



Using PepperTools 8.00.01 Applications PeopleBook

Using PepperTools 8.00.01 Applications PeopleBook

SKU MTUPr8SP1B 1200

PeopleBooks Contributors: Teams from PeopleSoft Product Documentation and Development.

Copyright © 2001 by PeopleSoft, Inc. All rights reserved.

Printed in the United States of America.

All material contained in this documentation is proprietary and confidential to PeopleSoft, Inc. and is protected by copyright laws. No part of this documentation may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, including, but not limited to, electronic, graphic, mechanical, photocopying, recording, or otherwise without the prior written permission of PeopleSoft, Inc.

This documentation is subject to change without notice, and PeopleSoft, Inc. does not warrant that the material contained in this documentation is free of errors. Any errors found in this document should be reported to PeopleSoft, Inc. in writing.

The copyrighted software that accompanies this documentation is licensed for use only in strict accordance with the applicable license agreement which should be read carefully as it governs the terms of use of the software and this documentation, including the disclosure thereof.

PeopleSoft, the PeopleSoft logo, PeopleTools, PS/nVision, PeopleCode, PeopleBooks, and Vantive are registered trademarks, and *PeopleTalk* and "People power the internet." are trademarks of PeopleSoft, Inc. All other company and product names may be trademarks of their respective owners.

Contents

About This PeopleBook

Audience	xiii
Before You Begin.....	xiii
Related Documentation	xiv
Documentation on the Internet.....	xiv
Documentation on CD-ROM	xiv
Hardcopy Documentation	xiv
Typographical Conventions and Visual Cues.....	xv
Comments and Suggestions.....	xvi

Chapter 1

Navigating Through the System

Understanding the PepperTools Application Interface.....	1-1
Using the Keyboard and Mouse.....	1-2
Working with Menus	1-4
Understanding Menu Features	1-4
Selecting a Menu Item.....	1-6
Using Popup Menus	1-6
Canceling a Menu	1-7
Choosing a Menu Command.....	1-8
Working with Pages.....	1-8
Understanding Components	1-8
Understanding Property Sheets	1-9
Working with Page Elements.....	1-9
Using Data Entry Fields.....	1-10
Using Lists	1-10
Using Grids	1-11
Using Scroll Bars	1-12
Using Check Boxes and Radio Buttons	1-13
Using Command Buttons.....	1-13
Using the Main Toolbar.....	1-14
Copying and Moving Data.....	1-15
Understanding Message Boxes.....	1-16

Understanding the Status Bar	1-16
------------------------------------	------

Chapter 2

Using the Data Browser

Accessing the Data Browser	2-1
Working with the Data Browser	2-2
Navigating Data Names	2-2
Viewing Data Instances.....	2-2
Moving and Sorting Data Instance Rows.....	2-3
Using Data Browser Command Buttons	2-3
Exploring the Data Structure	2-4
Understanding the Data Hierarchy	2-4
Understanding Data Instance Columns	2-9
Closing the Data Browser.....	2-12
Using Data Browser Pages	2-12
Understanding PepperTools Pages.....	2-12
Working with Add Pages	2-13
Working with Find Pages.....	2-14

Chapter 3

Using the Supply Chain Viewer

Accessing the Supply Chain Viewer	3-1
Working with the Supply Chain Diagram	3-2
Accessing Vendor and Unit Information	3-2
Exploring the Diagram Toolbar	3-3
Using the Find Unit Page	3-3
Editing the Diagram	3-4
Selecting Nodes.....	3-4
Deselecting Nodes.....	3-5
Modifying Node Position.....	3-5
Modifying Node Size and Shape	3-5
Customizing the Diagram Query	3-6
Filtering Content	3-6
Setting Display Options	3-6
Applying Your Settings.....	3-6
Using the Viewer Command Buttons	3-7

Chapter 4

Saving and Loading Files

Understanding File Saving and Loading	4-1
---	-----

Using the File Browser	4-1
Loading Files.....	4-1
Saving Files.....	4-3
Working with Data Files.....	4-3
Loading a New Dataset.....	4-4
Saving Control panel Settings.....	4-5
Loading Control panel Settings	4-6
Working with Snapshots.....	4-7
Using What-if Snapshots.....	4-7
Using Backup Snapshots.....	4-7
Saving Archive Snapshots.....	4-7
Loading Snapshots	4-8

Chapter 5

Using the Main Menu

Viewing the Menu Before Connecting the Server	5-1
Establishing the Server	5-1
Configuring the Registration Server	5-2
Setting the Catalog Location	5-2
Connecting to the Server	5-2
Viewing the Menu After Connecting the Server	5-3
Viewing the Menu After Loading a Dataset.....	5-4
Using the Home Base Page.....	5-12
Understanding Server User Information.....	5-14
Working with Bookmarks.....	5-14
Understanding the Edit Bookmark Page	5-15
Creating a Bookmark	5-16
Switching Between Bookmarks	5-17
Deleting a Bookmark	5-18

Chapter 6

Using Graphic Charts

Accessing Gantt Charts and Histograms.....	6-1
Exploring a Graphic Chart Page	6-2
Understanding the Chart Toolbar and Timeline.....	6-2
Understanding Chart Types and Features	6-3
Reading Gantt Charts	6-4
Understanding Task Bar Features	6-5
Understanding Task Bar Colors	6-5
Reading Histogram Charts	6-6

Viewing a Resources Histogram.....	6-6
Viewing an Inventory Histogram.....	6-7
Exploring Combination Charts.....	6-8
Accessing an Inventory Report.....	6-8
Accessing a Resource Report.....	6-8
Accessing Task Detail.....	6-9
Navigating Gantt Charts and Histograms	6-9
Navigating with the Gantt Chart Popup Menu.....	6-10
Working with Tasks in a Gantt Chart	6-11
Modifying Tasks with the Mouse.....	6-11
Using the Task Popup Menu	6-12
Expanding and Compressing Tasks	6-13
Viewing Task Intervals	6-14
Freezing and Unfreezing Tasks.....	6-14
Viewing Demand for a Task	6-15
Viewing Supply for a Task	6-16
Viewing Task Detail	6-17
Viewing Task Properties	6-17
Working with a Resource Gantt Chart.....	6-18
Using the Resource Popup Menu	6-19
Cascading and Collapsing Tasks.....	6-19
Dragging Tasks from One Resource to Another	6-20
Using the Histogram Chart Popup Menu.....	6-20
Closing Gantt Charts and Histograms	6-22

Chapter 7

Using Spreadsheets

Accessing Spreadsheets	7-1
Exploring a Spreadsheet Component.....	7-1
Understanding Spreadsheet Pages.....	7-2
Viewing Page Grids	7-3
Working with Spreadsheet Pages	7-4
Using Spreadsheet Page Features.....	7-4
Modifying Date Columns.....	7-5
Setting a Start Date.....	7-5
Setting the Number of Periods	7-5
Setting the Period Duration.....	7-5
Choosing Rows to Display	7-5
Adding a Row Type	7-6
Deleting a Row Type	7-9

Setting the Number of Rows	7-10
Managing Sorting Behavior	7-10
Changing the Sort Order	7-10
Sorting in Ascending or Descending Order	7-11
Enabling and Disabling Sorting	7-11
Setting Filter Criteria.....	7-12
Applying Basic Filtering	7-12
Applying Advanced Filtering.....	7-13
Adding Summary Information	7-14

Chapter 8

Using the Class Editor

Understanding the Class Editor Component.....	8-1
Understanding the Class Editor Header	8-2
Understanding the Class Slot Pages	8-2
Viewing and Editing a Class.....	8-3
Understanding the Add SubClass Page	8-5
Adding a Subclass.....	8-7

Chapter 9

Editing and Creating Pages with the Form Painter

Installing and Starting the Form Painter	9-1
Creating a Page	9-2
Opening a Page	9-4
Saving a Page.....	9-4
Editing Page Properties.....	9-5
Editing the Topic, Default Buttons, and Label on a Page	9-5
Making the Page Editable or Non-editable	9-7
Creating and Editing Components	9-8
Adding Controls to a Page	9-9
Deleting Controls	9-10
Editing Controls.....	9-11
Making a Control Editable or Non-editable	9-11
Editing Properties for a Control	9-11
Adding Columns to Grid and Spreadsheet Controls	9-13
Editing Columns.....	9-13
Editing Timeline (CTC) Controls	9-15
Moving, Aligning and Resizing Controls	9-16

Chapter 10

Using the Resource Language to Modify Pages

Understanding the Unit Page Code.....	10-1
Understanding the Unit Page Popup Menu Code	10-5
Understanding the Unit Set Menu	10-6
Understanding the Unit Instance Menu.....	10-7
Using the Resource Language	10-7
Populating Pages and Controls	10-8
Populating a Page with a Topic.....	10-8
Populating a Control with a Topic and Value	10-9
Displaying Pages and Menus.....	10-9
Displaying Pages	10-10
Displaying Menus	10-10
Displaying Set Menus	10-10
Displaying Instance Menus	10-10
Writing a Resource Language File	10-11
Building Resource Language Files	10-11
Form Files Contain Form Pages.....	10-11
Class Files Contain Property Sheet Pages and Menus	10-12
Modifying the Catalog File	10-13
Writing Resource Language Statements.....	10-14
Writing Statement Properties	10-14
Writing Statement Events.....	10-24
Writing Event Commands.....	10-26
Writing Methods	10-31
Writing Method Parameters	10-34
Writing Menu Statements	10-35
Menus: BEGINMENU, ENDMENU	10-35
Submenus: BEGINSUBMENU, ENDSUBMENU	10-36
Menu Items: MENUITEM.....	10-36
SEPARATOR	10-37
Writing Statements That Create Pages	10-37
Form Page: BEGINFORM, ENDFORM	10-37
Property Sheet Page: BEGINPROPSHEET, ENDPROPSHEET	10-39
Writing Control Statements That Contain Other Controls	10-40
Coordinated Time Control Widget: BEGINCTC, ENDCTC.....	10-40
TIME_FENCE	10-44
Grids: BEGINGRID, ENDGRID	10-46
Pages or Components: BEGINPAGE, ENDPAGE.....	10-49
Spreadsheets: BEGINSPREADSHEET, ENDSPREADSHEET	10-51

Writing Control Statements	10-55
BARCHART	10-55
BITMAP.....	10-59
Calendar inside a CTC: CALENDAR_CONTROL.....	10-61
Capacity Histogram inside a CTC: CAPACITY_HISTOGRAM.....	10-62
CHECKBOX.....	10-64
Columns in a Grid or Spreadsheet: COLUMN	10-67
Drop-down List: COMBOBOX	10-71
CONTROLSLIDER.....	10-74
Field to Enter Data: EDIT	10-77
Grouping Controls Together: GROUPBOX	10-81
Inventory Histogram inside a CTC: INVENTORY_HISTOGRAM	10-82
PENALTY.....	10-84
Buttons: PUSHBUTTON.....	10-87
Resource Gantt inside a CTC: RESOURCE_GANTT.....	10-89
Non-editable Text Strings: STATIC	10-93
Task Gantt inside a CTC: TASK_GANTT	10-95
Writing a Data Browser	10-99
BEGINBROWSER, ENDBROWSER.....	10-100
BEGINFOLDER, ENDFOLDER	10-101
Writing a Diagramming Control.....	10-103
BEGINDIAGRAM, ENDDIAGRAM	10-103
DRAWINGAREA.....	10-106
NODE.....	10-108
ARC.....	10-110
Node and Arc Instance Menus: BEGINNODEMENU and BEGINARCSMENU	10-112
BEGINNODEMENU, ENDMENU.....	10-112
BEGINARCSMENU, ENDMENU	10-113
BEGINDIAGRAMMENU, ENDMENU	10-114
Understanding Examples Of Resource Language Code.....	10-114
Understanding the Add Unit Page Code	10-115
Understanding the Build Schedule (with Resource) Spreadsheet Page Code.....	10-120
Understanding the Control panel Code	10-135
Understanding the Score Card Page Code	10-138
Understanding the Purchase Order Page Code.....	10-143
Understanding the Inventory Report Page Code.....	10-154
Understanding the About Box Page Code.....	10-156
Understanding the Item Filter Page Code	10-158
Understanding the Data Browser Page Code	10-167

Understanding the Supply Chain Viewer Page Code.....	10-170
Understanding Pages Formatting Standards	10-175
Font	10-175
Display Target.....	10-175
Regional Settings Time Style.....	10-175
Text Labels.....	10-175
Edit Fields	10-175
Using Standard Pages Transactions	10-178
Getting Data with Page Transactions	10-179
form_get_slot_instance	10-179
form_get_topic_instance	10-180
form_get_topic_class	10-181
form_get_slot_class.....	10-181
form_get_slot_string_class	10-182
form_get_class_of_instance.....	10-182
form_get_slot_boolean.....	10-183
form_get_subclasses_of_class	10-183
form_get_instances_of_class	10-184
form_get_enums_of_class.....	10-185
get_project_times	10-185
Setting Data with Page Transactions.....	10-186
form_set_slot_int	10-186
form_set_slot_float	10-187
form_set_slot_string.....	10-187
form_set_slot_date	10-187
form_set_slot_time.....	10-188
form_set_slot_instance.....	10-188
form_set_slot_boolean	10-189
Page Transactions that Perform Other Functions.....	10-189
form_check_topic_instance.....	10-189
form_sort_slot_instance_list	10-190
Arrays and Vectors	10-191

Chapter 11

Using the Visual Browser

Installing Tcl/Tk (Windows NT only).....	11-1
Running the Visual Browser.....	11-1
Using the Visual Browser Menus	11-7
Using the SPL File Entity List Page Buttons	11-11
Using the Action and Read-only Action Entity List Buttons	11-11

Using the Action Schema and Read-only Action Schema Entity List Buttons....	11-12
Using the Class Entity List Buttons	11-12
Using the Enum Entity List Buttons	11-13
Using the C++ Function Entity List Buttons.....	11-13

Chapter 12

Writing Queries

Understanding Query Syntax.....	12-1
Writing a Query class list	12-2
Writing a Query selection criteria.....	12-2
Using Query Expressions	12-2
Using Query Predicates.....	12-3
Using Relational Operators	12-3
Using Instance Operator Syntax.....	12-5
Using Arithmetic Operators	12-5
Using Parentheses To Enforce Operator Precedence	12-6
Using Conversion Operators	12-6
Using Oset Operators	12-6
Using Expression Comparisons	12-6
Nesting Select Statements	12-7
Writing a Query Ordering Clause.....	12-8
Using order randomly.....	12-9
Using order by	12-9
Arrays and Vectors	12-10

Index

ABOUT THIS PEOPLEBOOK

This PeopleBook provides you with information about using the components and interface of your PepperTools application.

Audience

You should be familiar with navigating around the system and adding, updating, and deleting information using PeopleSoft windows, menus, and pages. You should also be comfortable using the Microsoft® Windows 95 or Windows NT graphical user interface.

Because we assume you already know how to navigate around the PeopleSoft system, much of the information in this book is not procedural. That is, it does not typically provide step-by-step instructions on using tables, pages, and menus. Instead we provide you with all the information you need to use the system most effectively, and to customize the documentation to your organizational or departmental needs. This book expands on the material covered in PeopleSoft training classes.

Navigating Through the System describes the basics of interacting with the PepperTools applications' graphical user interface.

Using the Data Browser introduces the content, and describes the use of the Data Browser.

Using the Supply Chain Viewer describes the use of the Supply Chain Viewer.

Saving and Loading Files describes how to save and load data files, snapshots, and control panel settings.

Using the Main Menu describes the architecture of the PepperTools Main Menu.

Using Graphic Charts describes how to use Gantt charts and histogram charts to view inventories, capacity loading, and task scheduling.

Using Spreadsheets describes how to display, modify and interact with data in spreadsheets.

This section describes information you should know before you begin working with PeopleSoft products and documentation, including PeopleSoft-specific documentation conventions, information specific to PeopleTools, how to order additional copies of our documentation, and so on.

Before You Begin

To benefit fully from the information covered in this book, you need to have a basic understanding of how to use PeopleSoft applications. We recommend that you complete at least one PeopleSoft introductory training course.

You should be familiar with navigating around the system and adding, updating, and deleting information using PeopleSoft windows, menus, and pages. You should also be comfortable using the World Wide Web and the Microsoft® Windows or Windows NT graphical user interface.

Related Documentation

To add to your knowledge of PeopleSoft applications and tools, you may want to refer to the documentation of the specific PeopleSoft applications your company uses. You can access additional documentation for this release from PeopleSoft Customer Connection (www.peoplesoft.com). We post updates and other items on Customer Connection, as well. In addition, documentation for this release is available on CD-ROM and in hard copy.



Important! Before upgrading, it is *imperative* that you check PeopleSoft Customer Connection for updates to the upgrade instructions. We continually post updates as we refine the upgrade process.

Documentation on the Internet

You can order printed, bound versions of the complete PeopleSoft documentation delivered on your PeopleBooks CD-ROM. You can order additional copies of the PeopleBooks CDs through the Documentation section of the PeopleSoft Customer Connection Web site: <http://www.peoplesoft.com/>

You'll also find updates to the documentation for this and previous releases on Customer Connection. Through the Documentation section of Customer Connection, you can download files to add to your PeopleBook library. You'll find a variety of useful and timely materials, including updates to the full PeopleSoft documentation delivered on your PeopleBooks CD.

Documentation on CD-ROM

Complete documentation for this PeopleTools release is provided in HTML format on the PeopleTools PeopleBooks CD-ROM. The documentation for the PeopleSoft applications you have purchased appears on a separate PeopleBooks CD for the product line.

Hardcopy Documentation

To order printed, bound volumes of the complete PeopleSoft documentation delivered on your PeopleBooks CD-ROM, visit the PeopleSoft Press Web site from the Documentation section of PeopleSoft Customer Connection. The PeopleSoft Press Web site is a joint venture between PeopleSoft and Consolidated Publications Incorporated (CPI), our book print vendor.

We make printed documentation for each major release available shortly after the software is first shipped. Customers and partners can order printed PeopleSoft documentation using any of the following methods:

Internet

From the main PeopleSoft Internet site, go to the Documentation section of Customer Connection. You can find order information under the Ordering PeopleBooks topic. Use a Customer Connection ID, credit card, or purchase order to place your order.

PeopleSoft Internet site: <http://www.peoplesoft.com/>.

Telephone

Contact Consolidated Publishing Incorporated (CPI) at **800 888 3559**.

Email

Email CPI at callcenter@conpub.com.

Typographical Conventions and Visual Cues

To help you locate and interpret information, we use a number of standard conventions in our online documentation.

Please take a moment to review the following typographical cues:

`monospace font`

Indicates PeopleCode.

Bold

Indicates field names and other page elements, such as buttons and group box labels, when these elements are documented below the page on which they appear. When we refer to these elements elsewhere in the documentation, we set them in Normal style (not in bold).

We also use boldface when we refer to navigational paths, menu names, or process actions (such as **Save** and **Run**).

Italics

Indicates a PeopleSoft or other book-length publication. We also use italics for *emphasis* and to indicate specific field values. When we cite a field value under the page on which it appears, we use this style: ***field value***.

We also use italics when we refer to words as words or letters as letters, as in the following: Enter the number *0*, not the letter *O*.

KEY+KEY

Indicates a key combination action. For example, a plus sign (+) between keys means that you must hold down the first key while you press the second key. For ALT+W, hold down the ALT key while you press W.

Jump links	Indicates a jump (also called a link, hyperlink, or hypertext link). Click a jump to move to the jump destination or referenced section.
Cross-references	The phrase For more information indicates where you can find additional documentation on the topic at hand. We include the navigational path to the referenced topic, separated by colons (:). Capitalized titles in <i>italics</i> indicate the title of a PeopleBook; capitalized titles in normal font refer to sections and specific topics within the PeopleBook. Cross-references typically begin with a jump link. Here's an example:

For more information, see Documentation on CD-ROM in *About These PeopleBooks*: Related Documentation.

- Topic list
- Contains jump links to all the topics in the section. Note that these correspond to the heading levels you'll find in the Contents window.



Name of Page or
Dialog Box

Opens a pop-up window that contains the named page or dialog box. Click the icon to display the image. Some screen shots may also appear inline (directly in the text).



Text in this bar indicates information that you should pay particular attention to as you work with your PeopleSoft system. If the note is preceded by **Important!**, the note is crucial and includes information that concerns what you need to do for the system to function properly.



Text in this bar indicates For more information cross-references to related or additional information.



Text within this bar outlined in red indicates a crucial configuration consideration. Pay very close attention to these warning messages.

Comments and Suggestions

Your comments are important to us. We encourage you to tell us what you like, or what you would like changed about our documentation, PeopleBooks, and other PeopleSoft reference and training materials. Please send your suggestions to:

PeopleTools Product Documentation Manager
PeopleSoft, Inc.
4460 Hacienda Drive
Pleasanton, CA 94588

Or send comments by email to the authors of the PeopleSoft documentation at:

DOC@PEOPLESOFT.COM

While we cannot guarantee to answer every email message, we will pay careful attention to your comments and suggestions. We are always improving our product communications for you.

CHAPTER 1

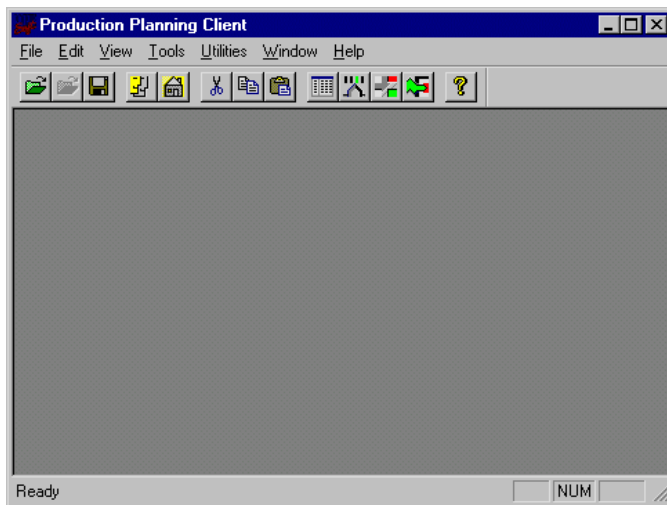
Navigating Through the System

This documentation familiarizes you with the graphical user interface (GUI) common to the various PepperTools client applications, including Enterprise Planning, Production Planning and Order Promising.

Understanding the PepperTools Application Interface

The PepperTools client applications are Enterprise Planning, Production Planning and Order Promising. All three use the same primary interface; any differences between the basic features of the applications will be noted below the section or topic heading for the feature being discussed.

Each PepperTools application appears as a program window that contains the Main Menu, the toolbar and the status bar.



Main Production Planning Interface

While the Main Menu is the most comprehensive means for accessing most windows, the program also has a Home Base page with buttons that invoke selected Main Menu commands and provide a more visibly accessible way to navigate.



For more information about the Main Menu and Home Base, see [Using the Main Menu](#).

Using the Keyboard and Mouse

Working with a PepperTools application consists of three primary activities, which may overlap:

- Navigating the program interface
- Selecting data for manipulation
- Invoking a feature of the program

You can use either the keyboard or the mouse for any of these activities, and switch between the two devices at will to suit your preference. You'll find the keyboard easier for some actions, and the mouse easier for others. Methods for using the keyboard and mouse (applicable to all three activities) are outlined in the following table, but specific uses for those methods are covered under the appropriate topics in this PeopleBook.



The mouse is currently the most popular computer pointing device, but any reference in this PeopleBook to the mouse applies similarly to alternate devices such as a trackball or a touchpad.

Keyboard and Mouse Methods

Device—Method	Input Focus*	Description
Keyboard—Selection bar	Menus, lists, spreadsheet cells	The selection bar is highlighting that indicates the currently selected item. Use the arrow keys to move the selection bar within a list or menu, or between spreadsheet cells.
Keyboard—Shift selection	Text, lists, spreadsheet cells	Used for selecting text, or multiple fields or cells. Hold down the Shift key as you move the selection bar or cursor.
Keyboard—Access key	Menus, most pages	An alphanumeric keyboard equivalent for menu access. Press Alt + the underlined letter in a menu name to display that menu; type the underlined letter in a visible menu item to invoke that item.

Keyboard—Shortcut	Text, most pages	A key combination that directly invokes a program function without having to navigate to a menu item or button. Some shortcuts may not have an equivalent menu item.
Mouse—Click an item	Anywhere	Multiple uses. Point, press and release the left (primary) mouse button. If nothing else, this may change the input focus.
Mouse—Double-click an item	Text, spreadsheet cells, designated controls	Multiple uses. Point, press and release the left mouse button twice in quick succession.
Mouse—Right-click an item	Designated fields, cells and controls	Used to access a popup menu. Point, press, and release the right mouse button to view the popup menu, then left-click to select the desired option from the menu.
Mouse—Drag an item	Supply Chain Viewer nodes	Used to reposition nodes in the Supply Chain Viewer diagram. Point, press and hold the left mouse button down, and keep it down as you move the pointer, dragging the node to its new location.
Mouse—Drag-selection	Text, Supply Chain Viewer nodes	Used to select text and objects. Point, press, and hold the left mouse button down, then keep it down as you move the pointer, highlighting the material you want to select.
Keyboard + Mouse—Shift-selection	Text, lists, spreadsheet cells	Used to select text, or multiple list items or cells. With the selection bar or cursor at the beginning of the desired selection, hold down the Shift key as you click at the desired endpoint.

Keyboard + Mouse—Control-selection	Lists, spreadsheet cells, Supply Chain Viewer nodes	Used to select noncontiguous multiple list items or cells. With the selection bar or cursor at the first desired item, hold down the Ctrl key as you click each additional desired item. Note —use the Shift key in the Supply Chain Viewer diagram.
Keyboard + Mouse—Drag an item	Designated grid columns	Used to reorder sort key columns. With the column selected, hold down the Shift key, then point at the column heading, press and hold the left mouse button down, and keep it down as you move the pointer, dragging the column to its new location.

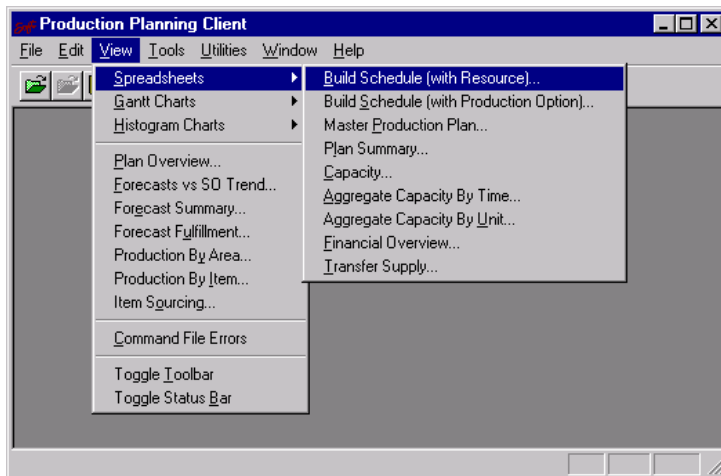
* The input focus is the point within the program currently ready to accept input from you. The appearance of the input focus varies by context; in text, it appears as a text cursor or insertion point; on a menu, list or field, it's a selection bar (a highlighted item).

