels of configuration hierarchy: product, product module, and object. Products own one or more product modules, and product modules own one or more objects. This hierarchy provides a logical view of an application system and is useful for implementing promotion and deployment strategies for the application system.
Production Workspace	A "live" system that must function flawlessly under real-world conditions. You normally require approval to make changes in the workspace.
Promote	The process of importing objects from one or more source workspaces into a target workspace.
PROPATH	A comma-delimited string that contains the operating system directory paths that will be searched by Progress for source code during the compile process.
R-code	Progress-compiled procedures have an extension of .r. Any compiled Progress procedure is referred to as r-code.
Release	A record of a specific event number in a workspace. It is used to re-create the exact contents of the workspace configuration that existed as of that event for purposes of promotion and deployment from a workspace. A

	release labels a version of a workspace.
Release Record	A record that features a sequential release number that uniquely identifies it in the workspace. Release records also contain a text label that describes the configuration of the system at the time the release record was created. For example, release number 23 from the Test workspace might have the descriptive label "Beta Release 2.1a."
Release Report	A detailed list of the changes in a workspace between two releases. Useful for testers and/or customers when receiving new releases.
Remote Site	A site that receives updates packaged by Roundtable.
RTBSETUP File	The Roundtable parameters file that contains workstation-specific options.
Runtask.p	A program used to set up your task environment if you need to work with your task outside of Roundtable.
Run-time, Query, or Development S-code	Progress products types. Progress source code.
Schema Objects	PFILE, PFIELD, and PDBASE objects.
Selective Compile	Provides you with a number of options for your compile. You can enter a specific task to compile or you can enter a 0 to disregard the task number. If you specify a task number, Roundtable compiles only the objects within the selected task that have changed.
Share Status	Checked out objects have a share status of Task, Public, Group, or Central. The share status controls where the object is edited and the visibility of the edits on the object to the rest of the users in a workspace.
Source Workspace	The promotion of code through a development cycle is defined by assigning source workspaces to each workspace in the system. The import process build searches each specified source workspace for objects that should be imported into a selected target workspace. For instance, if you have a development workspace and a testing workspace, you specify that the development workspace is a source workspace for

	the testing workspace. The import process pulls completed object versions included in the latest source workspace release level into the target workspace.
Subtype	Controls the naming, management, creation, and storage of PCODE objects. Up to nine file parts are used to define a system component. These system components can contain textual or binary data. In addition to file parts, you specify edit, naming, and procedure generation services for a subtype, as required.
Syntax Check	A compile that does not create R-code. It ensures the source code will compile without actually compiling.
Table Trigger	A database table might have a table trigger. The trigger specifies the name of a procedure file to execute when a trigger condition occurs. The trigger conditions include: FIND, CREATE, WRITE, and DELETE.
Target Workspace	The destination of objects in an import process. The import process is executed in the target workspace, and objects are imported into it from one or more source workspaces.
Task Directories	Created where checked out objects are copied. This makes it possible to isolate your work from others until it is ready to be checked in or promoted for review into the group or central workspace directories. The use of task directories is optional.
Task Number	The number you specify for a task. If you enter a task number, Roundtable compiles only the objects that have changed in that task. If you enter 0, Roundtable disregards the task numbers and compiles everything that needs compiling.
Task Management	The management of units of work performed on the application system managed by Roundtable. All work done on individual objects in the Roundtable system is performed under a task. Usually, managers assign tasks and programmers check out one or more objects under the task. Completing a task checks in all objects that were checked out under the task. Various task reports are available.



Testing Workspace	Where the application is created after importing objects and data from the development workspace. Although this workspace changes less frequently than the development workspace, some changes are necessary to complete the testing process. This workspace is stable enough for significant quality assurance Alpha testing.
Trigger	An event, such as reading, writing, deleting, or updating a record, that causes another procedure to execute.
Update Status	An attribute of an object's assignment to a workspace. The update status attribute can have a value of Current, Modified, or New. This update status attribute is one of the attributes used to determine if the object needs to be compiled or updated.
Unapplied Changes Report	A list of all schema changes that have been entered but not updated. Verifies the status of the schema in the workspace.
User Access Assignments	Specify what security privileges a user enjoys in a specified workspace. A user access assignment consists of a user paired with a group access privilege in a workspace. Users can be given different security privileges in each workspace.
User Maintenance	The process of adding, editing, or deleting users in the Roundtable system.
Variant	Roundtable allows you to create more than one variation of the same object to create variants. Variants have the same object name and type but belong to different product modules. For example, you might need three variations of an install program, one each for UNIX, DOS, and Windows. Only one of the variants of an object can appear in a workspace at the same time. Variants are often created in custom workspaces that are specific to customers or vertical markets.
Version Control	The ability to manage concurrent changes by multiple programmers by creating new object versions.
WIP	See Work-in-process.

WIP (Work-in-process)	The status of an object when it is checked out or when a task is being worked on.
Workspace	A copy of your application programs and related databases, the content of which is known and managed by Roundtable. Roundtable manages the creation, deletion, and modification of objects (code, databases, tables, fields, and text) within the workspace. A workspace is a configuration of object versions.
Workspace Directories	A workspace has a root directory under which the subdirectories and files that make up the software application exist. The root directory is specified as an absolute path in the Workspace Maintenance screen. Subdirectories are defined as part of workspace module definitions and subtype definitions.
Workspace Event History	Roundtable maintains a record of every change to object versions within a workspace. Roundtable records these changes—called events—in the event history file. Roundtable assigns event numbers to object versions when you complete, assign, or de-assign an object. You can view the event history of a workspace or an object in a workspace at any time.
Workspace Module	The contents of each workspace is broken up into one or more workspace modules. Unlike a file system where many levels of nested subdirectories can be seen, only one level of workspace modules exists. Each workspace module points to a single relative subdirectory under the workspace root directory. This subdirectory is defined in the workspace module definition. The contents of a workspace module are defined by the assignment of workspace module definitions to one or more product modules.
Workspace Module Definition	Names a workspace module, provides a description for the workspace module, and specifies a file system subdirectory relative to any workspace's root directory. You assign a workspace module definition to product modules. You assign product modules to workspaces to define the logical content of your application. Roundtable creates a workspace module and associated subdirectory in the workspace based on the workspace module definition associated with the assigned product module. You specify a product

	<p>module when creating a new object. The object is stored in the subdirectory specified in the workspace module definition, and the object is listed in the workspace module.</p>
Workspace Path	<p>Tells Roundtable the directory where you want to store the workspace and its components. The first workspace path should always be the root directory to your workspace. No two workspaces should share the same root directory. Roundtable uses the other specified paths to locate your source code.</p>
Workspace Releases	<p>Consist of changes or events that have occurred in a workspace since the last release. Roundtable allows you to create two types of releases, a developmental release and a formal release. The developmental release is an internal release distributed to other workspaces within the system, while a formal release is a release distributed to users. Normally, you create releases in conjunction with the quality assurance process.</p>
Workspace Report	<p>A complete list of the objects in a workspace ordered by a workspace module. Provides a hard-copy of the current workspace configuration.</p>
Workspace Source	<p>The import process promotes code from one or more workspaces (source workspaces) into the current (target) workspace.</p>
Workspace Target	<p>The import process promotes code from one or more workspaces (source workspaces) into the current (target) workspace.</p>
Xref Level	<p>The amount of cross reference information maintained in a workspace is defined by the user-specified Xref level. A level of 1 means that no cross reference information will be maintained.</p>

---