Working with Menus

Your PepperTools application makes most of its operations available with menu commands. You select a menu, then select a command from that menu to carry out an operation.

Understanding Menu Features

The program's primary display includes a menu bar called the Main Menu just below the title bar. The Main Menu consists of several drop-down menus, which contain a combination of submenus and commands with which you operate the program.



Drop-down Menu and Submenu



For more information about Main Menu features specific to PepperTools, see Using the Main Menu.

PepperTools application menu features include:

Drop-down menu

A drop-down menu is a list of options that drops down from the Main Menu. These options can be commands or submenus.

Submenu

A submenu is a secondary menu that contains more commands, connected to the current menu. If a triangular arrow displays next to a menu item, that item is the name of a submenu which will appear when you select it.

Popup menu

Popup menus provide a convenient way for you to access commands appropriate to certain data and control objects, usually by right-clicking the object.

Access key

The access key is an alphanumeric keyboard equivalent that opens a menu or submenu, or invokes a command. Press **Alt** + the underlined letter in a menu name to display that menu. When a menu is open, simply type the underlined letter in a visible menu item to invoke that command. The access keys are sometimes referred to as mnemonics.

Dimmed menu item

A dimmed menu item shows as gray (rather than black) text; it means that the item isn't currently available.

Ellipsis

An ellipsis (...) displays after a menu command if the command requires additional information to complete its execution. When you select the menu item, a page or dialog box opens so that you can type the required information.

Selecting a Menu Item

To select a menu item using the mouse, point to its name on the Main Menu and click the name to open it. You can drag the selection cursor down the menu if you want to move to a menu item immediately.

To select a menu item from the Main Menu using the keyboard:

1. Press **Alt** to select the menu bar.
2. Use the left arrow or right arrow key to select the menu you want.
3. Press **Enter** to open the selected menu.



When a Main Menu drop-down menu name has an access key (underlined letter), you can press **Alt** and the letter to open that menu.

4. Use the up arrow or down arrow key to select the menu item you want.



When an item name has an access key, you can type its underlined letter to invoke it.

Using Popup Menus

A popup menu displays at the mouse pointer's current location so it eliminates the need for you to move the pointer to the menu bar or toolbar. Popup menus only contain commands that apply to the selected object and its context.

	Unit*	Resource*	Resource Class
1	M04	03_CREW	03_DET_C_class
2	M04	04_CREW	04_DET_C_class
3	M04	CSD1_TOOL	CSD1_DET_T_class
4	M04	DTA_CREW	DTA_DET_C_class
5	M04	DTT_CREW	DTT_DET_C_class
6	M04	EST_MACH	DT-TEST_DET_M_class
7	M04	MACH	ICT1_DET_M_class
8	M04	NBA_CREW	NBA_DET_C_class
9	M04	NBT_CREW	NBT_DET_C_class
10	M04	Center_DAC	DAC_WC_Resource
11	M04	WorkCenter_DTC	DTC_WC_Resource
12	M04	WorkCenter_MA1	MA1_WC_Resource
13	M04	WorkCenter_MQ1	MQ1_WC_Resource
14	M04	WorkCenter_NAC	NAC_WC_Resource

Popup Menu

To open popup menus using a mouse, do one of the following:

- Right-click a row number or a column heading in any grid.
- Right-click in the upper left corner of any grid.
- Right-click any grid cell.
- Right-click any data item in the left page of the Data Browser.
- Right-click any node in the Supply Chain Viewer.

The commands available on popup menus depend entirely on the context. Some of the more common commands are:

- **Copy**, to copy the data for use in another location or application.
- **Add**, to invoke the Add page.
- **Find**, to invoke the Find page.
- **Properties**, to display a property sheet.



For more information about Add pages and Find pages, see *Using the Data Browser: Using Data Browser Pages*. For more information about property sheets, see *Working with Pages: Understanding Property Sheets* in this section.

Canceling a Menu

To cancel a menu, you can:

- Click anywhere outside the menu.
- Press **Alt** to move back to the application workspace.

- Press **Esc** to remain on the menu bar so you can select another menu.

Choosing a Menu Command

The items listed on the menus are commands your PepperTools application can execute. To choose a menu item from any displayed menu, move the mouse pointer along the menu; a selection bar highlights the entries. Click your choice to invoke it. You can also use the up and down arrow keys to select the item you want, then press **Enter**, or simply use its access key (type the underlined letter in its name).

Working with Pages

A page is a clearly defined area in a window, containing the controls you use to interact with your PepperTools application. Pages are used throughout the interface as a means of acquiring input from you, prompting you for decisions, enabling you to make settings, and presenting output. You can maximize or minimize each page using the standard Windows control buttons.

Understanding Components

A component consists of several pages within the same window. Only one page is visible at a time—the names of the other available pages are displayed on "tabs" just above the page area. By clicking a page name, you bring that page to the front, and it becomes the focus of your interaction with the program.

[illegible]

Component

Many components include a separate area, either just above the pages or in the top portion of the pages themselves. This area, the **page header**, stays the same no matter which page is currently in use, and contains common information and controls applicable to all the pages in the component.

Understanding Property Sheets

A property sheet is a page or component that displays an object's properties, for example, item properties, unit properties, or resource properties.

Double-click a data cell in a grid, or select **Properties** from its popup menu, to view the associated property sheet. You can view and edit information on property sheets just like any other page; in fact, many objects in a property sheet have their own property sheets.

Inventory Item Property Sheet



For more information about popup menus, see Working with Menus: Using Popup Menus in this section.

Working with Page Elements

The controls and other elements on a page can include data entry fields, lists, grids, scroll bars, check boxes, radio buttons and command buttons.



For more information about command buttons, see Using Command Buttons in this section.

Using Data Entry Fields

A data entry field is a rectangular area for entering information—a text box. When you click an empty data entry field, a cursor (flashing vertical bar) appears at the far left side of the field. The text you type starts at the cursor. If the field already contains text when you move to it, all the text in the field is automatically selected and any text you type or paste replaces it. If you use the mouse or the arrow keys to reposition the cursor, the text is deselected. When you type new text, it's inserted into the existing text at the cursor location if your keyboard is in the normal Insert mode. If you use the **Ins** key to toggle to the Overwrite mode, each character typed replaces an existing character.

To select text in a data entry field using the mouse, drag the pointer across the text you want to select. You can also double click to select one word at a time, and drag the pointer to extend that selection to multiple words.



Any data entry field containing blue text is read-only—you cannot alter the value displayed.



For more information about manipulating the contents of a data entry field, see Copying and Moving Data in this section.

Using Lists

A list, like a menu, displays a vertical array of choices. Unlike a menu, the contents of a list are parameters, options or data items rather than commands, and the list remains visible at a fixed location within a page. Lists come in two forms—**drop-down lists** and **list boxes**.

- A drop-down list looks like a data entry field with a button at one end. Click the button, and a list of choices drops down; you can only choose one item, which will be the only visible item when the list retracts again.
- A list box is list of choices in a fixed rectangular display area; it generally enables you to make multiple choices.



If the list contains more items than it has room for, a vertical scroll bar enables you to see the entire list.

To select a single item from a list, click the scroll arrows until your choice appears in the list, or type the item's first letter. The list displays that portion of the list. Click the item you want.



For more information about selecting multiple list items, see *Using the Keyboard and Mouse* in this section.

Using Grids

A grid displays information in tabular form, with data cells arranged in columns and rows. Grids appear in the Data Browser, on spreadsheets and on Find pages.

	Unit*	Item*	Description*	Planner Cc
1	M04	20001	Multimedia 3000 Desktop System/QOH:	
2	M04	20100_CFG:PC5200 BLUN4YNYYY	Configured Desktop PC /QOH: 6	Aw
3	M04	20100_CFG:PC5300 BLUN4YYYYN	Configured Desktop PC /QOH: 0	Aw
4	M04	20100_CFG:PC5300 BLUY4YYYYY	Configured Desktop PC /QOH: 0	Aw
5	M04	20100_CFG:PC5300 BLUY6YYYYY	Configured Desktop PC /QOH: 1	Aw
6	M04	20100_CFG:PC5300 GRYN4YYYYY	Configured Desktop PC /QOH: 1	Aw
7	M04	20100_CFG:PC5300 GRYN5YYYYY	Configured Desktop PC /QOH: 5	Aw
8	M04	20100_CFG:PC5300 GRYY6YYYYY	Configured Desktop PC /QOH: 5	Aw
9	M04	20100_CFG:PC5300 WHTN4YNYYN	Configured Desktop PC /QOH: 5	Aw
10	M04	20100_CFG:PC5300 WHTN4YYYYY	Configured Desktop PC /QOH: 10	Aw
11	M04	20100_CFG:PC5300 WHTY4YYYYY	Configured Desktop PC /QOH: 7	Aw
12	M04	40062	32 MB Memory Module /QOH: 13	Aw
13	M04	40100_CFG:P2 233MHZ 64:4	Configured CPU /QOH: 0	Aw
14	M04	40100_CFG:P2 233MHZ 64:8	Configured CPU /QOH: 0	Aw
15	M04	40100_CFG:P2 266MHZ 128:8	Configured CPU /QOH: 0	Aw
16	M04	40100_CFG:P2 300MHZ 64:4	Configured CPU /QOH: 0	Aw

Data Browser Grid



Any cell containing blue text is read-only—you cannot alter the value displayed.



Certain grid cells have an associated popup menu with commands appropriate to the data in the cell. Right-click a grid cell to display the popup menu.



For more information about popup menus, see *Working with Menus: Using Popup Menus* in this section.

Changing Column Width

You can change the width of the columns in a grid.

To change the column widths:

1. Move the mouse pointer to the heading of the column whose width you want to change. Position the pointer at the right edge of the heading. The mouse pointer changes to a double-arrow adjustment symbol.

2. Hold down the mouse button and drag left or right to the width you desire.
3. Release the mouse button.



You can temporarily remove a column from view by dragging the double-arrow symbol as far left as possible—it will stop just before the column's left edge. When you release the mouse button, the column appears to be a thin double line. You can restore it by dragging its right edge back out to your desired width.

Sorting Rows of Data

You can move and sort rows in a grid. Sorts execute in ascending order, using columns denoted with an asterisk (*) as keys, prioritized from left to right. The program sorts on the first column that contains an asterisk in the column name, then on the second column with an asterisk in the column name, and so forth. You can change this sorting sequence by repositioning the key columns.

To change the sorting sequence of key grid columns, complete the following steps:

1. Click a column heading that includes an asterisk (*) in the name. The column is highlighted.
2. Press the Shift key, then click and hold down the left mouse button on the heading. The cursor changes to an arrow with a box attached to its tail. Drag the arrow to your desired column position and release the mouse button. The data column moves to that location.
3. Click the Sort button to sort the data according to the new sequence.

Using Scroll Bars

A scroll bar is a thin rectangular control along the bottom or the right side of a display area, consisting of several components: The large rectangle is the scroll box container, with the square scroll box inside it. At either end of the container are two square buttons with arrows on them.



Horizontal Scroll Bar

When you're working with a page that displays Gantt charts, histograms, spreadsheets or lists, there's often more information than can be displayed at one time. If this is the case, the display area serves as a window into the data, and scroll bars appear so you can navigate to the rest of the data.

Click the appropriate scroll bar arrow button to shift the desired data into view in small increments. You can also click and drag the scroll box to shift large distances, or click the container next to the box to shift by the width or height of the display window.



For more information about Gantt charts and histograms, see *Using Graphic Charts*. For more information about spreadsheets, see *Using Spreadsheets*.

Using Check Boxes and Radio Buttons

A check box is a small labeled square just large enough to contain a checkmark. You use it to turn a single option on or off—click the box to produce a checkmark, and the option is turned on; click it again to remove the checkmark, and the option is turned off. A page can contain one or more check boxes; a group of them often applies to a common task, but each check box can always be set independently of the others.

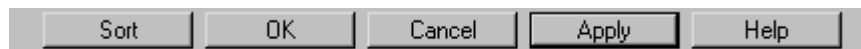


Some grid cells contain **X boxes**, which work the same way as check boxes, except that instead of a checkmark, you'll see an 'X' in the box when that option is turned on. These options use their column headings as their labels.

A radio button is a small labeled circle which can contain a black dot—when the dot is present, the option is selected. Radio buttons always come in groups, in which the buttons represent a set of mutually exclusive options. Exactly one radio button in a group is selected at any time. To change the selection, click the circle for the option you want; the dot appears in that circle, and is cleared from the others.

Using Command Buttons

A command button, sometimes referred to as a pushbutton, is a three-dimensional rectangle containing a label or a picture. When you click the button, the action depicted on it is executed. The PepperTools toolbar consists of command buttons appropriate to the program as a whole, rather than a specific page.



Command Buttons



For more information about the PepperTools toolbar, see *Using the Main Toolbar* in this section.

The following standard command buttons appear at the bottom of most pages:

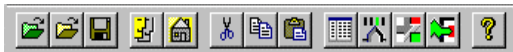
Standard Command Buttons

Command	Description
OK	Apply all changes and close the page.

Cancel	Ignore any changes and close the page.
Apply	Apply all changes but leave the page open.
Help	Provide access to help information.
Sort	Sort the rows of data in the page (appears only where sorting is appropriate).

Using the Main Toolbar








The PepperTools main toolbar is a strip of several buttons that provide quick access to commonly used commands. It initially appears just below the Main Menu, but you can reposition it according to your preference. Simply click anywhere in the space between the buttons, and drag the toolbar to any of the four sides of the application window, where it will snap into position when you release the mouse button. You can also release it in the middle of the window to use it as an independent "floating palette."









PepperTools Main Toolbar

To execute a command, click the appropriate toolbar button. The toolbar contains the following commands:

PepperTools Toolbar Buttons

Button	Menu Equivalent	Description
	File, Load Command File	Loads a command file from the current server.
	File, Load Snapshot	Loads a snapshot from the current server.
	File, Save Snapshot	Saves a new snapshot to the current server.
	File, Show Server Users	Shows users connected to the current server.
	Tools, Home Base	Opens Home Base page for easier navigation (Note—Home Base isn't part of the Order Promising application).
	Edit, Cut	Moves the selection to the clipboard.
	Edit, Copy	Copies the selection to the clipboard.

	Edit, Paste	Inserts the clipboard contents at the insertion point.
	Tools, Data Browser	Views data model.
	Tools, Control panel	Sets control panel options.
	Tools, Scorecard	Views exceptions in current schedule.
	Tools, Optimize	Opens optimization control window.
	Help	Invokes context-sensitive online documentation.

Copying and Moving Data

You can export text from your PepperTools application to another program, import text from another program to PepperTools, or transfer data between two different areas of your PepperTools application. You accomplish this with the **Cut**, **Copy** and **Paste** commands by using the Edit menu, keyboard shortcuts, or right-clicking highlighted data to display a popup menu. You can also use the appropriate toolbar buttons.



Any use of Cut, Copy or Paste in another program depends on whether that program makes the feature available. That program may not use the same method as PepperTools to invoke the feature.

To copy or move data, complete the following steps:

1. Select the source data you want to transfer.
2. Copy or cut the selected data:
 - Select **Edit, Copy**, or click the toolbar **Copy** button. Copy duplicates the selected data and transfers it to the system clipboard.
 - Select **Edit, Cut**, or click the toolbar **Cut** button. Cut duplicates the selected data, transfers it to the system clipboard, and deletes it from its original location.



You can also press the shortcut **Ctrl+C** to copy data, or **Ctrl+X** to cut data.

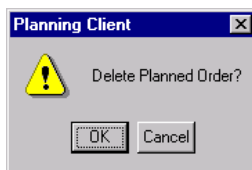
3. Navigate to the destination, and click where you want the data to appear. If you want to replace existing data, select the data to be replaced.
4. Select **Edit, Paste**, or the toolbar **Paste** button. Paste transfers the material currently in the system clipboard to the selected destination.



You can also press the shortcut **Ctrl+V** to paste data. A copy of the pasted material remains in the system clipboard, and can be reused until it's replaced by a new cut or copy action.

Understanding Message Boxes

A message box is a secondary window that provides you with information and prompts you for confirmation. Each one has a title specifying the current context, a control symbol indicating the reason it has appeared, an informational message or a question, and one or more command buttons. The program won't allow you to continue working until you click one of the buttons to indicate how you would like to proceed.



Message Box

Understanding the Status Bar

Your PepperTools application displays a status bar at the bottom of the main window which offers information about the current state of the application and your activities. The three boxes on the right indicate for convenience the state of three keyboard settings: Caps Lock, Num Lock and Scroll Lock.



Status Bar

CHAPTER 2

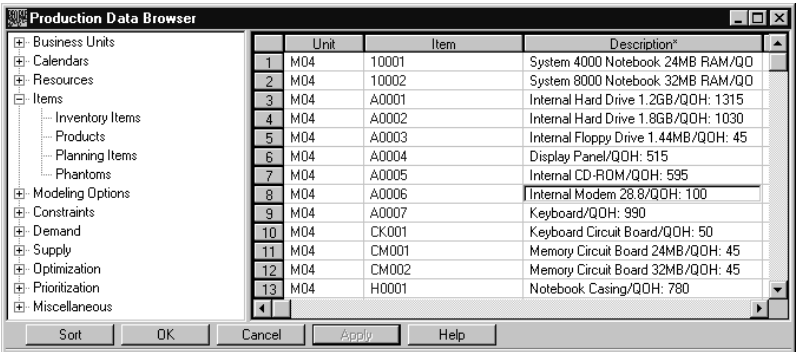
Using the Data Browser

The PepperTools Data Browser enables you to display and edit supply, demand, and facility model data that you've loaded via the DataBridge process.

Accessing the Data Browser

You can access the Data Browser in several ways:

- Select Tools, Data Browser.
- Click the **Browse Data** button on the Home Base page.
- Click the **Data Browser** button on the main toolbar.



Data Browser

The Data Browser has two **panes**; the left pane displays a categorical tree of data names and the right displays a grid of data instances corresponding to the data name selected. Until you select a data name, the right pane is initially empty.



Any editing you do to this data while in the Data Browser won't propagate back to your original database. You should make changes for simulation purposes only. Make any permanent changes to the data in the appropriate maintenance area of your original database.

Working with the Data Browser

This section describes the Data Browser and how to navigate it.

Navigating Data Names

Since the PepperTools data names are organized in hierarchical categories, the left pane of the Data Browser is presented as a tree structure. Each "branch" of the tree is a data category name, preceded by a box containing a plus ("+") or minus ("-") sign. Click the plus sign; it turns to a minus sign, and the members of that category are listed below. Click the minus sign; it turns to a plus sign, and the list contracts so it's no longer visible.

The members of a data category can be data subcategories or data names. Each subcategory name is preceded by a plus or minus box; any entry not so marked is a data name. To navigate to a data name, click the appropriate plus signs until the list containing the desired data name appears.



For more information about the data categories, see Exploring the Data Structure in this section.

Each data name has an associated popup menu. Right-click a data name to access its popup menu. The menu contains the **Add** command, the **Find** command or both; each invokes a page by that name.



For more information about Add pages and Find pages, see Using Data Browser Pages in this section.

Viewing Data Instances

In the left pane of the Data Browser, click the data name you want to view. The data instances for that data name appear in a grid in the right pane. Cells, rows, columns and the entire grid have associated property sheets and other pages accessible via popup menus.

- Right-click any cell in the data instance grid to access its popup menu.
- Right-click any row number or column heading to display a popup menu for the data that appears in that row or column. The row or column of data is highlighted.
- Right-click the box in the upper left corner of the grid to view a popup menu for all data instances. The entire grid is highlighted.



For more information about grids, see *Navigating Through the System: Working with Pages: Understanding Page Elements: Using Grids*. For more information about property sheets, see *Navigating Through the System: Working with Pages: Understanding Property Sheets*.

Moving and Sorting Data Instance Rows

You can move and sort data instances that appear in the right pane of the Data Browser. Sorts are executed from left to right in ascending order—the program sorts the rows based on the first column that contains an asterisk (*) in the column name, and then according to the second column, and so forth.

To reorder and sort key grid columns, complete the following steps:

1. Click a column heading that includes an asterisk (*) in the name. The column is highlighted.
2. Press the Shift key, then click and hold down the left mouse button on the heading. The cursor changes to an arrow with a box attached to its tail. Drag the arrow to your desired column position and release the mouse button. The data column moves to that location.
3. Click the Sort button to sort the data according to the new sequence.

Using Data Browser Command Buttons

You use command buttons to initiate a command or set an option. The following command buttons display at the bottom of the Data Browser:

Button	Function
Sort	Sort the rows of data in the right pane, in ascending order.
OK	Apply all changes and close the Data Browser.
Cancel	Ignore any changes and close the Data Browser.
Apply	Apply all changes, but remain in the Data browser.
Help	Invoke context-sensitive online documentation.



For more information about specific activities within the Data Browser, see *PeopleSoft Enterprise Planning Business Processes: Analyzing Your Enterprise Plan: Analyzing Your Plan Using the Data Browser* or *PeopleSoft Production Planning Business Processes: Analyzing Your Production Plan: Analyzing Your Plan Using the Data Browser*.

Exploring the Data Structure

Since the PepperTools data names are organized in hierarchical categories, the left pane of the Data Browser is presented as a tree structure. Use the tree to navigate to the data name you want, then click the data name to display its instances in the right pane.

Understanding the Data Hierarchy

The data names enable you to access the following facility, demand, and supply data:

Data Browser Data Names and Accessible Data

Categorized Data Name*	Data Instance Description	Data Instance Columns
Units—Units	Business unit names and descriptions.	Unit, Description
Calendars—Holiday Sets	Name of the holiday set.	Holiday Set
Calendars—Work Period Sets	Name of the work period set.	Work Period Set
Calendars—Calendar Specs	Name of the calendar spec.	Calendar Spec
Calendars—Calendars	Description of the calendar.	Calendar
Resources—Resource (PL/OP)	Parameters for resources.	Unit, Resource, Resource Class
Resources—Aggregate Resource (EP/OP)	Parameters for aggregate resources.	Unit, Resource, Resource Class
Resources—Changeover Entries (PL)	Item-to-item changeover costs for resources.	From Item Attribute, Value, To Item Attribute, Value, Resource Attribute, Value, Cost, Downtime
Resources—Downtimes (PL)	Schedule downtime periods for a selected resource.	Task, Start Time, End Time, Unit, Resource

<i>Categorized Data Name*</i>	<i>Data Instance Description</i>	<i>Data Instance Columns</i>
Items—Inventory Items	Inventory item data.	Unit, Item, Description, Planner Code, Item Category 1, Item Category 2, MPS
Items—Products	Product data.	Unit, Item, Description, Planner Code, Item Category 1, Item Category 2
Items—Planning Items	Planning BOM or the aggregate histograms for planning items.	Unit, Item, Description, Planner Code, Item Category 1, Item Category 2, MPS
Items—Phantom Items	Phantom item data.	Unit, Item, Description, Planner Code, Item Category 1, Item Category 2
Modeling Options—Attributes	Attributes data.	Mfg Attribute
Modeling Options—Purchase Options	Purchase options data.	Unit, Purchase Option, Item, Cost Per Item, Vendor
Modeling Options—Production options	Production options data.	Unit, Production option, Item, Cost, Cum Cost
Modeling Options—Transfer Options	Transfer options data.	From Unit, To Unit, Transfer Option, Carrier, Cost, Lead Time
Modeling Options—Transfer Preparations	Transfer preparation data.	Unit, Transfer Preparation
Modeling Options—Item Sourcing Logic—Sourcing Templates—Lowest Cost (EP/OP)	Create a lowest cost item sourcing strategy and give it a description.	Name, Description
Modeling Options—Item Routing Logic—Routing Templates—Lowest Cost (PL)	as above	as above
Modeling Options—Item Sourcing Logic—Sourcing Templates—Fixed Priority (EP/OP)	Create a fixed priority item sourcing strategy and give it a description.	Name, Description, Search Depth, Class

<i>Categorized Data Name*</i>	<i>Data Instance Description</i>	<i>Data Instance Columns</i>
Modeling Options—Item Routing Logic—Routing Templates—Fixed Priority (PL)	as above	as above
Modeling Options—Item Sourcing Logic—Sourcing Templates—Ratio (EP/OP)	Create a ratio item sourcing strategy and give it a description.	Name, Description
Modeling Options—Item Sourcing Logic—Sourcing Entries (EP/OP)	Select sourcing parameters.	Attribute, Attribute Value, Effective Start, Effective End, Template
Modeling Options—Item Routing Logic—Routing Entries (PL)	Select routing parameters.	as above
Modeling Options—Resource Preferencing Logic—Preferencing Templates—Fixed Priority (EP/PL)	Create a fixed priority resource preferencing strategy and give it a description.	Name, Description, Default
Modeling Options—Resource Preferencing Logic—Preferencing Templates—Fastest (EP/PL)	Create a fastest resource preferencing strategy and give it a description.	Name, Description
Modeling Options—Resource Preferencing Logic—Preferencing Entries (EP/PL)	Select resource preferencing parameters.	Attribute, Attribute Value, Resource Class, Effective Start, Effective End
Constraints—Capacity by Period Constraints (PL)	View and set the desired percentage for utilization of a given resource for a given period of time.	Unit, Resource, Start Time, End Time, Status
Constraints—Changeover Constraints (PL)	Edit and view resource changeover constraints.	Unit, Resource, Start Time, End Time, Status
Constraints—Stocking Constraints (EP/PL)	Edit and view safety and excess stock periods.	Unit, Item, Safety Status, Excess Status
Constraints—Reduce Routing Slack Constraints (EP/PL)	Edit and view routing slack constraints.	Distance Threshold, Aggressive Repair, Number to Repair, Status
Constraints—Purchase Contract Constraints (EP/PL)	Edit and view purchase contract constraints.	Constraint Name, Vendor, Status

<i>Categorized Data Name*</i>	<i>Data Instance Description</i>	<i>Data Instance Columns</i>
Constraints—Build Contract Constraints (EP/PL)	Edit and view build contract constraints.	Constraint Name, Status
Demand—Sales Orders	Edit and view sales order information, add and modify sales order lines, and view and edit Ship Sets.	Unit, Sales Order, Customer, Order Date, Revenue
Demand—Forecasts (EP/PL)	Edit and view forecast/target start times and dates, end times and dates, number of expected shipments in the specified time period, number of booked sales orders, and target quantities.	Unit, Item, Forecast Date, Forecast Qty, Forecast Type
Demand—Carriers	Edit and view specific carriers and assign a priority to each.	Carrier, Priority
Demand—Customers	Edit and view customer data.	Customer, Description
Demand—Customer Regions	Edit and view priority customer region settings.	Customer Region, Priority
Demand—Sales Territories	Edit and view designated sales territories and assign a priority value for each territory.	Sales Territory, Priority
Supply—Purchase Orders	Edit and view purchase order information, add and delete lines, view a deliveries Gantt chart, and edit quantities.	Unit, Purchase Order, Vendor, Order Date
Supply—Planned Purchase Orders	Edit and view planned order information, including Gantt charts, delete planned orders, and edit quantities and delivery dates.	Unit, Item, Vendor, Start Date, End Date, Planned Purchase Order
Supply—Vendors	Name of the vendor.	Vendor
Supply—Production Areas	Edit and view production area information.	Unit, Production Area, Description

<i>Categorized Data Name*</i>	<i>Data Instance Description</i>	<i>Data Instance Columns</i>
Supply—Production Orders	Edit and view production order information.	Unit, Production Order, Item Unit, Item, Production Area, Start Time, End Time, Starting Quantity, Order Quantity, Status, Production option
Supply—Production	Edit and view production information.	Production, Item Unit, Item, Production Area, Start Date, End Date, Starting Quantity, Order Quantity, Status, Production option, Unit, Production Order
Supply—Transfer Orders	Edit and view transfer order information.	Unit, Transfer Order, From Unit, From Item, To Unit, To Item, Start Date, End Date, Starting Quantity, Order Quantity, Status, Transfer Option
Supply—Transfers	Edit and view transfer information.	Transfer, From Unit, From Item, To Unit, To Item, Start Date, End Date, Starting Quantity, Order Quantity, Status, Transfer Option, Unit, Transfer Order
Optimization—Optimize Templates (EP/PL)	Edit and view optimize template and associated optimize phases.	Optimize Template, Rerun Phases
Optimization—Optimize Phases (EP/PL)	Edit and view how specific constraints are repaired during a set number of optimize iterations.	Optimize Phase, Iterations, Time Limit, Repairs Per Iteration
Optimization—Capacity By Period Scorer Lists (PL)	Edit and view capacity by period repair strategies.	Scorer List
Optimization—Generate Schedule—Queries (EP/PL)	Collect production tasks for a template entry.	Query, Item Class
Optimization—Generate Schedule—Dispatch Logic (EP/PL)	Specify how a set of production tasks, collected by a query, will be dispatched.	Dispatch Logic

<i>Categorized Data Name*</i>	<i>Data Instance Description</i>	<i>Data Instance Columns</i>
Optimization—Generate Schedule—Templates (EP/PL)	Maintain a group of queries in a specific order, so that they can be saved and reused..	Template
Prioritization—Prioritize Templates (EP/PL)	Edit and view order prioritization templates.	Prioritize Template, Maximum Depth
Prioritization—Selection Rules (EP/PL)	Edit and view order prioritization selection rules.	Selection Rule, Rule Subclass
Prioritization—Prioritize Rules (EP/PL)	Edit and view primary order prioritization rules.	Prioritize Rule, Rule Subclass
Prioritization—Scheduling Rules (EP/PL)	Edit and view order prioritization scheduling rules.	Scheduling Rule, Rule Subclass
Miscellaneous—Task Statuses	View task classes, ID's, behaviors, and edit Gantt chart task bar colors and patterns.	Task Class, Status ID, Status Behavior, Color Index, Pattern Index

*A data type followed by "EP", "PL" or "OP" indicates that it appears respectively in the Enterprise Planning, Production Planning or Order Promising client. Entries that include none of these appear in all three clients.

Understanding Data Instance Columns

The following is a description of most of the data instance columns. You'll find many of these in several data types.

Aggressive Repair

This is a check box. When checked, the system sets this option to aggressive repair, which could be violated as a result of the repair. When unchecked, the system sets this option to conservative repair—a Reduce Routing Slack repair won't violate any resource constraints.

Attribute

Name of the attribute.

Attribute Value

Value of the attribute.

Production option

Name of the production option.

Calendar

Name of the calendar.

Class

Class of resource.

Color Index

A number to indicate a color on task status Gantt charts.

Constraint Name	Name of the specified constraint.
Cost	Specified cost of item or operation.
Cost per Item	Cost per item.
Cum Cost	Cumulative cost for the item.
Customer	Name of the customer.
Description	Description of the specified item, unit, lowest cost template, or fixed priority template.
Effective Start	Beginning date and time for routing.
Effective End	End date and time for routing.
End Date	Concluding date and time of the specified operation.
End Time	End date and time of the specified operation.
Excess Status	Status of the excess constraint.
Forecast Qty	Total actual sales order demand for the item for the time period.
Forecast Date	Time period for the forecast.
Forecast Type	Assigned type for the forecast.
From Item	Name of item at source unit.
From Item Attribute	Association between source item and sourcing logic.
From Unit	Name of unit from which you ship an item.
Holiday Set	Name of the holiday set.
Item	Name or number of the item.
Item Category 1	First user defined item selection attribute.
Item Category 2	Second user defined item selection attribute.
Item Class	Name of the item class.
Item Unit	Location of the item.
Iterations	Number of repeats Optimizer makes for an optimal schedule.
Lead Time	Expected duration to transfer an item from unit to unit.
Mfg Attribute	Manufacturing attribute name.

Name	Name of the lowest cost or fixed priority template.
Order Date	Date the order was created.
Order Quantity	Amount of the production order.
Pattern Index	A number to indicate a pattern on task status Gantt charts.
Planner Code	Name or number of planner.
Production	Name of the production.
Production Order	Number of the production order.
Production Area	Functional component of a unit.
Purchase Order	Purchase order number.
Purchase Option	Name of the purchase option.
Resource	Description of the resource.
Resource Attribute	Attribute name of the resource.
Resource Class	Class of resource.
Revenue	Total revenue for the selected sales order.
Sales Order	User-assigned name or number for the sales order.
Selection Rule	Selection rule for order prioritization.
Start Date	Beginning date and time of the specified operation.
Starting Quantity	Amount of order quantity yielded.
Start Time	Start date and time of the specified operation.
Status	Status of operation
Status Behavior	Status behavior indicator—can be <i>Active</i> , <i>Canceled</i> , <i>Complete</i> , <i>Pending</i> , <i>Firmed</i> , <i>Dispatched</i> , or <i>Planned</i> .
Status ID	The user-defined Status ID options can include <i>Active</i> , <i>Canceled</i> , <i>Complete</i> , <i>Pending</i> , <i>Open</i> , <i>Firmed</i> , <i>Planned</i> , <i>Dispatched</i> , <i>In Process</i> , <i>Released</i> , or <i>In Transit</i> .
Task	Name of the task.
Task Class	Name of the task class for the task status.
Template	Name of the specified template.
To Item	Name of item at destination unit.

To Item Attribute	Association between destination item and sourcing logic.
To Unit	Name of unit to which you ship an item.
Transfer	Name of the transfer.
Transfer Option	Name of the transfer option.
Transfer Order	Name of the transfer order.
Transfer Preparation	Name of the transfer preparation.
Unit	Name of the specified unit location.
Value	Attribute value of the specified item.
Vendor	Name of the vendor.

Closing the Data Browser

You can close the Data Browser in any of the following ways:

- Click the standard Windows **Close** button in the title bar.
- Click the **OK** button to save your changes and close the Data Browser.
- Click the **Cancel** button to close the Data Browser without saving your changes.

Using Data Browser Pages

This section shows you how to work with the pages you access through the Data Browser.

Understanding PepperTools Pages

There are three categories of PepperTools pages that enable you to work with data such as business units, calendars, resources, items, modeling options, constraints, demand, and supply. They are:

- Add pages.
- Find pages.
- Property sheets.

Add pages enable you to add data, find pages enable you to apply filter criteria to find specific data, and property sheets enable you to view and edit data.

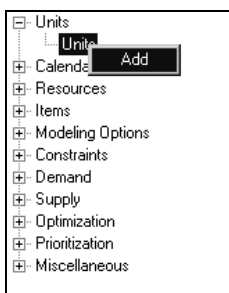
Working with Add Pages

Use Add pages to add instances of units, calendars, resources, items, modeling options, constraints, demand, supply, optimization or prioritization data to your simulation.



Important! Any editing you do to this data while in the Data Browser won't propagate back to your original database. You should make changes for simulation purposes only. Make any permanent changes to the data in the appropriate maintenance area of your original database.

1. Select Tools, Data Browser to display the Data Browser.
2. Use the tree in the left pane to navigate to the data name you want.
3. Right-click a data name. A popup menu appears.



Add Command in Popup Menu

4. Select Add to display an Add page.

Add Unit Page

5. Enter the information for your new data instance.

In the Add Unit page, for example, you enter a **Unit** name, select a **Class** from the drop-down list, enter a **Description** of the business unit, select a **Period Of Supply Calendar** from the drop-down list, and fill in the rest of the fields.

6. Click OK to complete the add procedure.

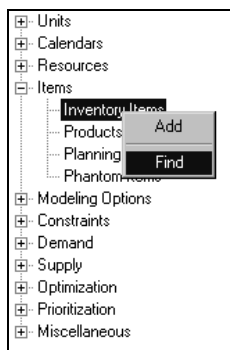


You can also access Add pages from other areas of the program by clicking an **Add** button when it appears in a page.

Working with Find Pages

Use the Find pages to apply filter criteria to find instances of resources, items, modeling options, demand, or supply data. Find pages contain query parameters as well as the results of the query.

1. Select Tools, Data Browser to display the Data Browser.
2. Use the tree in the left pane to navigate to the data name you want.
3. Right-click a data name. A popup menu appears.



Find Command in Popup Menu



The Find Item page is only available for certain categories and data names. The **Find** command appears on the popup menu only when this is the case.

4. Select Find to display a Find Item page.

Find: Item

Find Now Unit Filter: Planner Code Filter:

Select Item Filter: Item Category 1 Filter:

Description Filter: Item Category 2 Filter:

Item Class: All Items Distribution Plan:

Master Plan:

Material Plan:

Sort

	Unit*	Item*	Description*	Item Class
1	SM	ALUMINUM	Aluminum Tubing	Inventory_Item
2	SM	BLUE_BICYCLE	Blue Bicycle	Inventory_Item
3	SM	BLUE_FRAME	Blue Frame	Inventory_Item
4	SM	BLUE_FRONT_FORK	Blue Front Fork	Inventory_Item
5	SM	BLUE_FRONT_WHEEL_ASSEMBLY	Blue Front Wheel Assembly	Inventory_Item
6	SM	BLUE_PAINT	Bicycle Blue Paint	Inventory_Item
7	SM	BLUE_PIGMENT	Blue Pigment	Inventory_Item
8	SM	BLUE_TANDEM_BIKE	Blue Tandem Bicycle	Inventory_Item
9	SM	BLUE_TANDEM_FRAME	Blue Tandem Frame	Inventory_Item

OK Cancel Apply Help

Find Item Page

5. Enter the appropriate filter criteria in the page header.

In the Find Item page, for example, you enter filter information in the **Unit Filter**, **Item Filter**, and **Description Filter** fields and select an **Item Class** from the drop-down list.

6. Click Find Now to apply the filter criteria you entered; it displays in the grid in the lower part of the form.
7. Choose a row in the grid and click the Select button to display the row's data in the associated form.



You can also access Find pages from other areas of the program by clicking a **Select** button when it appears in a page.

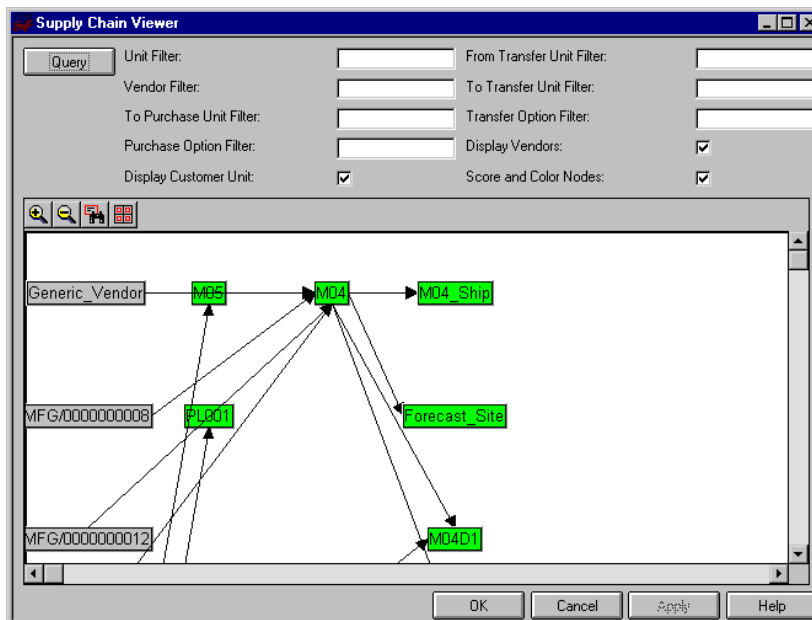
CHAPTER 3

Using the Supply Chain Viewer

The PepperTools Supply Chain Viewer is a page that enables supply chain planning as well as collaborative planning by building a visual representation of your supply chain model. You can modify and rearrange this diagram to more appropriately display the supply chain for your purposes. You can solve problems quickly by troubleshooting your plan at a given business unit and drilling down to more detailed information. The Supply Chain Viewer shows all business units and all possible connections between them.

Accessing the Supply Chain Viewer

To start the Supply Chain Viewer, in Enterprise Planning or Production Planning select **Tools, Supply Chain Viewer**; in Order Promising select **Tools, Advanced, Supply Chain Viewer**.



Supply Chain Viewer

The viewer page consists of three main areas:

- The page header, which contains data entry fields, query check boxes, and the **Query** button.
- The main work area, which includes the diagram display, the diagram toolbar and the diagram scrollbars.

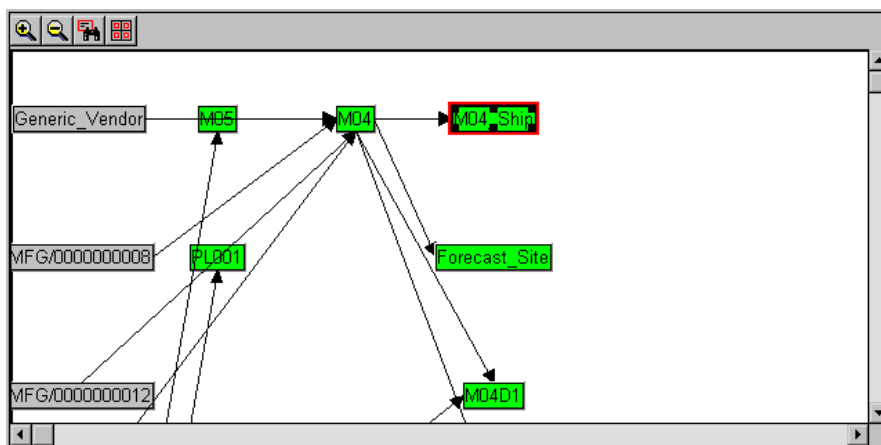
- The standard command buttons at the bottom of the page.



The diagram display is initially blank.

Working with the Supply Chain Diagram

Click the **Query** button to generate the supply chain diagram based on the business unit, vendor, purchase option and transfer option data available within the server.



Supply Chain Diagram and Toolbar



For more information about modifying the selection of source data used for the diagram, see Customizing the Diagram Query in this section.

The supply chain diagram can easily be larger than the display area, so you can use the scrollbars provided to shift the desired portion of the diagram into view. You can also change the size of the Supply Chain Viewer page using its standard Windows controls.

Accessing Vendor and Unit Information

The supply chain diagram consists of **nodes**—labeled boxes that represent vendors and business units, and **connections**—arrows that represent purchase options and transfer options. By default, the diagram is constructed with the vendors sorted vertically on the left, then uses the purchase option data to construct connections to the first column of business units, and continues to use transfer option data to generate each subsequent column, with customer units at the far right.

Double-click any node or connection to view its property sheet. You can also right-click any business unit to display a popup menu, then select the **Unit Scorecard** for that unit.



In a property sheet or scorecard, any data entry field or cell containing blue text is read-only—you cannot alter the value displayed.



For more information about using the Supply Chain Viewer for planning, see *PeopleSoft Enterprise Planning Business Processes: Analyzing Your Enterprise Plan: Troubleshooting Your Plan Using Supply Chain Viewer* or *PeopleSoft Production Planning Business Processes: Analyzing Your Production Plan: Using the Supply Chain Viewer*.

Exploring the Diagram Toolbar

The Supply Chain diagram toolbar contains the following buttons:



The **Zoom In** button increases the magnification of your supply chain diagram by approximately 25%. This doesn't change the content of the diagram, only its appearance.



The **Zoom Out** button decreases the magnification of your supply chain diagram by approximately 25%. This doesn't change the content of the diagram, only its appearance.



The **Find** button invokes the Find Unit page to quickly locate the node or nodes you want to troubleshoot.



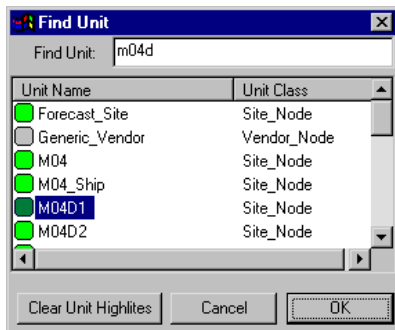
The **Align** button repositions all nodes on an invisible grid so they're lined up neatly in rows and columns.



When you click the Align button, each node is moved to the nearest grid position, so it's possible for multiple nodes to end up at the same position, with the top one obscuring the others at that spot. To avoid this, click and drag each node so it's close to a different grid location from the others, then click the Align button.

Using the Find Unit Page

The Find Unit page enables you to locate and select one or more nodes. The page contains a sorted scrolling list of node names with columns for Unit Name and Unit Class, a **Find Unit** data entry field at the top, and three command buttons at the bottom—**Clear Unit Highlites**, **Cancel** and **OK**.



Find Unit Page

If the **Score and Color Nodes** check box was selected for the current query, a color code appears before each node name in the list, matching the node's color in the diagram.



When the Find Unit page opens, the node at the top of the list is automatically selected, in addition to any other nodes already selected.

Click the **Clear Unit Highlights** button to deselect all nodes.

- To select a single node, enter part or all of a node name in the **Find Unit** field; the list will scroll to the first matching name as you type, and select that node. You can also scroll down the list to the desired node name, and click the name to select it.
- To select multiple nodes, click the name of the first one you want, then scroll through the list and hold down the **Ctrl** key as you click each additional name.

When you've selected the nodes you want, click the **OK** button or **Cancel** button to return to the diagram.



OK and **Cancel** currently have the same effect—selected nodes remain selected.

Editing the Diagram

The diagram is easy to modify so you can visualize the supply chain according to your preferences. You can change the size, shape or position of any node, and the connections automatically reorient themselves to maintain the same logical relationships between units.

Selecting Nodes

- Click any single node, and it's surrounded by a red box to indicate that it's selected.

You can select several nodes simultaneously, using one of the following methods:

- Click one of the desired nodes to select it. Hold down the **Shift** key as you click each additional desired node. All selected nodes are surrounded by red boxes.
- Click on the background and drag the pointer across the desired nodes; a rectangular "marquee" surrounds the nodes that will be selected when you release the mouse button.
- You can also select multiple nodes with the Find Unit page.



For more information about selecting nodes with the Find Unit page, see Exploring the Diagram Toolbar: Using the Find Unit Page in this section.

Deselecting Nodes

To deselect all nodes, simply click anywhere in the background of the diagram display.

Modifying Node Position

Click the center of any node and drag to reposition it in the display area; it maintains its connections to the other nodes as you drag.

If you've selected multiple nodes, you can move them as a group without changing their positions relative to each other. Click the center of any node in the selected group and drag to reposition it; the rest of the group moves in unison, and they maintain their connections as you drag.



If you change the positions of any nodes in the diagram, then click the **Query** button, the nodes will revert to the positions they were in when you started the Supply Chain Viewer. To retain your changes through a new query, you must save the current state: Click **OK** to exit, then restart the Supply Chain Viewer and reapply your query.

Modifying Node Size and Shape

When you select a node, a black box appears with six **control points**, at each corner and in the center of its longer sides. If you select multiple nodes, only one of them will have the black box with control points. You can click and drag any control point to resize the node or change its proportions:

- Use the control points on the left and right side to alter the node's width.
- Use the top and bottom points to alter its height.
- Use the corner points to alter width and height simultaneously.



For more information about modifying the diagram display, see Exploring the Diagram Toolbar in this section.

Customizing the Diagram Query

When you invoke the Supply Chain Viewer, the diagram display is blank. Before generating the diagram, you can make settings in the page header to modify the content and appearance of the diagram. You can also change these settings and regenerate the diagram at any time.

Filtering Content

Use the data entry fields in the page header to restrict the diagram to just the nodes and connections on which you want to concentrate. The fields filter out all but the values you enter, to produce a diagram relevant to only those nodes and connections.

The filter fields are as follows:

Unit Filter	From Transfer Unit Filter
Vendor Filter	To Transfer Unit Filter
To Purchase Unit Filter	Transfer Option Filter
Purchase Option Filter	

Setting Display Options

In addition to the data entry fields, the page header has three check boxes:

Display Customer Unit	When Display Customer Unit is checked, the customer units and their associated transfer options on the server will be displayed as nodes and connections in the diagram. When it isn't checked, they'll be omitted from the diagram.
Display Vendors	When Display Vendors is checked, the vendors and their associated purchase options on the server will be displayed as nodes and connections in the diagram. When it isn't checked, they'll be omitted from the diagram.
Score and Color Nodes	When Score and Color Nodes is checked, the colors of the nodes in the diagram correspond to your chosen scoring scenario. When it isn't checked, they'll display in gray.

Applying Your Settings

Click the **Query** button to generate the supply chain diagram based on the business unit, vendor, purchase option, and transfer option data available within the server. Any filter entries or display options you set are used to modify the query and its resulting diagram accordingly.

Using the Viewer Command Buttons

When you've completed your analysis, click the **OK** button to exit the Supply Chain Viewer. The positions of diagram nodes, the contents of the filter fields, and the state of the check boxes in the page header are saved. These settings will remain in effect as long as the current server is still running.



Node sizes and the current magnification of the diagram are never saved—they'll be reset to their defaults.



OK and **Cancel** currently have the same effect—the current state of the viewer is saved.

CHAPTER 4

Saving and Loading Files

This section explains how to name, store and access external data, constraint settings, and the current state of your PepperTools application.

Understanding File Saving and Loading

The PepperTools applications work with three kinds of stored information that you can access using the File Browser:

- The primary datasets, which are supplied by the DataBridge process as **.command** files. PepperTools can load these files for planning, but can't save to the **.command** format—use **File, Export to RDBMS** to generate **.sql** output data instead.
- Control panel settings, which you can name and save for later retrieval. Save control panel settings as **.page** files.
- Snapshots, which are files that preserve an image of the schedule as it exists at a given time or at a selected stage of optimization. Save your snapshots as **.lps** files.



You can save bookmark settings as snapshots, to make them available in future sessions



For more information about bookmarks, see Using the Main Menu: Working with Bookmarks.

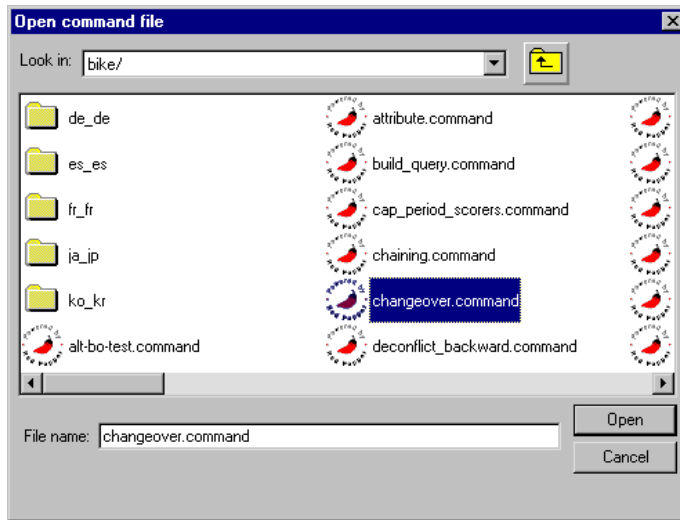
Using the File Browser

The File Browser enables you to navigate in your directory structure, find the data files that you want to use, load desired data into your PepperTools application, and save configuration files. When you call the file browser, its fields and lists are the same as when you last used it.

Loading Files

To Use the File Browser to load a dataset command file, control panel setting file or snapshot file:

1. Start your PepperTools application.
2. Access the File Browser to load data by selecting one of the following options:
 - File, Load Command File.
 - Tools, Control panel, Load.
 - File, Load Snapshot.



Open Command File Window

3. To select the correct dataset command file, control panel setting file or snapshot file, either:
 - Enter the correct full path name in the **File name** field, or
 - Click the appropriate directories to show the correct path, then click the filename in the browser file list. When you select a file or directory from the list, its path appears in the **Look in** field.
4. Click **Open** to load the selected file into memory.



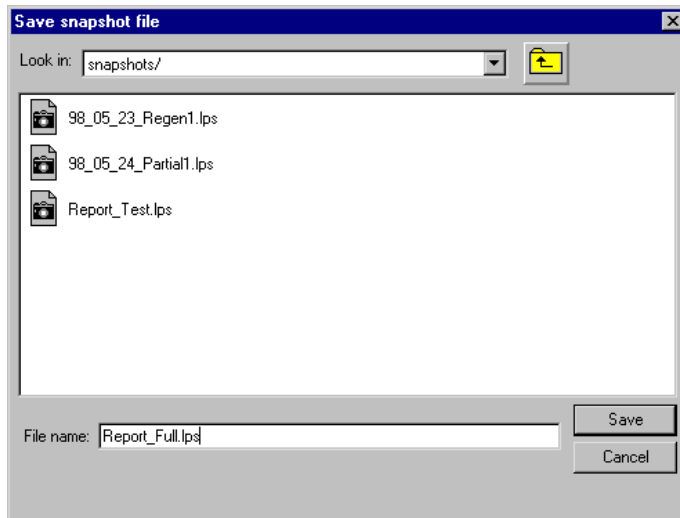
You can't load more than one complete dataset or one snapshot at a time. You can load incremental data files that relate to the dataset that you're currently running.



For more information, see Working with Data Files, Loading Control panel Settings, and Working with Snapshots in this section.

Saving Files

1. To save a control panel setting file or snapshot file with the File Browser:
2. Access the File Browser to save data by selecting one of the following options:
 - Tools, Control panel, Save As.
 - File, Save Snapshot.



Save Snapshot File Window

3. To specify a filename, either:
 - Enter the correct full path name in the **File name** field, or
 - Click the appropriate directories to show the correct path, then enter a filename in the **File name** field. When you select a file or directory from the list, its path appears in the **Look in** field.
4. Click **Save**.



For more information, see Working with Data Files, Saving Control panel Settings, and Working with Snapshots in this section.

Working with Data Files

Your PepperTools application contains the facility model and preliminary schedule it creates using data from the DataBridge process in a set of files collectively referred to as the data file. You load the data file by loading the appropriate command file from the file browser.

The DataBridge process stores the data files in the data/directory where all users access the files. Typically, the data/directory has dated subdirectories where the daily data files are stored. If your unit downloads more than once per day—once per shift, for example—each day will have a set of subdirectories with appropriate identifiers for each download. A typical path for the data file directories is:

```
/company-specific_path/data/mm_dd_yy.1/  
/company-specific_path/data/mm_dd_yy.2/  
/company-specific_path/data/mm_dd_yy.3/
```

For example, the data files from the end of the second shift on March 19, 1997 would be in:

```
/auto/u2/sched/data/03_19_97.2/
```

Enter the name of the command file that you want to use, then click **Open** to load the data.



For more information about finding, selecting and loading files, see Using the File Browser in this section.

Loading a New Dataset

If your PepperTools application is already running, and you want to work with a different dataset or snapshot than the one you're currently using, follow steps 1-6; if the server you want to use is newly started, follow steps 4-6.

1. Select **File, Save Snapshot** to produce a snapshot of your current dataset.
2. Select **File, Restart Server** to flush the current dataset.

The Restart Server page appears, with a list of current server users, and the default arguments to be used by the restart procedure.

3. Enter the appropriate arguments, and continue according to your local procedure. Check with your system administrator for specifics.



Important! When you restart a server, its current users will be disconnected, and any changes since the last saved snapshot will be lost.

4. Select **File, Connect Server** and choose your desired server from the Select Server page.



Give the server enough time to start or restart before trying to connect. If it hasn't started, you'll see the message "No servers were found on this registration server." Once it has started, you'll see the Select Server page.

5. To load the data, either:
 - Select **File, Load Command File** to retrieve a new dataset, or
 - Select **File, Load Snapshot** to retrieve a snapshot from a previous session.
6. Use the File Browser to find and load the desired command file or snapshot.



Your PepperTools application can't load data from a previous session unless you saved it as a snapshot.

Saving Control panel Settings

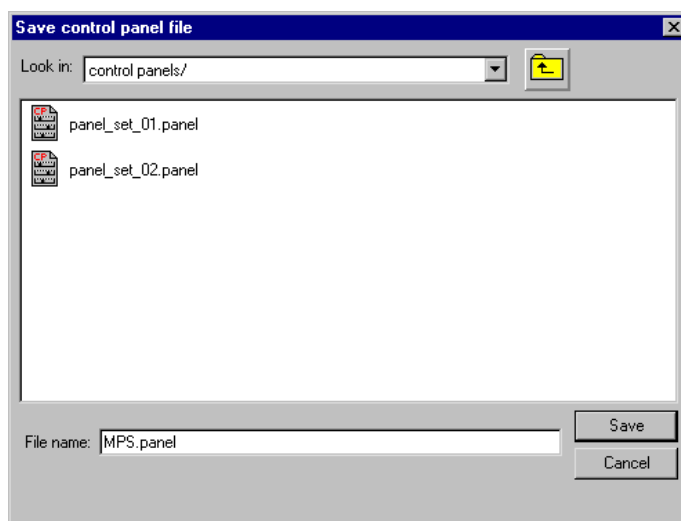
The Current Setting field in the control panel enables you to save control panel settings as a named set that you can later retrieve and reuse.



For more information about the control panel, see *PeopleSoft Enterprise Planning Business Processes: Managing Constraints and Optimization* or *PeopleSoft Production Planning Business Processes: Managing Constraints and Optimization*.

To save your control panel settings:

1. Select **Tools, Control panel, Save as** to display the Save Control panel File window.



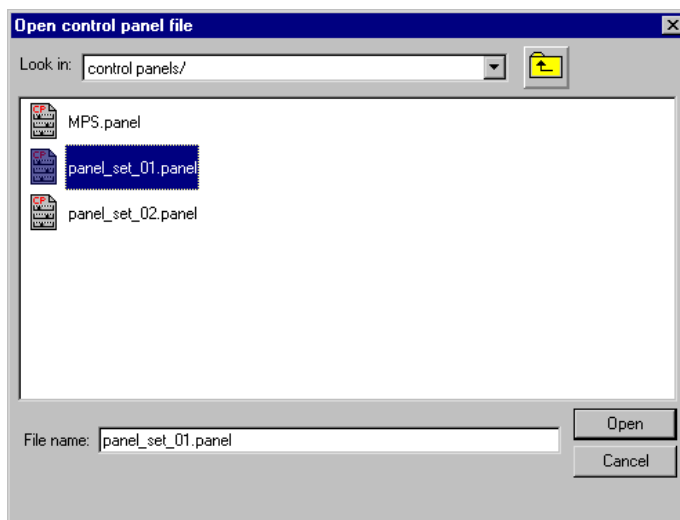
Save Control panel File Window

2. To specify a filename, either:
 - Enter the correct full path name in the **File name** field, or
 - Click the appropriate directories to show the correct path, then enter a filename in the **File name** field. When you select a file or directory from the list, its path appears in the **Look in** field.
3. Click **Save**.

Loading Control panel Settings

To load a control panel setting file with the file browser:

1. Start your PepperTools application.
2. Select **Tools, Control panel, Load** to display the Open Control panel File window.



Open Control panel File Window

3. To specify the correct control panel file, either:
 - Enter its full path name in the **File name** field, or
 - Click the appropriate directories to show the correct path, then click the filename in the browser file list. When you select a file or directory from the list, its path appears in the **Look in** field.
4. Click **Open** to load the selected file into memory.

Working with Snapshots

A snapshot is a PepperTools file that preserves an image of a schedule as it exists at a given time or at a certain stage of optimization. You only save a snapshot to load back into Your PepperTools application—it's not an external output.

Depending on the way your company's system is set up, users have their own snapshots directory for the snapshots that they choose to save. A typical path for the snapshots directories is:

```
/company-specific_path/user-specific_path/snapshots/
```

Using What-if Snapshots

Use what-if snapshots to help determine the best possible solution for your scheduling conflict. You do this by running the Optimizer several times using different prospective combinations each time.

For example, a production control scheduler needs to update the data files from the end of the second shift on March 20, 1999. The production control scheduler enters each of the prospective combinations, runs the Optimizer for each scenario, and saves the results of each run in a snapshots directory:

```
/auto/u2/sched/jmontan/snapshots/snap_032099_mixA
```

```
/auto/u2/sched/jmontan/snapshots/snap_032099_mixB
```

```
/auto/u2/sched/jmontan/snapshots/snap_032099_mixC
```

The information from the what-if snapshots will help the production control scheduler determine the most effective scheduling scenario.

Using Backup Snapshots

You can use snapshots to back up your work. Save a snapshot periodically during a long planning session to minimize the chance of losing your work in the event of a power failure or system crash. You can also save your snapshots to a file called **backup** (or something similar), overwriting the file each time.



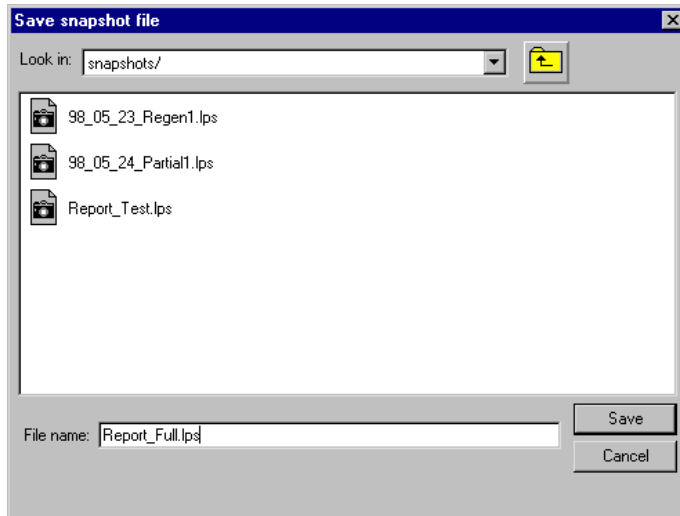
If your file is large, it can take a long time to save.

Saving Archive Snapshots

Use snapshots to save the schedule you receive from the PepperTools DataBridge process at the beginning of your session as well as the one you transmit back to the external system at the end of your session.

To save an archive snapshot:

1. Select **File, Save Snapshot** to display the Save Snapshot window.



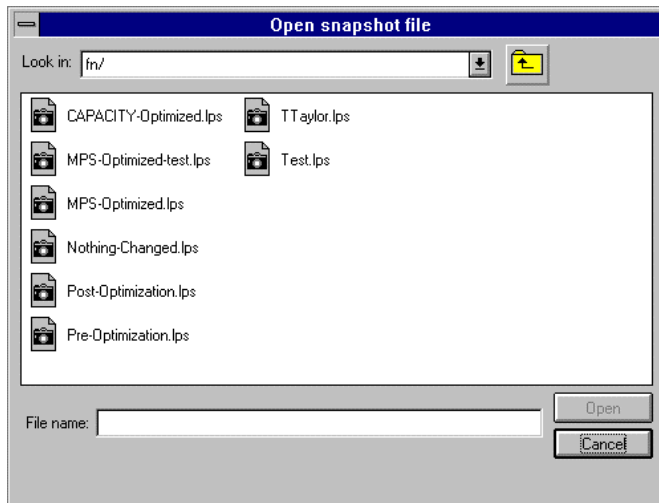
Save Snapshot File Window

2. To specify a filename, either:
 - Enter your desired full path name in the **File name** field, or
 - Click the appropriate directories to show the correct path, then enter a filename in the **File name** field. When you select a file or directory from the list, its path appears in the **Look in** field.
3. Click **Save**.

Loading Snapshots

To load a snapshot:

1. Start your PepperTools application *without loading a dataset* according to your local procedure.
2. Select **File, Load Snapshot** to display the Load Snapshot window.



Open Snapshot File Window

3. To specify the correct snapshot file, either:
 - Enter its full path name in the **File name** field, or
 - Click the appropriate directories to show the correct path, then click the filename in the browser file list. When you select a file or directory from the list, its path appears in the **Look in** field.
4. Click **Open** to load the selected file into memory.



You can't load a snapshot if a dataset is already loaded. You can't load more than one snapshot at a time.



For more information about switching between snapshots, see Loading a New Dataset in this section.

CHAPTER 5

Using the Main Menu

This section describes your PepperTools application's Main Menu.

Viewing the Menu Before Connecting the Server

Before connecting a server, your PepperTools application displays the following menu options:

Main Menu Contents Before Connecting the Server

Menu, Command	Description
File, Connect	Connect client to a data server.
File, Connection Options	Specify the location and socket of the Registration Server.
File, Set Catalog Location	Specify the location of your catalog file.
Help Files Location	Configure where PepperTools looks for the help files. They can be located on a web server, on a network shared drive, or a local drive.
File, Exit	Exit the program.
View, Toolbar	Display and hide toolbar.
View, Status Bar	Display and hide status bar.
Help, Help Topics	Invoke context-sensitive online documentation.
Help, About Planning Client	Display general information about the client application in use.

Establishing the Server

When you first start your PepperTools application, you need to do two things before connecting the server:

- Configure the Registration Server.

The Registration Server keeps track of all currently running data servers on your network and

provides the list of servers that you see when using the **Connect Server** command.

- Specify the location of your catalog file, which configures the PepperTools client for your application.



You should only have to complete these two steps the first time you set up the client on your computer.

Configuring the Registration Server

To specify the location of the Registration Server:

1. Select **File, Configure RServer**.
2. Type the name of the RServer host, and the RServer socket number.



Talk to your system manager to get the correct values for these two items.

3. Click **OK**.

Setting the Catalog Location

To identify the location of your catalog file, complete the following steps:

1. Select **File, Set Catalog Location** to display the Open File Browser dialog box.
2. Navigate to your computer's files. Check with your system manager to determine the location of the catalog file for the product you are running.
3. Select the *Catalog.000* file.
4. Click **Open** to load the catalog file.

Connecting to the Server

Once you've completed the above steps, you can connect to a server. To connect to the server, complete the following steps:

1. Select **File, Connect Server**.

The Select Server page displays your user ID and a list of available servers.

2. Select a server from the list of available servers.

If no list displays, make sure that you've correctly configured your registration server. If there are still no servers listed, then none are running on your network. Check with your system manager.

3. Click **OK** to connect to the server.

If a dataset is already loaded from that server, the Home Base page appears. Otherwise, you'll see the message "The server you have connected to contains no data."



Clients and servers should be attached to the correct "matched" version of each other. If you run a current client and connect to an old server, the following message appears: "Server is too old for client." If you run an old client and connect to a current server, the following message appears: "The client is too old for the server." Either message means that you should run matching client and server versions.

Viewing the Menu After Connecting the Server

After the server has been connected, but before you load a dataset or snapshot, the main menu changes somewhat, as follows:

Main Menu Contents After Connecting the Server

Menu, Command	Description
File, Disconnect from Server	Detach the client from the current server.
File, Load Command File	Load or add to a dataset.
File, Load Snapshot	Load a previously used dataset and state.
File, Create Main Environment	Set up the program's interface without loading a dataset, providing full access to all features for generating a dataset interactively.
File, Exit	Exit the program.
View, Toolbar	Display and hide toolbar.
View, Status Bar	Display and hide status bar.
Help, Help Topics	Invoke context-sensitive online documentation.
Help, About Planning Client	Display general information about the client application in use.



For more information about loading a dataset, see Saving and Loading Files.

Viewing the Menu After Loading a Dataset

After loading a dataset, the main menu becomes substantially larger. The following table describes all the available commands on the main menus of Enterprise Planning, Production Planning and Order Promising. The first column indicates the sequence of menus and submenus you select to find each command, the second column lists the command itself, and the third column outlines the command's purpose:

Main Menu Commands After Starting the Server

Menu Path*	Command*	Description
File	Show Server Users	Show users on the client server.
File	Disconnect from Server	Detach the client from the current server.
File	Restart Server	Disconnect the client from the server and restart the server.
File	Load Command File	Load or add to a dataset.
File	Save Snapshot	Save current data in the server to disk.
File, Reports	Planned Purchase Orders (EP/PL)	Print planned orders sorted by order number, item, buyer, and/or vendor.
File	Export to RDBMS (EP/PL)	Save .sql data externally.
File	Exit	Exit Production Planning.
Edit	Cut	Remove highlighted text to the Clipboard.
Edit	Copy	Copy highlighted text to the Clipboard.
Edit	Paste	Paste Clipboard text at the insertion point or highlighted text.
Edit	Edit Class (EP/PL)	View/edit time fences.

Menu Path*	Command*	Description
Edit	Bookmark (EP/PL)	Set, switch to, and delete bookmarks.
Edit, Preferences	Environments	View/edit production environments including time fences, mfg, optimization, struggle, changeover, mfg attrib. categories, shuffle, capable to promise, and date penalties.
Edit, Preferences	Time Fences	View/edit time fences.
Edit, Preferences	Priority Scale Factors (EP/PL)	Set scale factors for sales order priorities.
Edit, Preferences	Update Display Names (EP/PL)	Update task names and all display names.
Edit, Preferences	Generate Items Where Used (EP/PL)	Generates item where used information for material usage calculations.
Edit, Preferences	Clear Items Where Used (EP/PL)	Clears item where used information to reduce memory usage.
View, Spreadsheets	Build Schedule (with Resource) (EP/PL)	View build schedule resource quantities.
View, Spreadsheets	Build Schedule (with Production Option) (EP/PL)	View production quantities including production options.
View, Spreadsheets	Master Production Plan (EP/PL)	View MPS information. Enables you to balance the supply and demand by using master scheduling.
View, Spreadsheets	Plan Summary (EP/PL)	View Planning Summary.
View, Spreadsheets	Capacity (PL)	View/edit factory capacity information.
View, Spreadsheets	Aggregate Capacity by Time (EP/PL)	View aggregate capacity by Time information.
View, Spreadsheets	Aggregate Capacity by Unit (EP/PL)	View aggregate capacity by Unit information.
View, Spreadsheets	Financial Overview (EP/PL)	View the total cost, profit, and revenue of the Production Planning schedule.

Menu Path*	Command*	Description
View, Spreadsheets	Transfer Supply (EP/PL)	View supply transfers between business units.
(EP) View, Gantt Charts	All Tasks	View Gantt chart of all tasks sorted by start time.
(PL) View, Gantt Charts, Task	as above	as above
(EP) View, Gantt Charts	Selected Tasks	View Gantt chart of all selected tasks sorted by start time.
(PL) View, Gantt Charts, Task	as above	as above
(EP) View, Gantt Charts	Production	View Gantt chart of production sorted by start time.
(PL) View, Gantt Charts, Task	as above	as above
(EP) View, Gantt Charts	Production Operations	View Gantt chart of production operations—subtasks—sorted by start time.
(PL) View, Gantt Charts, Task	as above	as above
(EP) View, Gantt Charts	Selected Shipments	View Gantt chart of selected shipment tasks.
(PL) View, Gantt Charts, Task	as above	as above
(EP) View, Gantt Charts	Forecasts	View Gantt chart of forecast shipments sorted by start time.
(PL) View, Gantt Charts, Task	as above	as above
(EP) View, Gantt Charts	Selected Transfers	View Gantt chart of selected transfer tasks.
(PL) View, Gantt Charts, Task	as above	as above
(EP) View, Gantt Charts	Selected Purchase Orders	View Gantt chart of selected purchase order line delivery tasks.
(PL) View, Gantt Charts, Task	as above	as above

Menu Path*	Command*	Description
(EP) View, Gantt Charts	Planned Purchase Orders	View Gantt chart of planned order deliveries sorted by start time.
(PL) View, Gantt Charts, Task	as above	as above
View, Gantt Charts, Resource	All Resource (PL)	View Gantt chart of all resources.
View, Gantt Charts, Resource	Selected Resource (PL)	View Gantt chart of selected resources.
View, Histogram Charts	All Inventory (EP/PL)	View histogram chart of all inventory.
View, Histogram Charts	Selected Inventory (EP/PL)	View histogram chart of selected inventory.
View, Histogram Charts	All Resource (PL)	View histogram chart of resource utilization.
View, Histogram Charts	All Aggregate Resource (EP)	View all aggregate resource usage.
View, Histogram Charts	Selected Resource (PL)	View histogram chart of selected resource utilization.
View, Histogram Charts	Selected Aggregate Resource (EP)	View selected aggregate resource usage.
View	Plan Overview (EP/PL)	View/edit capacity and inventory plan.
View	Forecast vs SO Trend (EP/PL)	View information about forecast versus sales orders.
View	Forecast Summary (EP/PL)	View planning periods and demand and forecast numbers for each production area.
View	Forecast Fulfillment	View Forecast fulfillment
View	Production by Area (EP/PL)	View/edit production by shift and production area.
View	Production by Item (EP/PL)	View production by item.
View	Item Sourcing (EP/PL)	View item sourcing.
View	Command File Errors	View/edit a downloaded error file.

Menu Path*	Command*	Description
View	Toggle Toolbar	Display toolbar and hide toolbar.
View	Toggle Status Bar	Display status bar and hide status bar.
Tools	Home Base (EP/PL)	Display Home Base.
(EP/PL) Tools	Supply Chain Viewer	Display the Supply Chain Viewer.
(OP) Tools, Advanced	as above	as above
(EP/PL) Tools	Data Browser	Display the Data Browser.
(OP) Tools, Advanced	as above	as above
(EP/PL) Tools	Model Browser	Display the Model Browser.
(OP) Tools, Advanced	as above	as above
Tools	Solver Settings and Run Control	Invoke several utilities, such as roll forward, late supplies inquiry, lateness tolerance global settings, material planning.
Tools	Control panel	Display the Control panel.
(EP/PL) Tools	Scorecard	Display the Scorecard.
(OP) Tools, Advanced	as above	as above
Tools, Advanced	Delete Orders/Tasks (OP)	Specify the completed or canceled orders/tasks you want to delete.
Tools	Optimize (EP/PL)	Invoke the Optimizer.
Tools, Optional	Control panel (EP/PL)	Display the Optional Control panel with 5 of the 14 constraint types.
Tools, Optional	Scorecard (EP/PL)	Display the Optional Scorecard with 5 of the 14 constraint types.
Tools	Build Query (PL)	Create an executable production schedule for selected business unit and resource.
Tools, Schedule	Dispatch List (EP/PL)	View/edit released production operations.

Menu Path*	Command*	Description
Tools, Schedule	Generate Schedule (EP/PL)	Generate a schedule from scratch.
Tools, Schedule	Generate Executable Schedule (EP/PL)	Generate executable schedule.
Tools, Prioritization	Prioritize (EP/PL)	Reschedule production and shipments based on prioritize and schedule rules.
Tools, Prioritization	Exceptions (EP/PL)	View prioritization sales order shipment exceptions that are chosen by selection rules and fall within the defined start and end dates, but can't be satisfied.
Tools, Prioritization, Clear	Clear All Exceptions (EP/PL)	Clear the prioritize exceptions only. Use to improve memory usage and performance.
Tools, Prioritization, Clear	Clear All Temporary Objects (EP/PL)	Remove all internal system objects that were created during the order prioritization run.
Tools	Capable to Promise (PL)	Determine when a prospective customer order can be promised.
Tools	Material Usage (EP/PL)	View material usage in a specified product and the maximum quantity that can be produced
Tools	Compare Plans (EP/PL)	Generate a comparison report between two selected plans.
Tools	Compare Data (EP/PL)	Create before and after pictures of planning process and generate rescheduling messages based on the comparison.
Tools, Rescheduled	Production Rescheduled (EP/PL)	Comparison report of production rescheduled during planning process.

Menu Path*	Command*	Description
Tools, Rescheduled	Purchase Order Rescheduled (EP/PL)	Comparison report of purchase orders rescheduled during planning process.
Tools, Rescheduled	Planned Purchase Order Rescheduled (EP/PL)	Comparison report of rescheduled purchase orders rescheduled.
Tools, Rescheduled	Transfers Rescheduled (EP/PL)	Comparison report of transfers rescheduled during planning process.
Tools, Rescheduled	Sales Order Rescheduled (EP/PL)	Comparison report of sales orders rescheduled during planning process.
Tools, Rescheduled	Resource Rescheduled (EP/PL)	Comparison report of resources rescheduled during planning process.
Tools, Exception Inquiry	Production Exceptions	View production options effectivity date exceptions.
Tools, Advanced, Histogram Charts	All Inventory (OP)	View histogram chart of all inventory.
Tools, Advanced, Histogram Charts	Selected Inventory (OP)	View histogram chart of selected inventory.
Tools, Advanced, Histogram Charts	All Resource (OP)	View histogram chart of resource utilization.
Tools, Advanced, Histogram Charts	Selected Equipment (OP)	Select a resource and view a histogram chart for that resource.
Tools, Advanced, Spreadsheets	Master Production Plan (OP)	View MPS information.
Tools, Advanced, Spreadsheets	Horizontal MRP (OP)	View horizontal MRP information.
Tools, Advanced, Spreadsheets	Capacity (OP)	View/edit factory capacity information.
Utilities, Delete Planned Purchase Orders	All Planned Purchase Orders (EP/PL)	Delete all planned orders.
Utilities, Delete Planned Purchase Orders	Unfirmed Planned Purchase Orders (EP/PL)	Delete all planned orders that don't have a firmed status.

Menu Path*	Command*	Description
Utilities	Delete Orders/Tasks (EP/PL)	Specify the completed or canceled orders/tasks you want to delete.
Utilities	Remap Orders/Tasks (EP/PL)	Specify mass changes of behavior associated with the remap flag on specified orders/tasks.
Utilities, Freeze Tasks	Freeze Forecasts (EP/PL)	Freeze/unfreeze forecasts.
Utilities, Freeze Tasks	Freeze Planned Purchases (EP/PL)	Freeze/unfreeze planned purchases.
Utilities, Freeze Tasks	Freeze PO Line Deliveries (EP/PL)	Freeze/unfreeze production order line deliveries.
Utilities, Freeze Tasks	Freeze Production (EP/PL)	Freeze/unfreeze production.
Utilities, Freeze Tasks	Freeze Transfers (EP/PL)	Freeze/unfreeze transfers.
Utilities, Freeze Tasks	Freeze SO Shipments (EP/PL)	Freeze/unfreeze sales order shipments.
Utilities, Freeze Tasks	Freeze Selected Tasks (EP/PL)	Freeze/unfreeze selected tasks.
Utilities, Change Sourcing Option on Planned Orders, Change Transfer Option on Planned Transfers	Change Source Unit/Transfer Option (EP/PL)	Change the transfer options for the source unit.
Utilities, Change Sourcing Option on Planned Orders, Change Transfer Option on Planned Transfers	Change Destination Unit/Transfer Option (EP/PL)	Change the transfer options for the destination unit.
Utilities, Change Sourcing Option on Planned Orders	Change Purchase Option on Planned Transfers (EP/PL)	Change purchase option on planned purchases.
Utilities, Change Sourcing Option on Planned Orders	Change Production Option on Planned Transfers (EP/PL)	Change production option on planned purchases.
Utilities	Roll Forward Utility (EP/PL)	Rolls forward past due orders and tasks to current time Production Planning.
Utilities	Recalculate Demand Time Fences (EP/PL)	Recalculates demand time fences when changes are made to current time.

Menu Path*	Command*	Description
Utilities	Regenerate Forecast Consumption (EP/PL)	Update forecasts that changed in the planning session.
Utilities	Consolidate Planned Purchase Orders (EP/PL)	Group planned purchase orders into one planned purchase order.
Utilities	Calculate Low Level Code (EP/PL)	Calculate low level code.
Window	Cascade	Cascade open windows.
Window	Arrange Icons	Sets buttons in a line at the bottom of the window.
Window	Close All Windows	Close all open windows.
Window	Settings	Saves form sizes and settings that are made in grids, graphical charts, and calendars.
Help	Help Topics	Display a list of help topics.
Help	Task Help	Display help for a task.
Help	Tech Support	Display how you reach PeopleSoft Technical Support.
Help	About	Display Production Planning version numbers for the server and client.

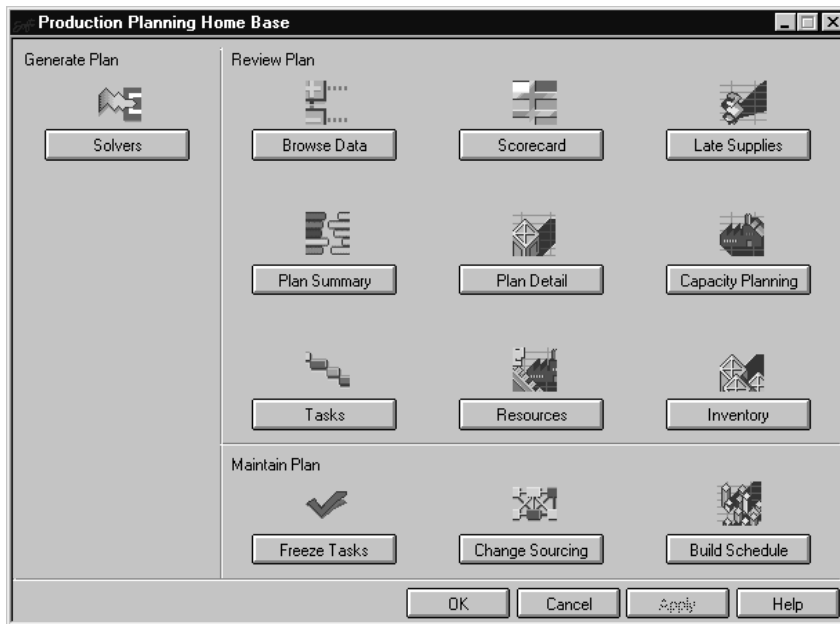
*A menu path preceded, or a command followed by "EP", "PL" or "OP" indicates a command that appears respectively in the Enterprise Planning, Production Planning or Order Promising client. Entries that include none of these appear in all three clients.

Using the Home Base Page



Home Base is not part of the Order Promising application.

Select **Tools, Home Base** to access the Home Base page. Home Base is a unique page in PepperTools applications; it contains sixteen command buttons arranged in four groups. Each button has an illustration above it that symbolizes the button's function. The buttons in each group are functionally related, and each button corresponds to a frequently used menu command. The commands aren't necessarily grouped the same way as on the menus. This page is a fixed size, but you can minimize it using its standard Windows minimize button.



Production Planning Home Base

The four functional groups in the page are called **Tools**, **Planner's Toolbox**, **Charts** and **Executive Overview**. The following table lists each button and its menu equivalent:

Home Base Menu Equivalents

<i>Group—Button*</i>	<i>Menu Equivalent*</i>
Generate Plan—Solvers	Solver Settings and Run Control
Review Plan—Browse Data	Tools, Data Browser
Review Plan—Scorecard	Tools, Scorecard
Review Plan—Late Supplies	Tools, Solver Settings and Run Control (Late Supplies Inquiry button)
Review Plan—Plan Summary	View, Spreadsheets, Plan Summary
Review Plan—Plan Detail	
Review Plan—Capacity Planning	View, Spreadsheets, Capacity
Review Plan—Tasks	View, Gantt Charts, Selected Tasks (EP); View, Gantt Charts, Task, Selected Tasks (PL)
Review Plan—Resources	View, Histogram Charts, Selected Aggregate Resource (EP); View, Gantt Charts, Resource, Selected Resource (PL)

Group—Button*	Menu Equivalent*
Review Plan—Inventory	View, Histogram Charts, Selected Inventory
Maintain Plan—Freeze Tasks	Utilities, Freeze Tasks
Maintain Plan—Change Sourcing	Utilities, Change Sourcing Option on Planned Orders
Maintain Plan—Build Schedule	View, Spreadsheets, Build Schedule (with Resource)

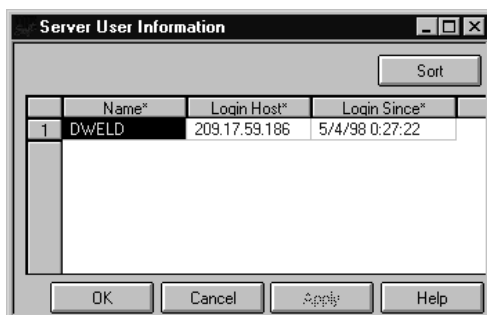
*A button name or menu equivalent followed by "EP" or "PL" indicates a button or command that appears respectively in the Enterprise Planning or Production Planning application. Entries that include neither of these appear in both applications.



For more information about the menu equivalents, see Viewing the Menu After Loading a Dataset in this section.

Understanding Server User Information

Use the Server User Information page to view users connected to your server. Select **File, Show Server Users** to display this page.



Server User Information Page

This page displays information about all users currently connected to the server. The **Name** column designates each user's login ID, the **Login Host** is the IP address from which the user is connecting, and **Login Since** shows the date and time the user started the current connection to the server.

Working with Bookmarks

Your PepperTools application enables you to create in-memory images called bookmarks that you use to save the state of your work for a quick return. You can use bookmarks to quickly compare schedules, for example.



Bookmarks are not available in the Order Promising application.

Bookmarks are in-memory images that enable "what-if" scenarios. You set, switch to, and delete bookmarks using either the Main Menu or transactions. A bookmark enables you to keep the current schedule by setting a mark in your PepperTools application. You return to a bookmark to compare schedules. When you return to a bookmark, the program returns to the state it was in at the time you set the bookmark.

If you return to a bookmark before setting a bookmark for the current state of the program, you lose the current information. For example, if you set a Bookmark at point A, continue to work on the schedule to Point B, and then decide to return to point A without setting a bookmark at Point B, you lose everything done after setting Bookmark A.



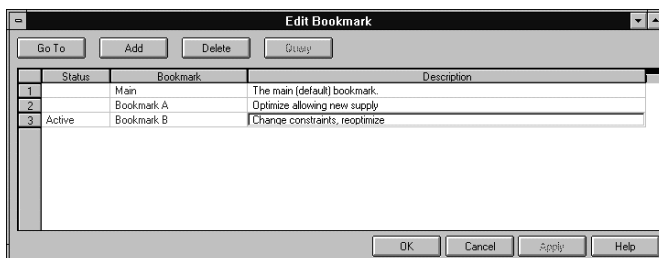
Important! Bookmarks are temporary. When you disconnect from a server, the bookmarks you created no longer exist. You have the option before disconnecting of going to each bookmark and saving it as a snapshot.



For more information, see ***Saving and Loading Files:*** Working with Snapshots.

Understanding the Edit Bookmark Page

To display the Edit Bookmark page, select **Edit, Bookmark**.



Edit Bookmark Page

The Edit Bookmark page contains the following command buttons:

Button	Function
Go To	Switch to a previous bookmark.
Add	Set a new bookmark.
Delete	Delete a bookmark.

Button	Function
Query	Refresh the bookmark list after a bookmark is added.
OK	Apply all changes and close the Edit Bookmark page.
Cancel	Ignore any changes and close the Edit Bookmark page.
Apply	Apply all changes but leave the Edit Bookmark page open.
Help	Invoke context-sensitive online documentation.

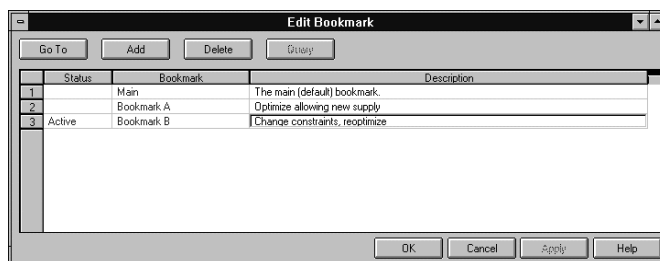
It also contains the following columns:

Column	Description
Status	State of the bookmark. There is only one status, the <i>active</i> status. It indicates that the bookmark is currently being used.
Bookmark	Name of the bookmark.
Description	Description of the bookmark.

Creating a Bookmark

To create a new bookmark:

1. Select **Edit, Bookmark** to display the Edit Bookmark page. Bookmarks you previously set appear in the page.



Edit Bookmark Page Showing Previously Set Bookmarks

2. To add a new bookmark, click **Add**.



The image shows a 'Set Bookmark' dialog box. It has a title bar with a close button and the text 'Set Bookmark'. Inside, there are two text input fields: 'Bookmark name:' with the text 'Bookmark A' and 'Bookmark description:' with the text 'Optimize allowing new supply'. At the bottom right, there are four buttons: 'OK', 'Cancel', 'Apply', and 'Help'.

Set Bookmark

3. Type a bookmark name in the **Bookmark name** field.



You can't use the same name an existing bookmark uses.

4. Type an appropriate description in the **Bookmark description** field.
5. Do either of the following:
 - Click **OK** to set the bookmark and close the page, or
 - Click **Apply** to set the current bookmark without closing the page. You can then set multiple bookmarks by deleting the current name and repeating steps 3 through 5 as many times as you want.
6. To refresh the bookmark list, click **Query**. The bookmarks you set will appear in the list.

Switching Between Bookmarks

You compare bookmarks by switching between two or more previously saved bookmarks. Be sure you set a bookmark for any changes you make to a schedule before switching to another bookmark. If you don't set a bookmark, those changes are lost.

Switching to another bookmark so you can compare schedules:

1. Select **Edit, Bookmark** to display the Edit Bookmark page.
2. Click the **row number** of the bookmark to which you want to switch. Its row is highlighted.
3. Click **Go To** to switch to the highlighted bookmark. A message box appears asking you for confirmation that you want to go to the selected bookmark.



If you made changes since you set the current bookmark, the program forgets those changes when you switch to another bookmark.

4. The state of the server resets to its state at the time you set the selected bookmark. The program closes all windows and displays the Home Base page.

Deleting a Bookmark

To delete a bookmark:

1. Select **Edit, Bookmark** to display the Edit Bookmark page.
2. Click the *row number* of the bookmark you want to delete. Its row is highlighted.
3. Click the **Delete** button. The program prompts you for confirmation, then removes the bookmark from the list.



If the bookmark you're deleting is the active one, the program makes the earliest bookmark on the list active, then deletes the one you selected.

CHAPTER 6

Using Graphic Charts

Your PepperTools application uses both Gantt charts and histogram charts. A Gantt chart displays a schedule with tasks represented by horizontal bars extending along a timeline. A histogram is a type of bar chart that displays a quantity or state over time. The program uses histograms to display resources, inventory, and attributes.

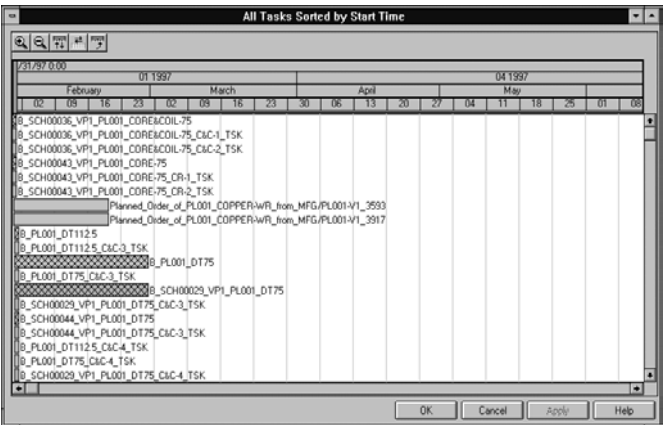
Accessing Gantt Charts and Histograms

You can have multiple Gantt charts and histograms open at the same time.

To access Gantt charts in Enterprise Planning, select **View, Gantt Charts**; in Production Planning, select **View, Gantt Charts, Task** or **View, Gantt Charts, Resource**. The program displays a submenu listing the available Gantt charts. Choose the chart you want to display. You can also access Gantt charts via popup menus by right-clicking any item, resource or task.



Gantt charts in Order Promising are only available via popup menus.



Production Planning All Tasks Gantt Chart

To access histograms, in Enterprise Planning or Production Planning select **View, Histogram Charts**; in Order Promising select **Tools, Advanced, Histogram Charts**. The program displays a submenu of available histogram charts. Choose the histogram chart you want to display.

Exploring a Graphic Chart Page

The graphic chart page consists of four main areas:

- The chart toolbar at the top of the page.
- The chart timeline.
- The main work area, which includes a Gantt chart, a histogram, or both; plus a vertical scrollbar for each chart and a horizontal scrollbar for the entire display.
- The standard command buttons at the bottom of the page.




The timeline and any charts displayed are aligned with respect to time; scrolling left or right will shift these components in your field of view simultaneously.



Understanding the Chart Toolbar and Timeline



Graphic Chart Toolbar

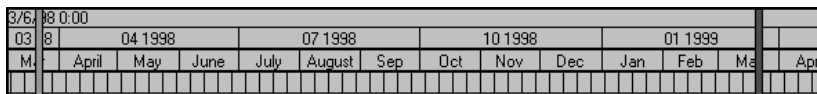
The graphic chart toolbar contains the following buttons:

Button	Command	Description
	Zoom In	Increase the horizontal magnification of the timeline and chart(s) by approximately 2x. The display accommodates a shorter time period, showing more detail.
	Zoom Out	Decrease the horizontal magnification of the timeline and chart(s) by approximately 2x. The display accommodates a longer time period, showing less detail.
	Calendar Scale	Choose the smallest time interval on the timeline. It can be one <i>Hour</i> , one <i>Shift</i> , one <i>Day</i> , one <i>Week</i> , or one <i>Month</i> , and a shift can be set at 4, 6, 8 or 12 hours.

Button	Command	Description
	Move Granularity	Set the minimum increment of movement when dragging a task. It can be <i>None</i> (determined by mouse movement), <i>Minutes</i> , <i>1/2 Hour</i> , <i>1 Hour</i> , <i>2 Hours</i> , one <i>Shift</i> , one <i>Day</i> , one <i>Week</i> , or one <i>Month</i> . The default is one <i>Hour</i> .
	Go To	Scroll the timeline and chart(s) to the date and time specified, positioning that point at the left edge of the display.



If **Move Granularity** is set to a smaller increment than the mouse pointer can distinguish, the move increment will be determined by the smallest amount of pointer movement.



Production Planning Graphic Chart Timeline

The graphic chart timeline is just below the toolbar, and has the following features:

- The earliest visible date and time in the upper left hand corner.
- Marks and labels for the current **Calendar Scale** intervals, and the two next larger time intervals.
- Marks for the early and late fences. The early fence is green, and the late fence is red.

You can drag the fence marks with your mouse, in increments as small as the mouse pointer can distinguish. This movement isn't governed by the Move Granularity setting.

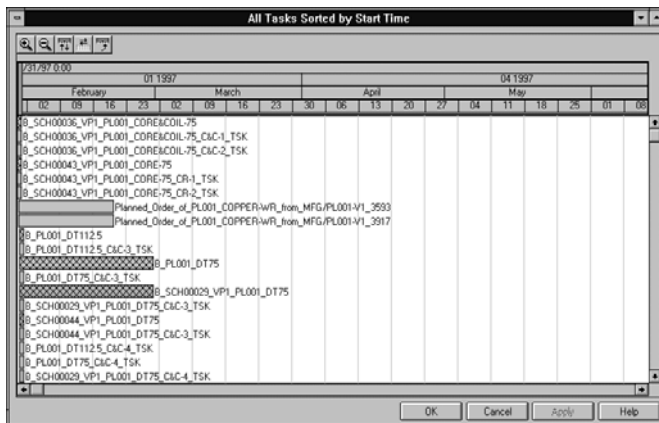
Understanding Chart Types and Features

Each type of chart in your PepperTools application has a different set of graphical and functional components, which enable you to navigate the chart, modify its values and analyze the data it presents. Following is an overview of these components.

Reading Gantt Charts

You can use Gantt charts to assess your current schedule, review the status of tasks, and make changes to tasks. A Gantt chart is a graphical representation of tasks in a timeline. Each task is represented by a horizontal **task bar** labeled with the name of the task, extending along the portion of the timeline that covers the duration of that task.

Your PepperTools application sorts the task bars on the Gantt chart from top to bottom, from earliest to latest start time. When tasks with the same start time have different durations, the program sorts them from shortest to longest.



Production Planning All Tasks Gantt Chart

Several tasks may be combined into a group consisting of one **parent task** and one or more **subtasks**. When you change the start time or duration of one task in a group, the program automatically alters the other tasks so they have the same start time, then resorts the group on the chart. When two or more groups have identical start times, the individual tasks are sorted by duration, so the groups may be intermingled.

Some Gantt charts display in combination with a histogram chart, which enables you to easily scroll through tasks and see where the resource or inventory is in relation to the schedule.



For more information about modifying tasks, see Working with Tasks in a Gantt Chart in this section. For more information about combination charts, see Exploring Combination Charts in this section.

Understanding Task Bar Features

The shape and color coding of task bars gives you information about the status and type of tasks. The following table outlines the features common to all task bars:

Feature	Meaning
Left edge of bar	Start time of the task.
Right edge of bar	End time of the task.
Length of bar	Duration of the task.
Text	Task name.
Cyan (very light blue)	Frozen task. You've frozen this task in time; the program can't move it.
Cross hatch pattern	The task is a parent.
Vertical line	The task is wrapped.

Understanding Task Bar Colors

The task bar colors represent various status conditions. The following table lists the default color assignments for every possible status setting in your PepperTools application. Each task class uses a subset of the status conditions shown. "All" refers to all classes to which the listed status applies.

Task Status Color Coding

Task Bar Color*	Task Class	Status
Bright Green	Shipment_Parent	<i>In Process</i>
	Base_Task	<i>Pending</i>
	Production_Parent, Transfer_Parent	<i>Released</i>
	all other classes	<i>Open</i>
Yellow	Downtime_Task	<i>Open</i>
Red	all	<i>Canceled</i>
Magenta	Base_Task	<i>Active</i>
	Production_Parent	<i>In Process</i>
	Transfer_Parent	<i>In Transit</i>
Primary Blue	all	<i>Firmed</i>
Dark Green	all	<i>Complete</i>
Gray Blue	all	<i>Planned</i>
Purple	all	<i>Dispatched</i>

* If a task is frozen, its color is Cyan (very light blue) regardless of the status you set for it.

Reading Histogram Charts

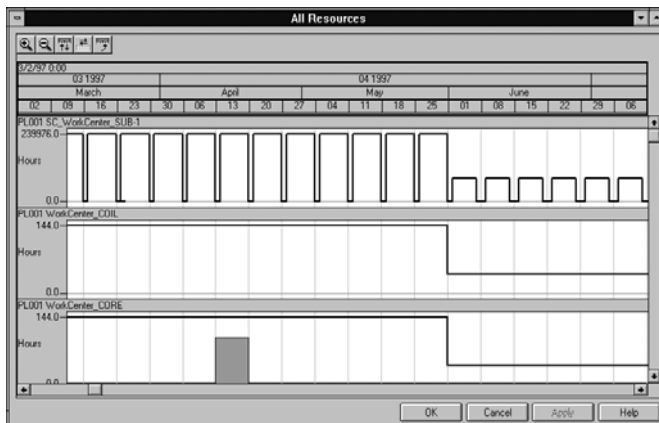
Histogram charts graphically display information about resource utilization and item inventory shortfall and excess along the timeline of the schedule. They display both history and the projected future based upon the current plan.

There are two types of histogram charts:

- Resource charts, which display how much of each resource is available, how much is being used, and whether or not it's overloaded.
- Inventory charts, which display available inventory as well as shortages.

Viewing a Resources Histogram

Resources histogram charts display how much of each resource is available—the capacity, how much is being used—the load, and whether or not it's overloaded.



Production Planning All Resources Histogram

The histogram displays the following:

Capacity

Displays as a black line during the time the resource is on the calendar. The height of the black line represents magnitude of the capacity.

Load

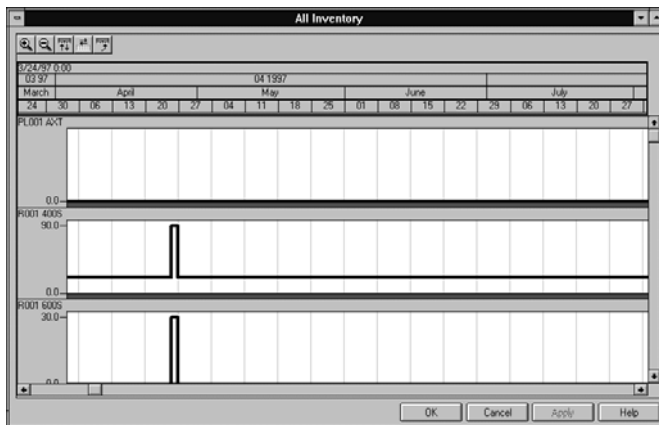
Displays as a green bar below the black line during the time the load is applied. The height of the green bar represents the magnitude of the load.

Overload

When demand for the resource exceeds its capacity, displays a red bar extending above the black line shown during the time the excess demand is applied. The height of the red bar represents the magnitude of the demand. The vertical scale of the graph changes to match the largest allocation.

Viewing an Inventory Histogram

Inventory histogram charts display item inventory status shortages.



Production Planning All Inventory Histogram

The histogram displays the following:

Inventory in stock

Available inventory—produced but not yet consumed—is shown as a black line on the histogram. Overallocation (inventory shortages) is shown by the black line extending below the zero level.

Safety stock

Safety stock targets are shown as a red bar on the histogram.

Excess stock

Excess stock targets are shown as a blue bar on the histogram.

No excess/no shortage

When an item is consumed as soon as it is produced or delivered—just in time—the item is never counted as inventory so the inventory histogram shows zero inventory change for that item for that time. Zero on the chart can also indicate zero inventory and zero demand for that item for that time.

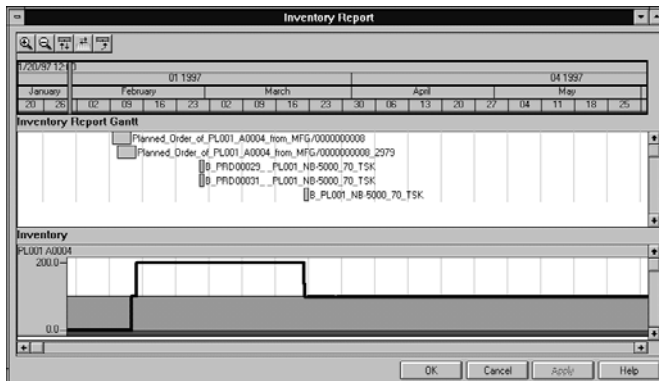
Exploring Combination Charts

A combination chart is a report that combines a Gantt chart and a histogram. Combination charts display functional problems within certain areas of the schedule. The window contains multiple graphs and charts appropriate to the problem being illustrated. PepperTools provides three types of combination charts, which are accessed only via popup menus:

- The Inventory Report.
- The Resource Report.
- The Task Detail.

Accessing an Inventory Report

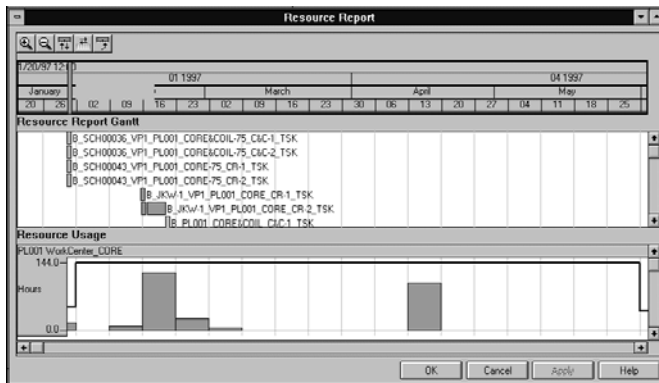
Right-click an item in the Data Browser to display a popup menu, and select **Inventory Report** to access an Inventory Report for that item. The Inventory Report is a combination Gantt chart/histogram that displays inventory vs. time and all producing and consuming operations for the selected item.



Inventory Report

Accessing a Resource Report

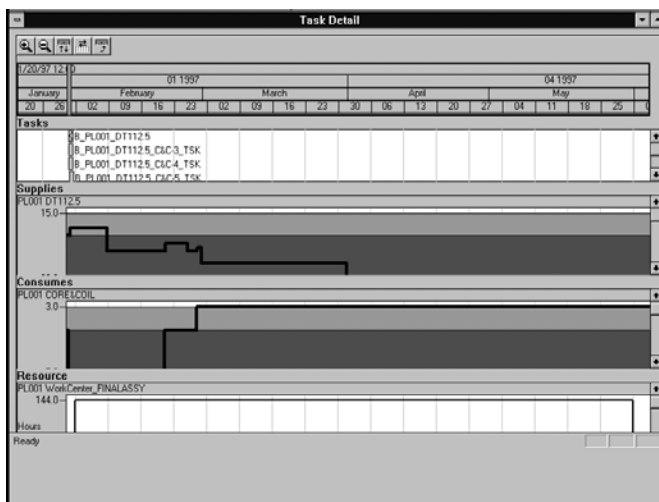
Right-click a resource in the Data Browser to display a popup menu, and select **Resource Report** to access a Resource Report for that resource. The Resource Report is a combination Gantt chart/histogram that displays resources vs. time and all operations utilizing the selected resources.



Resource Report

Accessing Task Detail

Right-click a task bar in a Gantt chart to display a popup menu and select **Show Task Detail** to access the Task Detail for that task. Task Detail is a combination Gantt chart/histogram that displays the task vs. time and all producing and consuming resources needed for the selected parent task.



Task Detail

Navigating Gantt Charts and Histograms

Gantt charts and histograms are often larger than a single screen, so only part of the schedule appears. You have several tools for moving around in the Gantt chart or on a histogram:

- Use the scroll bars to move the window left and right along the timeline and up and down through the task display or individual histogram charts.
- Use the toolbar **Zoom In** and **Zoom Out** buttons to show more or less time on the timeline.

- Use the toolbar **Calendar Scale** button to change the calendar display to hours, shifts, days, weeks, or months.
- Use the toolbar **Go To** button to move to a specified date and time.



For more information about using the chart toolbar, see Exploring a Graphic Chart Page in this section.

- Use the popup menu in a Gantt chart to go directly to any part of the schedule you want to see.

Navigating with the Gantt Chart Popup Menu

You can right-click anywhere in a task Gantt chart to display a popup menu. If the pointer is over the chart background, a short version of the menu, the Gantt chart popup menu, appears. If the pointer is over a task, a long version, the task popup menu, appears with task-specific commands. Both display the following commands that enable you to navigate to specific tasks:

Command	Description
Find Task by Name	Display a dialog for entering a search string, and go directly to the task named.
Find Selected Task	Go directly to the currently selected task.
Scroll to Start Time	Scroll to the start time of the specified task.*
Scroll to End Time	Scroll to the end time of the specified task.*

* The specified task is the one on the same line as the point where you right-clicked.



These commands aren't available in Production Planning resource Gantt charts.



For more information about the task popup menu, see Working with Tasks in the Gantt Chart: Using the Task Popup Menu in this section. For more information about resource popup menus, see Working with a Resource Gantt Chart in this section.

Working with Tasks in a Gantt Chart

In the Gantt chart you can make the following changes:

- You can modify subtasks, but you can't modify parent tasks directly. When you modify the start time, end time, or duration of a subtask, the parent task automatically changes as well to reflect that modification.
- You can modify the start time of a task directly on the chart by dragging its task bar with the mouse.
- You can modify the duration of a task directly on the chart by lengthening or shortening its task bar with the mouse.
- You can modify tasks using the task popup menu.

Modifying Tasks with the Mouse

Click the task bar of the task you want to work with. The task bar is highlighted by a rectangle to indicate that it's selected, with square sizing handles at the left and right edges.



When you select a task, the previously selected task is no longer selected. You can only select one task at a time.

To change the start time of a selected task, but not its duration:

1. Move the mouse pointer over the center of the selected task bar. A four-arrow cursor appears, and a popup information box shows the task's name as well as its start and end time.
2. Click and drag the task bar to the left or right. A shadowed rectangle appears as you drag the task bar, and the start time changes to reflect its changing position.
3. Release the mouse button when the start date and time are satisfactory. Because the task has a new start time, the program may need to resort order in the chart.

To change the duration of a selected task:

1. Move the mouse pointer over the sizing handle at the left or right edge of the selected task bar. A two-arrow cursor appears, and a popup information box shows the task's name as well as its start and end time.
2. Click and drag the task bar's edge to the left or right. A shadowed rectangle appears as you resize the task bar, and the start or end time changes as you drag.
3. Release the mouse button when the new duration is satisfactory.



It's not actually possible to modify the end time of a task this way. When you drag and release the right edge of the task bar, the program shifts the entire task bar to maintain the original end time, thus changing the **start time** by an equal but opposite amount. For example, if you enlarge the task bar by dragging its right edge to the right, the program will shift the entire task bar an equal distance to the left, resulting in a new **duration** and **start time**, but the same **end time** as before.

Because the task has a new start time, the program may reposition it to maintain the correct top-to-bottom sort order in the chart.



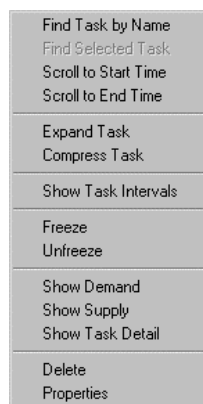
When you change the start time or duration of one task in a group, the program automatically alters the other tasks so they have the same start time, then resorts the group on the chart.

Using the Task Popup Menu

The **task popup menu** enhances your ability to view and modify tasks. Position the cursor over the task you want, and right-click to produce the menu. Choose the option you want from the menu. The top four commands are the same ones that appear on the general Gantt chart popup menu.



For more information about the four navigation commands at the top of the task popup menu, see *Navigating Gantt Charts and Histograms: Navigating with the Gantt Chart Popup Menu* in this section.



Task Popup Menu

The task popup menu provides a fast way to view and modify tasks. It also provides a way to get to the End Item Pegged Demand page, Task Supply page, and Task Detail chart. The Gantt chart changes to reflect your edits.

Use the following menu items to view and modify tasks:

Command	Description
Expand Task	Expand the task.
Compress Task	Compress the task.
Show Task Intervals	Display the intervals covered by the task, with their start and end times.
Freeze	Freeze the task.
Unfreeze	Unfreeze the task.
Show Demand	Show how demand affects the task in the End Item Pegged Demand page.
Show Supply	Show how supply affects the task in the Task Supply page.
Show Task Detail	Show supply and demand detail about the task in the Task Detail combination Gantt/histogram chart.
Delete	Delete the task.
Properties	Display the property sheet for the selected task.

Expanding and Compressing Tasks

Expanding a task means moving the task as late as possible in the schedule without violating any time or capacity constraints. This moves the task no later than the late time fence.

To expand a task:

1. Position the cursor on the task you want to expand and right-click to display the task popup menu.
2. Select Expand Task.

The task's interval parameters are adjusted to reposition the task as close as possible to the late time fence, and the task bars are resorted according to the new values.

Compressing a task means moving the task as early as possible in the schedule without violating any time or capacity constraints. This moves the task no earlier than the early time fence.

To compress a task:

1. Position the cursor on the task you want to compress and right-click to display the task popup menu.
2. Select Compress Task.

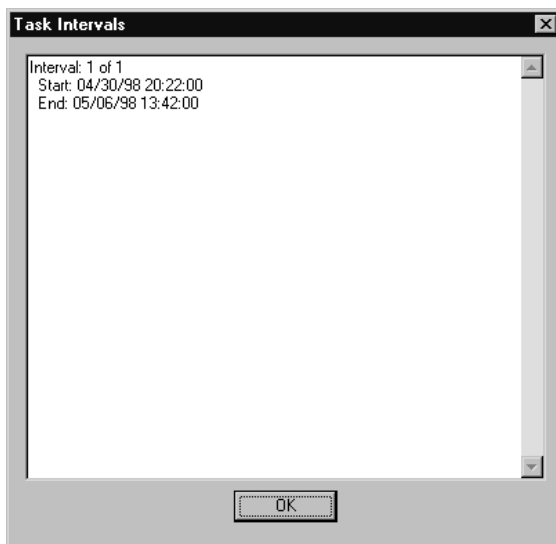
The task's interval parameters are adjusted to reposition the task as close as possible to the early time fence, and the task bars are resorted according to the new values.

Viewing Task Intervals

You can view details about the intervals comprising a task with the Task Intervals page.

To view interval information for a task:

1. Position the cursor on a supplying task you want to examine and right-click to display the task popup menu.
2. Select **Show Task Intervals**. The Task Intervals page appears, showing the start date and time and end date and time for each interval.



Task Intervals Page



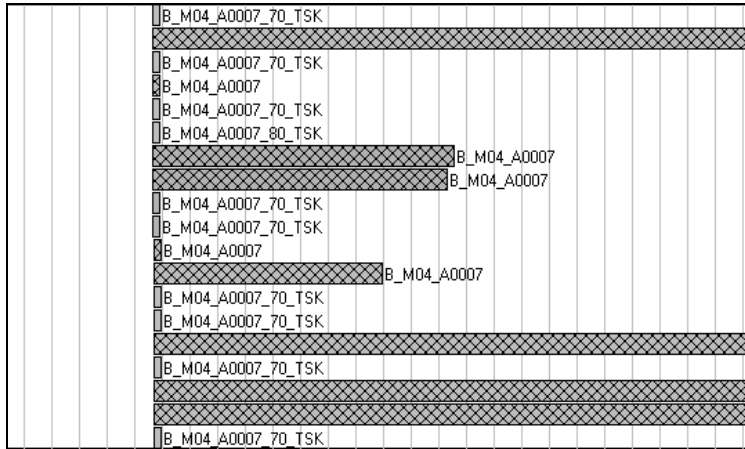
Multiple intervals occur only in wrapped tasks, which are indicated by a vertical line through the task bar.

Freezing and Unfreezing Tasks

Freezing a task fixes the task at its current point on the timeline. On a Gantt chart, the task bar color turns cyan (very light blue) to indicate a frozen task. The program is prevented from moving or altering the frozen task when you move an associated task or when you optimize the schedule. Unfreezing a task removes the restrictions imposed by freezing.

To freeze a task:

1. Position the cursor on the task you want to freeze and right-click to display the task popup menu.
2. Select Freeze Task.



Two Frozen Tasks

To unfreeze a task:

1. Position the cursor on the task you want to unfreeze and right-click to display the task popup menu.
2. Select Unfreeze Task.

Viewing Demand for a Task

You can view how demand (sales orders) affects tasks in the End Item Pegged Demand page.

To view demand for a task:

1. Position the cursor on a supplying task you want to examine and right-click to display the task popup menu.
2. Select **Show Demand**. The End Item Pegged Demand page appears.

End Item Pegged Demand

Task: B_M04_A0007_70_TSK

Sales Orders

Sort

	Unit*	Sales Order*	Line*	Unit*	Item*	Quantity*	UOM

Forecast

Sort

	Unit*	Item*	Period Start*	Period End*	Quantity*	UOM*
1	M04	A0007	5/1/98 0:00:00	5/2/98 0:00:00	1000.0	EA

OK

Cancel

Apply

Help

End Item Pegged Demand Page

Viewing Supply for a Task

You can view how supply (purchase orders, production orders, planned orders, and forecasts) affects a task in the Task Supply page.

To view supply for a task:

1. Position the cursor on the parent task you want to examine and right-click to display the task popup menu.
2. Click **Show Supply**. The Task Supply page appears.

[illegible]

Task Supply Page

Viewing Task Detail

You can view a combination Gantt/histogram chart that contains supply and demand detail about a selected task.



For more information, see Understanding Chart Types and Features: Exploring Combination Charts: Accessing Task Detail in this section.

Viewing Task Properties

To view task properties:

1. Position the cursor on the task you want to examine and right-click to display the task popup menu.
2. Select **Properties**. The property sheet for the task appears.

Task Property Sheet

You can view and edit information on the task property sheet just like any other page, including name, start time, end time, duration, status, description and freeze condition.



Edit task properties with caution.

To change the status of a task:

1. Click the **Status** drop-down list and select one of the status conditions from the list.
2. Click **Apply** or **OK** to apply the change.

Depending on the class of a task, you can change its status to *Active*, *Canceled*, *Complete*, *Dispatched*, *Firmed*, *In Process*, *In Transit*, *Open*, *Pending*, *Planned* or *Released*. When you apply your changes, the program will change the task bar color to reflect its new status.



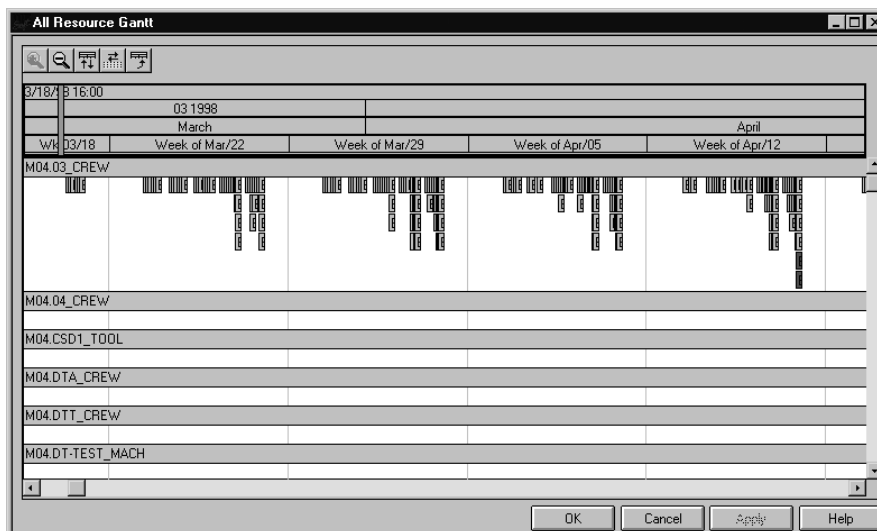
When the status of one of the subtasks in a group changes, the program automatically makes appropriate changes in the status of its parent tasks.



For more information about task bar status colors, see Understanding Chart Types and Features: Reading Gantt Charts: Understanding Task Bar Colors in this section.

Working with a Resource Gantt Chart

Select **View, Gantt Charts, Resource, All Resources** or **Selected Resource** in Production Planning to view a resource Gantt chart.



All Resource Gantt Chart with Cascaded Tasks



Resource Gantt charts are not available in Enterprise Planning or Order Promising, except as part of a Resource Report combination chart.

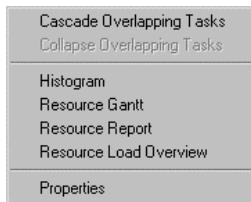
You can display either a resource popup menu or a task popup menu on the resource Gantt chart. A resource popup menu enables you to view resource histograms, resource Gantt charts, resources reports, and resource load overview as well as cascade and collapse overlapping tasks. A task popup menu enables you to inspect and modify tasks.



For more information about the task popup menu, see *Working with Tasks in a Gantt Chart* in this section.

Using the Resource Popup Menu

To open the resource Gantt chart popup menu, position the cursor on the gray horizontal bar containing the name of the resource you want and right-click. Select the option you want from the menu.



Resource Gantt Chart Popup Menu

You can access the following Resource Gantt chart popup menu options:

Command	Description
Cascade Overlapping Tasks	Move overlapping tasks—red bars—to a separate line.
Collapse Overlapping Tasks	Move the cascaded tasks back to the single task line.
Histogram	Display a Resources histogram for the selected resource.
Resource Gantt	Display a Resource Gantt chart for the selected resource.
Resource Report	Display the Resource Report for the selected resource.
	Display the Resource Load Overview for the selected resource.
Properties	Display the property sheet for the selected resource.

Cascading and Collapsing Tasks

You can cascade and collapse tasks to easily view overlapping tasks. Overlapping tasks appear in red on the resource Gantt chart.

To cascade tasks, right-click the gray horizontal bar containing the name of the resource with overlapping tasks, and select **Cascade Overlapping Tasks**. The program shifts all overlapping tasks to a line below the tasks they overlap.

To collapse tasks, right-click the gray horizontal bar containing the name of the resource with overlapping tasks, and select **Collapse Overlapping Tasks**. The program shifts all overlapping tasks up to the same line as the tasks they overlap. All tasks are displayed on the same line.

Dragging Tasks from One Resource to Another

You can drag a task from one resource to another resource on a resource Gantt chart.



Be sure that the resources are of the same class at the same unit.

To drag a task from one resource to another:

1. In Production Planning, select **View, Gantt Charts, Resource, All Resources** to display the All Resources Gantt chart.
2. Click the task you want to move. The task is highlighted with a rectangle and sizing handles to indicate that you selected it.
3. Move the mouse over the selected task. A four-arrow cursor appears and the task's name appears in a popup information box along with the start and end time of the task.
4. Drag the task to another resource. A shadowed rectangle appears as you drag the task.

Using the Histogram Chart Popup Menu

The histogram chart popup menu enables you to get additional information on availability, shortage, or order quantities. You can access the Item or Resources pages from the histogram chart popup menu.

To open the histogram popup menu:

1. Position the cursor on the resource or item on which you want to operate. For example, position the cursor in the exact green or red area on which you want to get additional quantity information.
2. Right-click to display the histogram chart popup menu.
3. Choose the option you want from the menu.



All Resources Histogram Popup Menu

The histogram chart popup menu includes combinations of the following commands, where they apply:

Command*	Description
Show Line Quantity	Show the availability or shortage level of the resource in the region selected. Note: Resources Resource Utilization History quantity is always displayed as 0.0000. This applies to the current release.
Show Bar Quantity	Show the availability or shortage level of the resource load in the region selected.
Histogram (inventory)	Display the Inventory histogram for the selected item.
Histogram (resource)	Display the Resources histogram for the selected resource.
Inventory Report (inventory)	Display the Inventory Report for the selected item.
Resource Gantt (resource)	Display the Resource Gantt for the selected resource.
Resource Report (resource)	Display the Resource Report for the selected resource.
Resource Load Overview (resource)	Display the Resource Load Overview for the selected resource.
Properties	Display the property sheet for the selected resource.

*If you right-click a task bar, the top command on the popup menu is **Show Bar Quantity**; otherwise it's **Show Line Quantity**. Commands applicable to just inventory or to just resources are noted as such.



The resource Gantt chart is not available in Enterprise Planning.

Closing Gantt Charts and Histograms

You don't have to close one Gantt chart or histogram to open another. Depending on memory limitations, you can have multiple Gantt charts and histograms open at the same time. When you're finished using a Gantt chart or histogram, click the **OK** or **Cancel** button.

CHAPTER 7

Using Spreadsheets

Your PepperTools application includes a wide variety of spreadsheets that you can use to view and edit scheduling information by shift, day, week, or month. A spreadsheet is a component containing several grid pages; some spreadsheets also contain one or two data entry pages.

Accessing Spreadsheets

You can have multiple spreadsheets open at the same time.

To access spreadsheets in Enterprise Planning and Production Planning, select **View, Spreadsheets**; in Order Promising select **Tools, Advanced, Spreadsheets**. The program displays a submenu listing the available spreadsheets. Choose the spreadsheet you want to display.

Unit	Inventory_Item	Row Type	Past Due	3/20/98	3/21/98	3/22/98	3/23/98	3/24/98	3/25/98
M04	20001	Net Forecast	0.00	0.00	0.00	0.00	0.00	0.00	0
M04	20001	Customer Orders	0.00	0.00	0.00	0.00	0.00	0.00	0
M04	20001	Dependent Demand	0.00	0.00	0.00	0.00	0.00	0.00	0
M04	20001	Total Demand	0.00	0.00	0.00	0.00	0.00	0.00	0
M04	20001	Total Supply	0.00	0.00	0.00	0.00	0.00	0.00	0
M04	20001	Planned On Hand	0.00	0.00	0.00	0.00	0.00	0.00	0
M04	20100_CFG:PC5200 B	Net Forecast	0.00	0.00	0.00	0.00	0.00	0.00	0
M04	20100_CFG:PC5200 B	Customer Orders	0.00	0.00	0.00	0.00	0.00	0.00	0
M04	20100_CFG:PC5200 B	Dependent Demand	0.00	0.00	0.00	0.00	0.00	0.00	0
M04	20100_CFG:PC5200 B	Total Demand	0.00	0.00	0.00	0.00	0.00	0.00	0

Production Planning Master Production Plan Spreadsheet

Each spreadsheet is arranged as a component. Initially the primary grid is empty; click the **Query** button to populate it with current data.

Exploring a Spreadsheet Component

The spreadsheet component consists of three main areas:

- The page header at the top of the component.
- The individual pages, with page-specific buttons, check boxes and parameter fields.
- The standard command buttons at the bottom of the component.



In the page header you can specify the **Start Date**, **Number Of Periods**, and **Period Duration**. For more information, see Working with Spreadsheet Pages: Modifying Date Columns in this section.

Understanding Spreadsheet Pages

Production Planning, Enterprise Planning and Order Promising each provide a different selection of spreadsheets, but the pages that comprise these components come from the seven in the following table, which lists each page name, the spreadsheet component(s) it appears in, and a short description:

Spreadsheet Pages

Page	Appears In	Description
Spreadsheet	All components	Primary data display grid. Includes a Query button and a Print Report button.
Row Type Setup	All components	Grid used to specify what row types are displayed, as well as their sort order based on type.
Key Setup		Grid used to filter the display of rows, and their sort order, based on their keys. The page header includes an identical grid that can also be used for filtering.
Summary Setup		Grid used to configure the inclusion of summary information about each inventory item or resource per row type.
Row Type Total		Secondary data display grid. Displays current totals for each row type in each time period. Includes a Query button and a Print Report button.

Page	Appears In	Description
Adv Inventory Item Filter		Data entry page. Sets item filter criteria, including Inventory Item, Item Description, Planner Code, Item Category 1 Attribute, Item Category 2 Attribute, and MPS Type.
Adv Base Item Filter	Material Movement	Data entry page. Sets item filter criteria. The search criteria includes Base Item, Item Description, Planner Code, Item Category 1 Attribute, Item Category 2 Attribute, and MPS Type.

Viewing Page Grids

Each page grid includes one or more of the columns listed in the following table, plus a **Past Due** column and a series of dated columns.

Spreadsheet Columns*

Column Name	Description
Production option	Name of the production option.
From Item	Name of the inventory item at the From Unit.
From Unit	Name of the From Unit of the transfer.
Inventory Item	Name or number of the item.
Resource	Name of the resource.
Row Type	Name of the row type.
To Item	Name of the inventory item at the To Unit—after the transfer.
To Unit	Name of the To Unit of the transfer.
Transfer Option	Name of the transfer option used for this route.
Unit	Name of the unit.

*All the columns listed are sort keys, except for the Row Type column.

The Past Due column shows the quantities that applied before the date of the first date column. Each date column represents the start of a planning period that you can adjust. Each row is a unique combination of values for the column types, showing quantities for that combination.



If the Spreadsheet grid isn't wide enough to display all the date columns, it provides a horizontal scroll bar. When you use the scroll bar, only the date columns will scroll.



All the cells in the Spreadsheet and Row Type Total grids are read-only; they can't be edited directly.

Working with Spreadsheet Pages

Schedulers use spreadsheets to display and edit data on production, build schedules, supply and demand, capacity utilization, production costs, revenues, and profits. The two primary display pages are Spreadsheet and Row Type Total. You can modify many parameters to customize the results shown on these two pages, such as the start date, number of periods, period duration, row types to display, sort key filtering and ordering, inventory item and base item filtering, and column width.

Using Spreadsheet Page Features

The following notes apply to all spreadsheet components:

- Whenever you modify parameters in the current spreadsheet component, click the **Query** button on the Spreadsheet or Row Type Total page to generate new results that reflect your changes.
- Whenever you modify sort options, filters, summary information or the number of rows to display, you must make the server aware of those changes. Press the **Tab** key to complete the setting, then click the **Apply** button, which should then be available. If you don't apply the changes, the program will notify you when you try to select a different page. Click the **Query** button on the Spreadsheet or Row Type Total page to generate new results that reflect your changes.
- You can export the current results displayed on the Spreadsheet or Row Type Total page to a text file. Click the **Print Report** button, then enter the output filename, choose whether you want to overwrite the file or append to it, and choose whether you want to save the file and print it, or just save it. When you click the **OK** button, it will be saved to the working directory—from which you launched the program.
- Except for the date columns, you can change a grid column's width by positioning the mouse pointer at the right edge of the column and dragging left or right to respectively narrow or widen the column.



When you resize the **Past Due** column, all the date columns are resized to match it. They can't be resized individually.

Modifying Date Columns

Using the page header fields you can control how much time is covered in the spreadsheet, how it's divided, and when it starts.

Setting a Start Date

You can set a starting point from which the spreadsheet columns are calculated. To set this start date, type the date in the **Start Date** field at the top of the spreadsheet and press the **Tab** key.

Setting the Number of Periods

The number of periods is the number of displayed date columns. To set the number of periods, type the number in the **Number of Periods** field at the top of the spreadsheet and press the **Tab** key.

Setting the Period Duration

You can choose a period duration, such as shifts, days, weeks, or months, to use in a spreadsheet. To choose a period duration, click the down arrow in the **Period Duration** drop-down list and then choose the period duration you want: *S_SHIFT*, *S_DAY*, *S_WEEK*, *S_MONTH*.

Choosing Rows to Display

You can use the Row Type Setup page to choose which output variables you want to display in the spreadsheet. You can also use this page to specify the order in which the row types are displayed, and the maximum number of rows to display.

To manage row types, select the **Row Type Setup** page.

Spreadsheet | **Row Type Setup** | Key Setup | Summary Setup | Row Type Total | Adv Inventory Item Filter

Add Delete

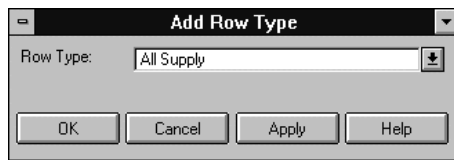
	Row Type	
1	Unconsumed Forecast Demand	Up
2	Actual Demand	Down
3	Dependent Demand	Top
4	All Demand	Bottom
5	All Supply	
6	Projected Available	

Maximum Rows: 10000

Row Type Setup Page

Adding a Row Type

1. Click the **Add** button to display the Add Row Type page.



Add Row Type Page

2. Click the down arrow to display a drop-down list of row types.
3. Select a row type from the list.
4. Click **OK** to apply your choice and exit the page, or click **Apply** to apply your choice and leave the page open.
5. While the page remains open, repeat Steps 2-4 to add more row types.



The drop-down list shows all valid row types for the current spreadsheet, including those which have already been included on the spreadsheet. It's possible to add them again, producing duplicate rows, which may generate incorrect results on the spreadsheet.

On the Row Type Setup grid, each newly added row type appears at the bottom of the column. The following table describes the row types:

Spreadsheet Row Types

Row Type	Description
Actual Supply	Scheduled receipts from either existing purchase orders, inventory transfer orders, or production orders. Actual Supply = Production Supply + Purchased Supply + Transfer Supply
ATP (Available-to-Promise)	Determines the projected available inventory that hasn't been committed to customer orders. Production Planning uses the following algorithm: ATP = Planned Receipts + Scheduled Receipts - Actual Demand - Carryover Demand.
Available Hours	Availability of a piece of equipment.
Available Units	Availability of a resource.

Row Type	Description
Carrying Cost	The cost of carrying inventory for all inventory items within the selection criteria.
Change in Inventory	The cost of the change in inventory from the beginning of the period to the end of the period for all inventory items within the selection criteria.
Cumulative ATP	Cumulative or running total of ATP.
Cumulative Profit	Show cumulative profit for each period through a given period.
Dependent Demand	All demand for an item that is for operations in the model—not from forecast or sales orders.
Down Hours	Downtime of a piece of equipment.
Excess Inventory	The maximum amount of inventory that is allowable. The planned inventory on hand should never exceed this number.
Forecast	Expected demand based on sales history, leading indicators, promotions, and so on.
Forecast Revenue	The revenue of all unconsumed forecast shipments within the selection criteria.
Independent Demand	All forecast and sales orders for an item.
Inter-Unit Orders	Demand from transfer orders placed on the source business unit.
New Planned Orders	The sum of Planned Production, Planned Purchases, and Planned Transfers.
Non Production Order Supply	Production supply not attached to a production order.
Non Transfer Order Supply	Transfer supply not attached to a transfer order.
Periods of Supply	A forward-looking calculation of that period's Planned On Hand balance, which determines how many days into the future that period's Planned On Hand balance will cover the Total Demand.
Planned Production	Suggested production orders created by Production Planning.
Planned Purchases	Suggested purchase orders created by Production Planning.

Row Type	Description
Planned Transfers	Suggested transfer orders created by Production Planning.
Production Cost	Cost of all production within the selection criteria.
Production Demand	The sum of demand this is from production.
Production Order Supply	Production supply attached to a production order.
Production Supply	Production supply for an item.
Production Supply Start	Production supply starting quantity, prior to yield and order modifiers.
Projected On Hand	Prior period ending on hand + (Actual Supply - Total Demand). This doesn't include new planned orders.
Purchase Cost	The cost of all planned orders and purchase orders within the selection criteria.
Purchase Supply	Purchase order and planned order supply for an item.
Required Hours	Required hours for a piece of equipment.
Required Units	Required units for a resource.
Safety Stock	The minimum quantity of stock planned to always be in inventory to protect against fluctuations in demand or supply.
Scheduled Production	The sum of demand from production with a status of Firm , Released , Dispatched , or In Process .
Scheduled Purchases	The sum of purchase supply with a status of Firm or Open .
Scheduled Transfers	The sum of transfer supply with a status of Firm , Released , or In Transit .
Starting On Hand	Available items at the start of a period.
Target Inventory Level	The average of (Safety Stock + Excess Inventory).
Total Cost	The sum of all the cost rows.
Total Orders	This sum of Customer Orders and Interunit Orders.

Row Type	Description
Total Profit	Difference between total revenue and total cost for a given period.
Total Revenue	Total of the actual and forecast revenue.
Transfer Cost	Total of the purchase, carrying, change in inventory, and production costs.
Transfer Order Supply	Transfer supply attached to a transfer order.
Transfer Supply	The scheduled receipts from other business units in the form of transfer orders.
Transfer Supply Out	Item quantity actually transferred during the time period.
Transfer Supply Start	Total item quantity to be transferred. This is represented as the quantity built. This quantity only exists in the time period in which the transfer begins.
Transfer Total Cost	Cost of all transfers within the selection criteria.
Unused Hours	Unused hours of a piece of equipment.
Unused Units	Unused units of a resource.
Utilization	Percentage utilization of a piece of equipment.

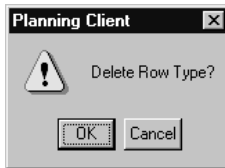


Each spreadsheet uses a valid subset of this list.

Deleting a Row Type

Deleting a row type isn't permanent; you can always restore it with the **Add** button. To delete a row type from the spreadsheet:

1. On the Row Type Setup grid, click the **row number** of the row type you want to delete, to select it. You can only manipulate a row type if it's selected this way.
2. Click the **Delete** button. A warning message box appears asking you whether you want to delete the row type.



Delete Row Type Warning Message Box

3. Click **OK** to delete the row type, or **Cancel** to abort the operation.

Setting the Number of Rows

By default, the program displays 100 rows of data on the Spreadsheet page. You may want to see more data, or speed up the query by requiring the program to display a smaller number of rows. To set the number of rows:

1. On the Row Type Setup page, type the number of rows you want to display in the **Maximum Rows** field.
2. Press the **Tab** key to complete the entry.
3. Click the **Apply** button to apply the new setting.
4. In the Spreadsheet page, click the **Query** button to generate the number of rows you specified.

Managing Sorting Behavior

Your PepperTools application uses all the columns in a Spreadsheet page grid (except the date and Past Due columns) for sorting the rows. The Row Type Setup and Key Setup pages provide controls for determining the sort order, specifying whether the sort is ascending or descending, and specifying if sorting is enabled. Every time you click the **Query** button on either of these pages, the data is sorted according to the latest settings. During sorting, rows are sorted first by key in the order specified, then by row type for each key combination.



In the Material Movement spreadsheet, the Spreadsheet page's column headings don't match the sort key names, but when the keys are in the default sort order, that order matches the left to right positions of the columns.

Changing the Sort Order

The positions of row types in the Row Type Setup grid, and the positions of keys in the Key Setup grid, combine to determine the sort order of rows on the Spreadsheet page (and the Row Type Total page, where applicable). The controls for positioning row types and keys are identical on the Row Type Setup and Key Setup pages.

To change the position of a row type or a key:

1. Select the Row Type Setup or Key Setup page, as appropriate.
2. To select the row type or key you want to move, click its **row number**. You can't manipulate a row type or key unless you select it this way.
3. Click one of the four positioning buttons:
 - **Up** moves the selected row type or key up one position on the grid.
 - **Down** moves the selected row type or key down one position on the grid.
 - **Top** moves the selected row type or key to the top of the grid.
 - **Bottom** moves the selected row type or key to the bottom of the grid.
4. After the row type or key is repositioned, the program deselects it. To move it again, repeat steps 1 and 2.

Sorting in Ascending or Descending Order

1. Select the Key Setup page.
2. For ascending order, click the **Ascending Sort** check box to *produce* a checkmark, OR
For descending order, click the **Ascending Sort** check box to *remove* the checkmark.
3. Click the **Apply** button to apply the new setting.
4. Click the **Query** button in the Spreadsheet page to generate the data with your specified sort order.

Enabling and Disabling Sorting

1. Select the Key Setup page.
2. To enable sorting, click the **Sorting Enabled** check box to *produce* a checkmark, OR
To disable sorting, click the **Sorting Enabled** check box to *remove* the checkmark.
3. Click the **Apply** button to apply the new setting.
4. Click the **Query** button in the Spreadsheet page to generate the data with your specified setting.

Setting Filter Criteria

You can restrict the data displayed on a spreadsheet by applying **filter** criteria to the sort keys. You can also apply advanced filtering for the **Inventory Item** and **Base Item** keys when those keys are present in the spreadsheet. The filter criteria limit the spreadsheet to rows that match the filter value you enter for a given key.



The program will consider any text string you enter for a filter to be matched if it occurs anywhere within the text of a key value. This means you can specify multiple values for a key if they all contain the same substring.



In the Material Movement spreadsheet, the Spreadsheet page's column headings don't match the sort key names, but when the keys are in the default sort order, that order matches the left to right positions of the columns.

Applying Basic Filtering

You can enter filter criteria in the Key Setup page grid or in the identical grid in the page header—each one mirrors the changes made in the other.

To set key filter criteria:

1. Select the Key Setup page if you want to use its grid.

Key Setup Page

2. In the **Filter** column of the Key Setup grid or the header grid, enter a value for each key you want to restrict. You can enter only one value per key, but it can be a substring that matches multiple keys. Press the **Tab** or **Enter** key after each entry.
3. Click the **Apply** button to apply the new setting.
4. Select the Spreadsheet page.

- Click the **Query** button to generate the data with your specified setting. The results displayed comply with the filter criteria that you entered and applied.



For any filter cell you leave blank, all possible values of that key are retrieved.

Applying Advanced Filtering

Some spreadsheets include pages that provide advanced filtering for the Inventory Item and Base Item keys, the Adv Inventory Item Filter and Adv Base Item Filter pages, respectively. These pages look identical except for the first data entry field. The program includes keys that match one or more of the filter criteria on each page. To apply advanced filtering:

- Select the page appropriate to the key you want to filter: Adv Inventory Item Filter or Adv Base Item Filter.

Spreadsheet	Row Type Setup	Key Setup	Summary Setup	Row Type Total	Adv Inventory Item Filter	Adv Base Item Filter
Inventory Item Filter: <input type="text"/>						
Description Filter: <input type="text"/>						
Planner Code Filter: <input type="text"/>						
Item Category 1 Filter: <input type="text"/>						
Item Category 2 Filter: <input type="text"/>						
MPS Filter: <input type="checkbox"/>						

Adv Inventory Item Filter Page



The **Inventory Item Filter** field at the top of the Adv Inventory Item Filter page is identical to the **Inventory_Item** row in the Key Setup grid, and will produce identical results.

Spreadsheet	Row Type Setup	Key Setup	Summary Setup	Row Type Total	Adv Inventory Item Filter	Adv Base Item Filter
Base Item Filter: <input type="text"/>						
Description Filter: <input type="text"/>						
Planner Code Filter: <input type="text"/>						
Item Category 1 Filter: <input type="text"/>						
Item Category 2 Filter: <input type="text"/>						
MPS Filter: <input type="checkbox"/>						

Adv Base Item Filter Page



The **Base Item Filter** field at the top of the Adv Base Item Filter page is identical to the **Base_Item** row in the Key Setup grid, and will produce identical results.

2. On each page, the data entry fields (except for the first one) enable you to specify values from the property sheets of the items. Enter a value for each field you want to restrict. You can enter only one value per field, but it can be a substring that matches multiple fields. Press the **Tab** or **Enter** key after each entry.
3. Click the **MPS Filter** check box to select or deselect it. If selected, only MPS items will be included in the spreadsheet for that key.
4. Click the **Apply** button to apply the new setting.
5. Select the Spreadsheet page.
6. Click the **Query** button to generate the data with your specified setting. The results displayed comply with the filter criteria that you entered and applied.



For any filter cell you leave blank, all possible values of that field are included.

Adding Summary Information

You can insert totals, averages, maxima and minima for each key in the spreadsheet.

To insert summary information:

1. Select the Summary Setup page.

Spreadsheet Row Type Setup Key Setup Summary Setup Row Type Total Adv Inventory Item Filter					
	Key	Total	Average	MAX	MIN
1	Site	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	Inventory_Part	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Summary Setup Page

2. Click the appropriate X boxes for your desired results:
 - Click the X box in the **Total** column to produce an X for each key for which you want to display a total value.
 - Click the X box in the **Average** column to produce an X for each key for which you want to display an average value.

- Click the X box in the **MAX** column to produce an X for each key for which you want to display a maximum value.
 - Click the X box in the **MIN** column to produce an X for each key for which you want to display a minimum value.
3. Click the **Apply** button to apply the new setting.
 4. Select the Spreadsheet page.
 5. Click the **Query** button to generate the data with the chosen summary information.

Spreadsheet Row Type Setup Key Setup Summary Setup Row Type Total Adv Inventory Item Filter										
Query		Print Report								
	Unit	Inventory Item	Row Type	Past Due	3/20/98	3/21/98	3/22/98	3/23/98	3/24/98	3/25/98
1	M04	20001	Net Forecast	0.00	0.00	0.00	0.00	0.00	0.00	0
2	M04	20001	Customer Orders	0.00	0.00	0.00	0.00	0.00	0.00	0
3	M04	20001	Dependent Demand	0.00	0.00	0.00	0.00	0.00	0.00	0
4	M04	20001	Total Demand	0.00	0.00	0.00	0.00	0.00	0.00	0
5	M04	20001	Total Supply	0.00	0.00	0.00	0.00	0.00	0.00	0
6	M04	20001	Planned On Hand	0.00	0.00	0.00	0.00	0.00	0.00	0
7		Total	Net Forecast	0.00	0.00	0.00	0.00	0.00	0.00	0
8		Total	Customer Orders	0.00	0.00	0.00	0.00	0.00	0.00	0
9		Total	Dependent Demand	0.00	0.00	0.00	0.00	0.00	0.00	0
10		Total	Total Demand	0.00	0.00	0.00	0.00	0.00	0.00	0

Spreadsheet with Summary Information



The total or subtotal for each key applies to all row types at once.

CHAPTER 8

Using the Class Editor

Use the Class Editor page to view sub-class information and edit default field values. You can also create new subclasses using this page.



Changing fields and actions can be dangerous if you are not familiar with Object-Oriented Programming.

Understanding the Class Editor Component

The Class Editor component consists of the following three areas:

- Class Editor Header (the area above the Class Slot pages).
- Class Slot pages.
- The standard command buttons at the bottom of the page.

Following is a Class Editor component.

The screenshot shows the 'Class Editor' window for the 'Production_Task' class. The 'Class' and 'Class Display Name' fields are both set to 'Production_Task'. Below these are buttons for 'Select...', 'Add...', 'Query', and 'Sort'. A tabbed interface shows 'Integers' selected, with other tabs for 'Floats', 'Strings', 'Dates', 'Times', 'Instances', and 'Actions'. A table lists various slots and their values:

Slot#	Slot	Value
27	isSplittable_task	0
28	is_start_milestone_task	0
29	is_transfer_parent_task	0
30	is_transfer_task	0
31	is_transport_task	0
32	is_unsplittable_leaf_task	1
33	is_unsplittable_parent_task	0
34	preemptable	0
35	priority	1
36	routing_step	0

At the bottom are buttons for 'OK', 'Cancel', 'Apply', and 'Help'.

Class Editor Component

Understanding the Class Editor Header

The following table describes each field found on the header for the Class Editor component.

Field	Description
Class	Name of the class.
Class Display Name	Display name of the class. The name that is shown in the Planning client pages for this class.

The following command buttons appear on the header for the Class Editor component.

Command	Description
Select	Find classes. Displays the Find: Class page.
Add	Add subclasses. Displays the Add SubClass page.
Query	Queries the data based on the specified class and displays the results in the Class Slot page.

Understanding the Class Slot Pages

The Class Editor component has the following Class Slot pages:

- Integers
- Floats
- Strings
- Dates
- Times
- Instances
- Actions

Use the Class Slot pages to view and edit slot fields and their default values.

The following command button appears on each Class Slot page.

Command	Description
Sort	Sort columns with an asterisk (*) after the column name in ascending order.

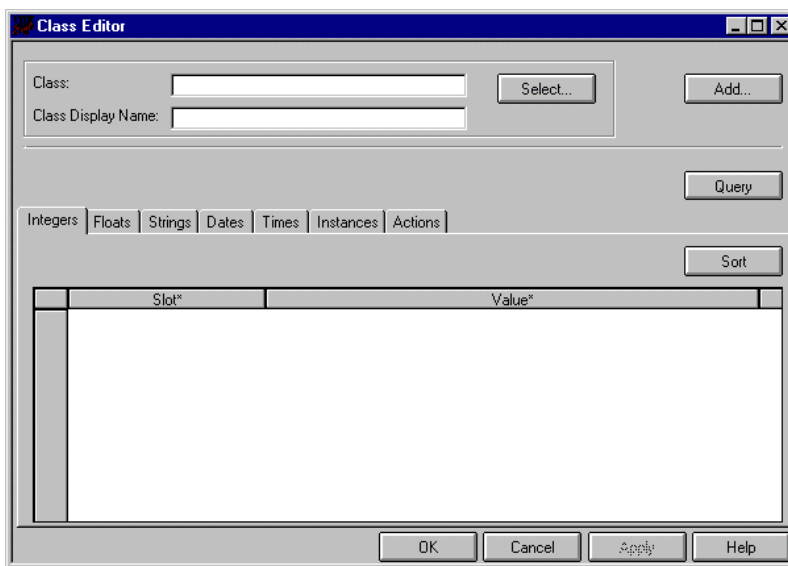
The following table describes each column found on each Class Slot page.

Column	Description
Slot	Name of the class field.
Value	Default value of the class field.

Viewing and Editing a Class

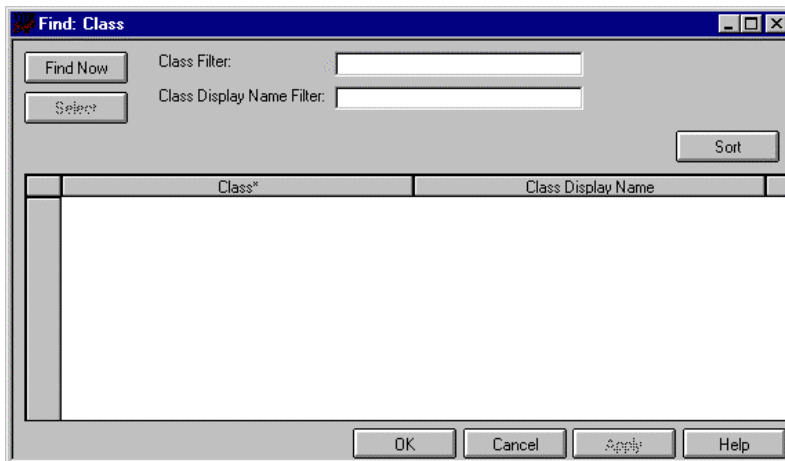
To view and edit a class, complete the following steps:

1. Select **Edit, Edit Class** to display the Class Editor page.



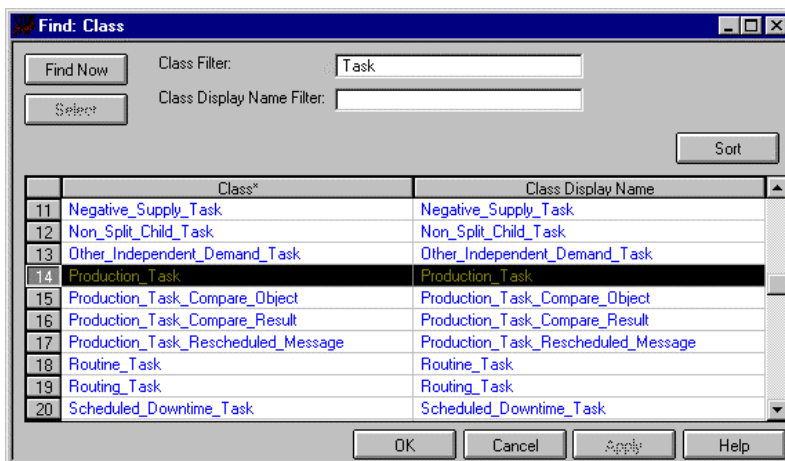
Class Editor Page

2. Click **Select**. The Find: Class page appears. Click **Find Now** to list all classes, or first enter a text string into Class Filter or Class Display Name to list classes that have that text string in their name or display name, and then click **Find Now**. Note: the text string is case sensitive.



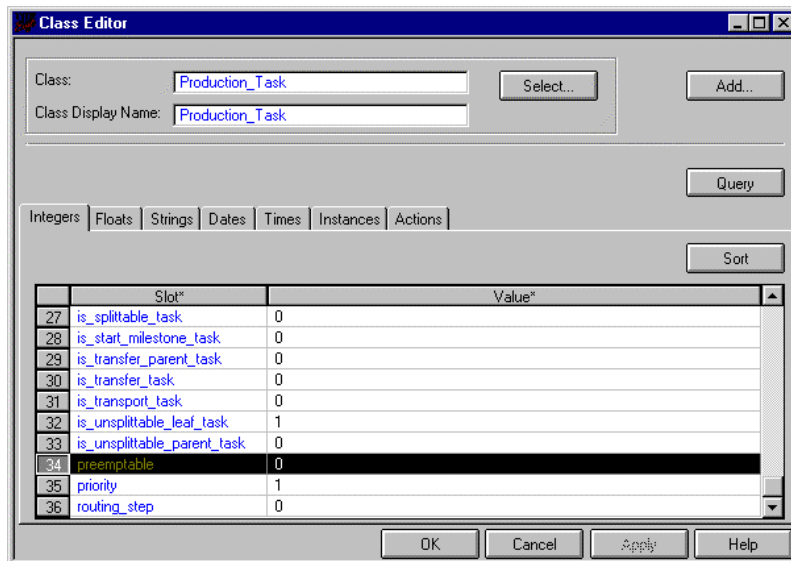
Find: Class Page

- Click the number to the left of the class name that you want to select, click **Select**, and then close the Find: Class page. The figure below shows the Production_Task class being selected, using Task as a filter for the class names.



Find: Class Page for Part

- In the Class Editor page, click **Query** to populate the Class Editor page with the selected class. The page displays slots and their default values for that class. Click on the tabs to display different slot types. The following page shows Integer slots for the Production_Task class.

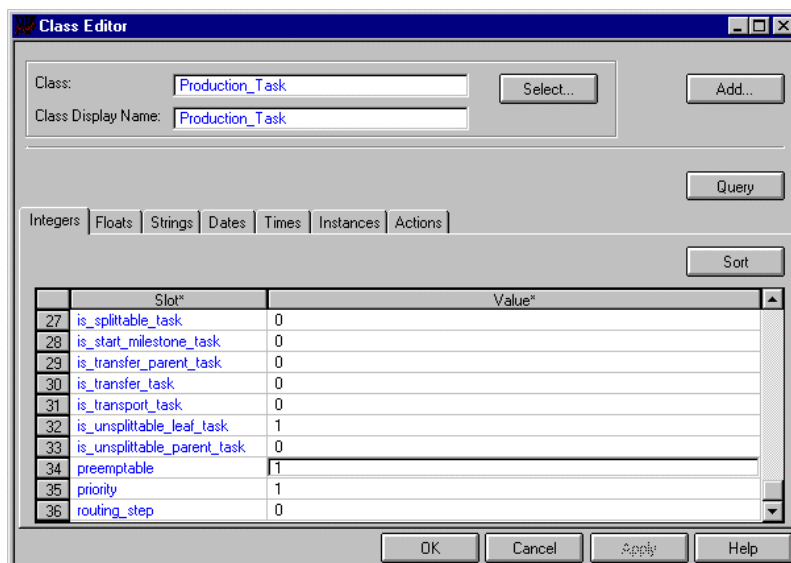


The Class Editor window shows the 'Production_Task' class. The 'Integers' tab is selected. The table below lists the integer slots and their current values.

Slot*	Value*
27 isSplittable_task	0
28 is_start_milestone_task	0
29 is_transfer_parent_task	0
30 is_transfer_task	0
31 is_transport_task	0
32 is_unsplittable_leaf_task	1
33 is_unsplittable_parent_task	0
34 preemptable	0
35 priority	1
36 routing_step	0

Class Editor: Integer Slots

- If desired, change the default values of the slots. Choose the slot name in the Slot* column whose default value you want to change, click on the Value* column to the right of the slot name, and type the desired default value. In the case of the Production_Task class, you could change the value of the integer slot preemptable from 0 to 1.



The Class Editor window shows the 'Production_Task' class. The 'Integers' tab is selected. The table below lists the integer slots and their current values. The value for the 'preemptable' slot (row 34) has been changed from 0 to 1.

Slot*	Value*
27 isSplittable_task	0
28 is_start_milestone_task	0
29 is_transfer_parent_task	0
30 is_transfer_task	0
31 is_transport_task	0
32 is_unsplittable_leaf_task	1
33 is_unsplittable_parent_task	0
34 preemptable	1
35 priority	1
36 routing_step	0

Class Editor: Integer Slot Value Changed

Understanding the Add SubClass Page

You can add a subclass to existing classes with the Add SubClass page.

Add SubClass Page

The following table describes each field found on the Add SubClass page.

Field	Description
SubClass Name	Name of the subclass.
Parent Class	Name of a parent class for this subclass.
Parent Class Display Name	Display name of a parent class. The name that is shown in the Planning client pages for this class.

The following command buttons appear on the Add SubClass page.

Command	Description
Select	Find classes. Displays the Find: Class page.
Add to List	Adds a class to the list of parent classes for this subclass.
Delete from List	Deletes a class from the list of parent classes for this subclass.

The following table describes each column found on the Add SubClass page.

Column	Description
Parent Classes	List of the parent classes for this subclass. The new subclass will inherit all the attributes from its parent classes.

Parent Classes Display Names	List of the display names of the parent classes. The name that is shown in the Planning client pages for this class.
------------------------------	--

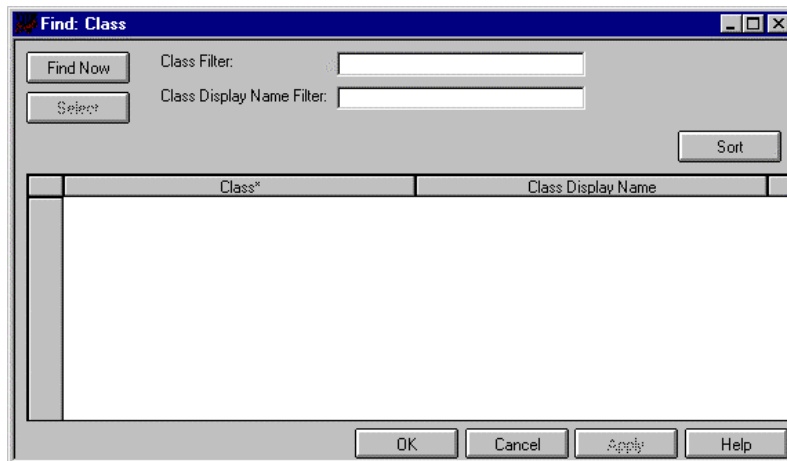
Adding a Subclass

To add a subclass, perform the following steps:

1. On the Class Editor page, click **Add**. The Add SubClass page appears.

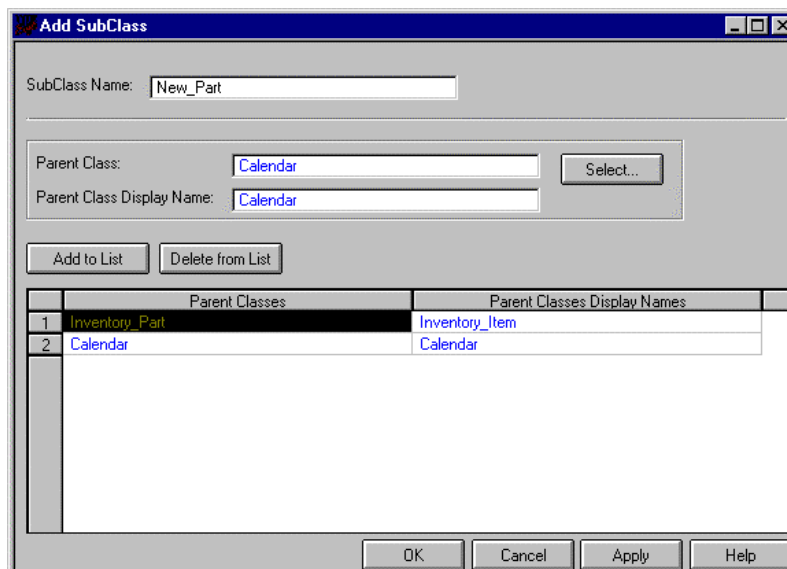
Add SubClass Page

2. In the Add SubClass page, click **Select**. The Find: Class page appears. Click **Find Now** to list all classes, or first enter a text string into Class Filter or Class Display Name to list classes that have that text string in their name or display name, and then click **Find Now**. Note: the text string is case sensitive.



Find: Class Page

3. Click the number to the left of the class name that you want to select, and either click **OK**, or click **Select** and close the Find: Class page.
4. In the Add SubClass page, the name of the parent class that you are adding to the subclass now appears in the Parent Class field. Click **Add to List** to add this class to the list of parent classes for this subclass.
5. Repeat steps 2 through 4 to add as many parent classes to this subclass as you wish. To remove a parent class from the list, click **Delete from List** in the Add SubClass page. The following page shows two parent classes—Inventory_Part and Calendar—having been added to the subclass that you are creating, named New_Part.



Add SubClass With Two Parent Classes

6. In the Add SubClass page, type the desired SubClass name in the SubClass Name field, and then click **OK**. The new subclass is created. The figure above shows the name New_Part for the subclass name, with the parent classes Inventory_Part and Calendar. If you were to create

and then find this subclass, you would see that it contains all the slots for its parent classes. For example, the New_Part class created in this example has all the instance slots of the Inventory_Part class, plus the added instance slot of calendar_spec from the Calendar class.

The screenshot shows the 'Class Editor' window. At the top, the 'Class' field is set to 'New_Part' and the 'Class Display Name' is also 'New_Part'. Below these are buttons for 'Select...', 'Add...', 'Query', and 'Sort'. A tabbed interface shows 'Instances' selected. The main area is a table with two columns: 'Slot*' and 'Value*'. The table lists seven slots, all with 'Null_Instance' as their value.

	Slot*	Value*
1	calendar_spec	Null_Instance
2	default_production_area	Null_Instance
3	planning_container	Null_Instance
4	relevant_status	_Relevant_Status_RELEVANT_REPAIR
5	site	Null_Instance
6	sourcing_logic_template	Null_Instance
7	stock_container	Null_Instance

At the bottom of the window are buttons for 'OK', 'Cancel', 'Apply', and 'Help'.

A Subclass: Instance Slots from all its parent classes

CHAPTER 9

Editing and Creating Pages with the Form Painter

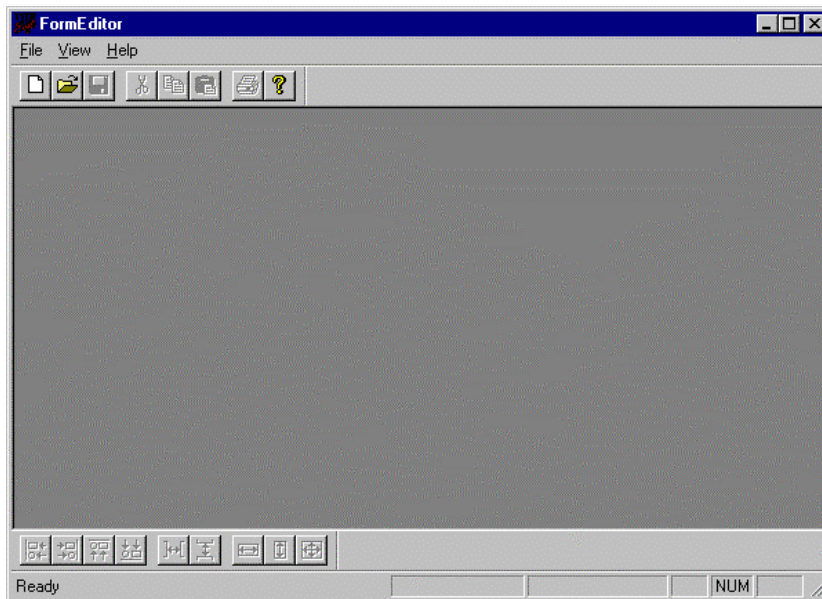
The Form Painter helps you to create pages that you can use in the Planning client. It also lets you modify existing pages. Formerly, you had to create these pages using the Resource Language; you would create a file containing the Resource Language code needed to create a form. Using the Form Painter allows you to use a GUI to create pages and their associated files containing the Resource Language code.

Installing and Starting the Form Painter

To install the Form Painter, perform the following steps:

1. Insert the CD-ROM from which you installed the client software.
2. Open the Form Painter folder.
3. Double-click the **Setup.exe** icon.
4. Follow the directions that appear in the Install Shield program.

To start the Form Painter if you installed it in the default location, select Programs > Form Painter > Form Painter from the toolbar. If you installed Form Painter in another location, select that location instead. The following page appears.



Form Painter Page

Creating a Page

Use the Form Painter to create the following:

- Forms, which create form pages. Forms contain controls that allow users to view and modify data. The controls are usually populated, or get the data that they display, from either an instance of a class or a slot on an instance.
- ClassFiles, which create property sheet pages. Property sheets are the same as a form except that they are associated with a class, and can thus have a set menu and an instance menu associated with them.



For more information about the resource language code of a property sheet page, refer to Property Sheet Page: BEGINPROPSHEET, ENDPROPSHEET. **For an example** of a set menu and instance menu, refer to Understanding the Unit Page Popup Menu Code.

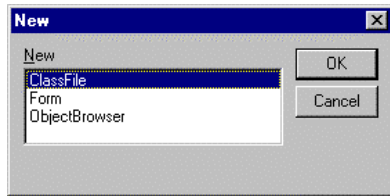
- ObjectBrowser, which create a data browser page.



For more information about the resource language code of a data browser, refer to Writing a Data Browser.

To create a Form page, ClassFile page, or ObjectBrowser page, perform the following steps:

1. From the File menu, select File > New. A page that lists the types of pages that you can create appears.



Page showing the page types

2. Click on one of the choices.

Clicking **Form** opens a blank form page.



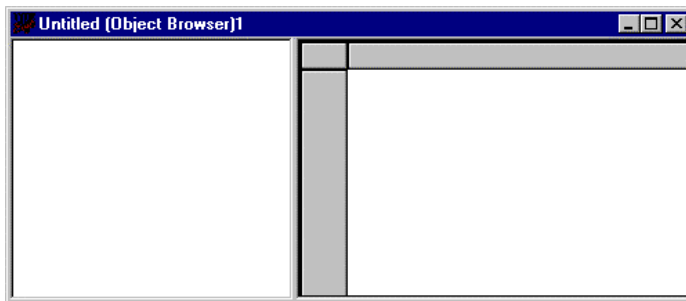
A New Form Page.

Clicking **ClassFile** opens a blank property sheet page.



A New Property Sheet Page.

Clicking **ObjectBrowser** opens a blank data browser page that resembles the following page.



A New Data Browser.



Note: Once you have created a form or property sheet page, it will, upon being closed and reopened, resize to fit the controls that have been inserted into it.

Opening a Page

You can open a previously existing page with the Form Painter. You can then edit its properties and controls.

To open a page, perform the following steps:

1. From the File menu, select File > Open. A window opens that shows the current directory and its filenames.
2. Navigate to the directory where the file for the page that you want to open is stored.
3. Double-click on the filename. The page opens in the Form Painter.



Note: Files that do not contain property sheet or form information do not open in the Form Painter. A few of the Red Pepper files in the Forms directory do not open in the Forms Painter for this reason. For example, BOR_Entry.000 and BOM_Entry.000 do not open because they contain only class and menu information.

Saving a Page

Once you have created a page or you have opened and edited an existing page, you can save the page by selecting File > Save or File > Save As. The page will be saved as a resource file with an extension of .000. This file will contain the Resource Language code that is created when you create pages, edit them, and add controls and properties to them using the Form Painter.

For example, if you create a form page and then immediately save it without adding any controls, you will create a file containing the following resource language code:

[FORM]

```
BEGINFORM
```

```
  LABEL  " " ;
```

```
ENDFORM;
```

Editing Page Properties

Once you create a form page or property sheet page, you should add properties to it. You can also open an existing page and edit its properties. The properties include the page's label, the actions taken when you click the page's buttons, and the topic that populates this page.

Editing the Topic, Default Buttons, and Label on a Page

Once you have created a new page or opened an existing page, double-clicking inside the page will bring up a dialog box containing several tabs that, when you click on them, allow you to edit the page's topic, default buttons, and label:



Note: Double-clicking inside a data browser page will not bring up a dialog box. To add functionality to a data browser, you must edit the object browser file that you created and add the code to it for the functionality the you want.

- **General:** Enter the label you want for this page.
- **Topic Populate:** Enter the TOPICPOPULATE that you want for this page. For example, entering the text TRANSACTION "ui_get_mfg_scorecard()" will result in the Resource Language TOPICPOPULATE code TOPICPOPULATE TRANSACTION "ui_get_mfg_scorecard()" being added to the code for this page, and will also result in the transaction ui_get_mfg_scorecard() being called to get the mfg scorecard instance to populate this page. This is done in the Score Card page. The Topic Populate tab has a default of PARAMETER ":ParentsTopic".



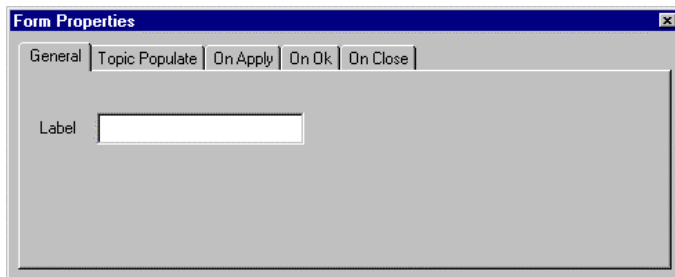
For the Score Card Page code, refer to Understanding the Score Card Page Code.

- **On Apply:** Enter the command that you want performed when you click **Apply** on this page. For example, entering the text TRANSACTION "ui_time_change_early_fence(:data %[].Value)" will result in the Resource Language code ONAPPLY {TRANSACTION "ui_time_change_early_fence(:data %[].Value)"}; being added to this page, and will also result in the transaction ui_time_change_early_fence(:data %[].Value) being called when **Apply** for this page is clicked. This is done in the Scorecard page.



For the Score Card Page code, refer to Understanding the Score Card Page Code.

- On OK: Same as for On Apply, except that you enter the command that you want performed whenever you click **OK** for this page.
- On Close: Same as for On Apply, except that you enter the command that you want performed whenever you close this page.



Form Page Properties Dialog Box.

A property sheet (ClassFile) will show the following dialog box. You can click the same tabs as for the form properties dialog box, plus the following tabs. These tabs allow you to set the set menu and instance menu that are used with a property sheet.

- Class: Enter the LABEL that you would like for the class, and the CLASS name for the class that you want associated with this page. For example, the Unit page, from the Site.000 file, has the LABEL “Unit” and the class “Site”.
- Set Menu: Enter the Resource Language code that you need for the set menu. A set menu is a menu that you want displayed when you right-click a reference to sets of instances of this class. For example, in the Planning Data Browser page, if you right-click **Units**, the Unit set menu is displayed. Where you would double-click to list a set of instances of a class, you can right-click to display the set menu.



For the Unit set menu resource language code, refer to Understanding the Unit Set Menu.

- Instance Menu: Enter the Resource Language code that you need for the instance menu. An instance menu is a menu that you want displayed when you right-click on a name for a single instance of this class. For example, in the Planning Data Browser page, if you right-click on a unit name, the Unit instance menu is displayed. Where you would double-click on an instance to show the page for an instance of that class, you can right-click to display the instance menu.



For the Unit instance menu resource language code, refer to Understanding the Unit Instance Menu.

```

[INSTANCEMENU]

BEGINMENU

    LABEL "Unit";

MENUITEM

    LABEL "Unit Scorecard"

    ALWAYSEDIT

    ONPICK {LOADPROPSHEET ("Site_Scorecard", :Passed_Site_Or_Site_Node
%[] .PopupMenuTopic)};

MENUITEM

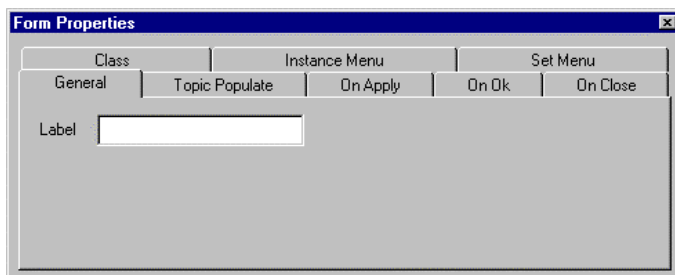
    LABEL "Properties"

    ALWAYSEDIT

    ONPICK{LOADPROPSHEET ("Site", :ParentsTopic %[] .PopupMenuTopic)};

ENDMENU;

```



Property Sheet (ClassFile) Properties Dialog Box.

Making the Page Editable or Non-editable

To make a page editable or non-editable to the user, perform the following steps:

1. Right-click inside the page. A popup menu appears with four choices: **Insert Component**, **Create New Tab Page**, **Delete Current Tab Page**, and **Properties**.
2. Select **Properties**, **NoEdit** from the popup menu to add the NOEDIT property to your page, preventing the user from changing the data shown in the page. Select **Properties**, **AlwaysEdit** from the popup menu to add the ALWAYSEDIT property to your page, allowing the user to change the data shown in the page. Selecting one of these properties will overwrite the other property.

Creating and Editing Components

Once you have a Form or Property Sheet page, you can change it into a component by adding tab pages to it.

To change a page into a component, perform the following steps.

1. Open the file for the page that you want to edit.

Either open a new page by following the steps in Creating a Page with the Form Painter, or open an existing page by selecting File >Open and navigating to the Resource Language file for the page that you want to modify. The Resource Language files are usually stored in C:\Rps75\pra (or era or sra)\Forms.



For more information about resource language files, refer to Writing a Resource Language File.

2. Right-click inside the page. A popup menu appears with four choices: Insert Component, Create New Tab Page, Delete Current Tab Page, and Properties.
3. Select **Create New Tab Page** from the popup menu. This changes your page into a component by adding a tab page to your page. This tab page acts as a page. To edit the label of the page/tab page, double-click on the page/tab page to bring up a properties dialog box, and enter the desired label in the Label field. To add controls to this page/tab page, right-click inside the tab page, and the same pop-up menu appears inside the page/tab page as for a page; use that menu to edit this page/tab page the same way you would a page.

Select Create New Tab Page for as many pages/tab pages that you want to add to your component. Selecting it twice results in a component containing two pages. The following tab pages are added to your page, and the following code is added to your page file.

```
BEGINPAGE
```

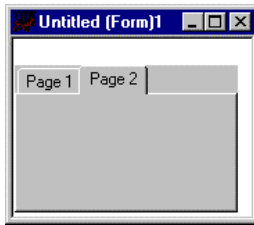
```
    LABEL "Page 1";
```

```
ENDPAGE;
```

```
BEGINPAGE
```

```
    LABEL "Page 2";
```

```
ENDPAGE;
```



A Component.

4. If you want to delete a page from a component, click on the tab for the page/tab page that you want to remove, right-click in the page/tab page that you want to delete, and select **Delete Current Tab Page** from the popup menu to remove that page/tab page from your component.



Right-clicking in a component outside of its pages/tab pages and selecting **Delete Current Tab Page** will not delete the current page/tab page; you must right-click directly on the page/tab page that you want to delete.

Adding Controls to a Page

Once you have a Form or Property Sheet page, you can add controls to it, such as Pushbuttons, Control Sliders, and Bitmaps. You can also make the pages and controls editable or non-editable.

To add a control, perform the following steps.

1. Open the file for the page to which you want to add controls.

Either open a new page by following the steps in Creating a Page with the Form Painter, or open an existing page by selecting File >Open and navigating to the Resource Language file for the page that you want to modify. The Resource Language files are usually stored in C:\Rps75\pra (or era or sra)\Forms.

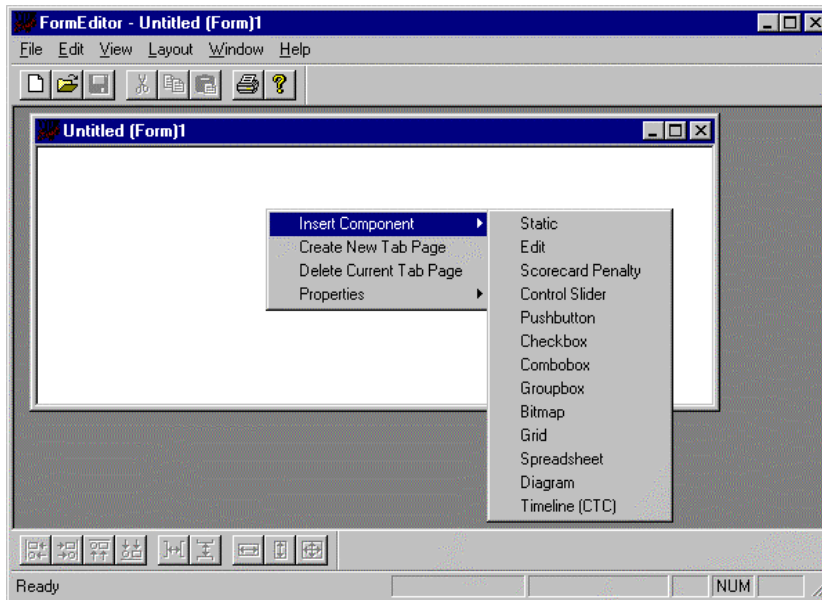


For more information about resource language files, refer to Writing a Resource Language File.

2. Right-click inside the page. A popup menu appears with four choices: **Insert Component**, **Create New Tab Page**, **Delete Current Tab Page**, and **Properties**.
3. Select **Insert Component** to list several controls that you can select to add that control to the page, and click on the control that you want to add to your page. You can add the following controls: Static, Edit, Scorecard Penalty, Control Slider, Pushbutton, Checkbox, Combobox, Groupbox, Bitmap, Grid, Spreadsheet, Diagram, and Timeline (CTC).

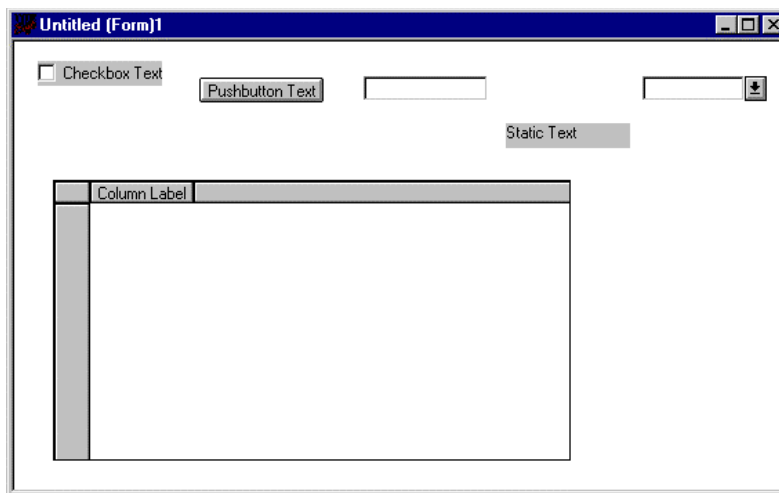


For more information about editing controls, refer to Editing Controls.



Pop-up Menu for Adding Controls

Here is a page with several controls added (Checkbox, Pushbutton, Edit, Static, Combobox, and Grid).



Page with controls added

Deleting Controls

To delete a control, perform the following steps.

1. Right-click the control. A popup menu appears with two choices: **Delete** and **Properties**.

2. Click **Delete**. The control is deleted.

Editing Controls

You can make a control editable or non-editable, and you can edit the control's properties.

Making a Control Editable or Non-editable

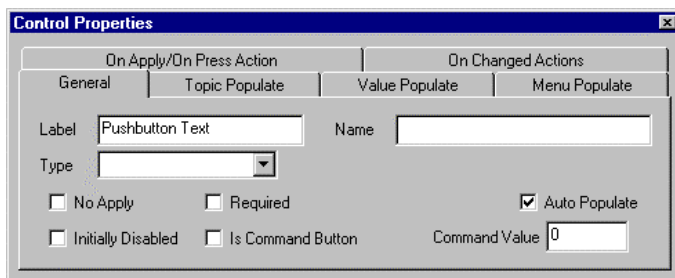
To make a control editable or non-editable, perform the following steps.

1. Right-click the control. A popup menu appears with two choices: **Delete** and **Properties**.
2. Click **Properties** to bring up the Properties drop-down menus. Select **Properties, NoEdit** to add the NOEDIT property to your control, preventing the user from changing the data shown in the control. Select **Properties, AlwaysEdit** to add the ALWAYSEDIT property to your page, allowing the user to change the data shown in the control. Selecting one of these properties will overwrite the other property.
3. The popup menu for the Edit control has one more choice: **Justification**. Select **Justification, (Left or Center or Right)** to choose how data should be justified within the Edit control.

Editing Properties for a Control

To edit a control's properties, double-click on the control. You will get a dialog box containing several tabs that allow you to edit the control's properties.

A control will show the following property dialog box.



Control Property Dialog Box (General Tab).

You can click on the following tabs. They will bring up General, Topic Populate, Value Populate, Menu Populate, OnChangedActions, and OnApply/OnPress Action.

- General tab: You can edit the following control properties:



For more information about the resource language code for these properties, refer to Writing Statement Properties.

- **Label:** Enter the label that you want for this control. This will add the LABEL property to this control.
- **Name:** Enter the name that you want for this control. This will add the NAME property to this control.
- **Type:** Enter the data type that you want for this control. This will add the TYPE property to this control. Depending upon the control, you can display a class with RPS_CLASS, a date and time with RPS_DATE, an enumeration with RPS_ENUM, a float with RPS_FLOAT, an instance with RPS_INSTANCE, and integer with RPS_INT, a character string with RPS_STRING, or a time with RPS_TIME.
- **No Apply:** Check this checkbox if you want the NOAPPLY property added to this control (if the user makes changes to the value shown in this control, the Apply button on the page remains disabled).
- **No Edit:** Check this checkbox if you want the NOEDIT property added to this control (prevents the user from changing the data shown in the control).
- **Required:** Check this checkbox if you want the REQUIRED property added to this control (prevents the user from leaving the page without entering data in this control; a Required error message box appears if the user tries to leave the page without filling out this control).
- **AutoPopulate:** Do not check this checkbox if you want the NOAUTOPOPULATE property to be added to this control (when this control is first displayed, this control will not be automatically populated, leaving the control blank).
- **Initially Disabled:** Check this checkbox if you want the DISABLED property to be added to this control (the initial state of this control is disabled).
- **Is Command Button:** Used with the PUSHBUTTON, or button, control. Adds the COMMAND command to the control. Check this checkbox if you want this to send a Windows command message when the pushbutton is clicked. (This is not used much any longer, since the COMMAND functionality is now available in the ONPRESS event.)
- **Command Value:** Used with the PUSHBUTTON, or button, control. Allows you to set the Windows command message number to send when the button is clicked (you must have the Is Command Button checkbox clicked). (This is not used much any longer, since the COMMAND functionality is now available in the ONPRESS event.)
- **Topic Populate tab:** Enter the TOPICPOPULATE that you want for this control. For example, entering the text TRANSACTION "get_oid main_environment;" will result in the Resource Language TOPICPOPULATE code TOPICPOPULATE TRANSACTION "get_oid main_environment;" being added to the code for this control, and will also result in the transaction get_oid main_environment being called to get the topic of the main environment instance, which is usually the topic of the page.



For more information about the resource language code for topic populate, refer to Writing Statement Properties. For more information about populating controls, refer to Populating Pages and Controls.

- **OnChangedActions** tab: Enter the command that you want performed when the user changes the value in this control (ONCHANGE).
- **OnApply/OnPress Action** tab: Enter the command that you want performed when you press a button (ONPRESS) or when the value shown in the control is changed (ONAPPLY).



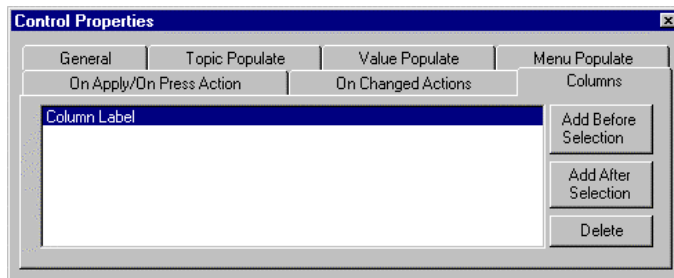
For more information about the ONCHANGE, ONPRESS, and ONAPPLY resource language code, refer to Writing Statement Events.

Adding Columns to Grid and Spreadsheet Controls

Grid and spreadsheet controls have one added tab in their dialog box: Columns.

To add a column to a grid or a spreadsheet, perform the following steps:

1. Double-click inside the Grid or Spreadsheet page to bring up its dialog box. Click the Columns tab.



Grid and Spreadsheet Dialog Box: Columns

2. Click on the column name to highlight it.
3. Click **Add Before Selection** to add the new column to the left of the existing column, or click **Add After Selection** to add the new column to the right of the existing column. Then double-click the column name where you want to add the new column to bring up the Column dialog box, and enter a new column name for the new column.

Editing Columns

The Column Dialog Box appears when you double-click on a column name in the Grid or Spreadsheet control properties dialog box, Columns tab.

Grid and Spreadsheet Dialog Box: Columns

The Columns dialog box allows you to edit a column. It allows you to edit the following column properties:



For more information about the resource language code for columns and column properties, refer to Columns in a Grid or Spreadsheet: COLUMN.

- General tab: You can edit the following column properties:
 - Label: Enter the label that you want for this column. This will edit the LABEL property for this column (or will add it for a new column).
 - Type: Enter the data type that you want to display in this column. This will edit the TYPE property for this column (or will add it for a new column). You can display a date and time with RPS_DATE, an enumeration with RPS_ENUM, a float with RPS_FLOAT, an instance with RPS_INSTANCE, an integer with RPS_INT, a character string with RPS_STRING, or a time with RPS_TIME.
 - Widget Type: You can use "CHECKBOX", "EDIT", or "COMBOBOX". For example, "CHECKBOX" would place a checkbox inside this column, and you can then populate and use that checkbox as you would a CHECKBOX control statement.
 - Justification: LEFT or CENTER or RIGHT. LEFT left-justifies the data in this column; CENTER centers it; RIGHT right-justifies it.
 - Sort: Enter a slot name. This column will then be sorted by the value of the slot on the instance of the row. For example, SORT "site.display_name" will sort the column on the site display name slot; you need to have populated the SPREADSHEET or GRID so that you can access site.display_name.



For an example of SORT being used in the resource language, refer to Understanding the Purchase Order Page Code.

- Format: Used when the TYPE is RPS_FLOAT. format is the format used to display data in this field. # displays a digit except for leading zeros, 0 displays a digit and trailing zeros, and . displays the decimal point. For example, "###,##0.00" means display a number of up to eight digits with two decimal places.

- Width: The number of units of width for the column. To calculate the units for minimum column width, multiply the number of characters you wish to display by 4.5.
- No Edit: Adds the NOEDIT property to this column, preventing the user from changing the data shown in the column.
- Value Populate tab: Enter the VALUEPOPULATE that you want for this column. For example, entering the text PATH "site" will result in the Resource Language VALUEPOPULATE code VALUEPOPULATE PATH "site" being added to the code for this column, and will result in the column being populated with an instance of the site class. This example requires that the GRID or SPREADSHEET be TOPICPOPULATED so that "site" is accessible to VALUEPOPULATE.



For an example of this being used in the resource language, refer to Understanding the Purchase Order Page Code.

- Menu Populate tab: Used only if Widget Type is COMBOBOX. Enter the MENUPOPULATE that you want for this COMBOBOX. The MENUPOPULATE property gives a COMBOBOX control statement an osset of values to display; an osset of enums, an osset of instances, or an osset of classes.
- On Apply tab: Enter the command that you wish to perform when the user presses the Apply button and the value in the COLUMN is changed. For example, you can have a transaction set the slot on the server that populated this COLUMN to the value set by the user.

Editing Timeline (CTC) Controls

Timeline (CTC) controls have one added tab in their control properties dialog box: Chart Components.



For more information on the resource language code for Timeline (CTC), refer to Coordinated Time Control Widget: BEGINCTC, ENDCTC.

- Scale: Allows you to set the time scale that will be used in this timeline (CTC). You can set it to HOUR, SHIFT, DAY, WEEK, MONTH, QUARTER, or YEAR. This sets the INITIAL_CALENDAR_SCALE property in the code.
- Zoom: Allows you to set the zoom scale that will be used in this timeline. You can set it to HOUR, SHIFT, DAY, WEEK, MONTH, QUARTER, or YEAR. This sets the INITIAL_CALENDAR_SCALE property in the code.
- In the large edit box, enter the code that you want for this timeline (CTC). You will add such statements to your timeline (CTC) such as Histograms or Gantt charts.

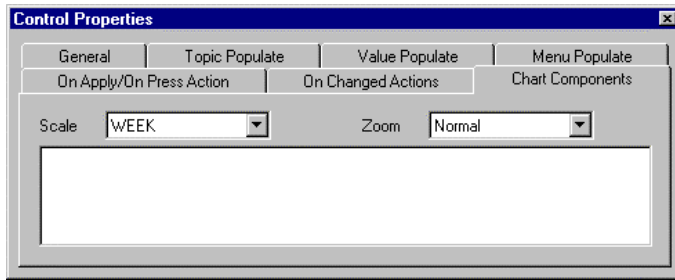


Diagram (CTC) Dialog Box: Chart Components

Moving, Aligning and Resizing Controls

To move a single control, click on it and drag it to the desired location. To resize a control, click on it to select it, then click on one of the handles that appear around the control, and then drag the control to the desired size.

You can align controls using either the Layout menu or the buttons at the bottom of the page; these both perform the same functions. The following graphic shows the Align and Layout buttons. From right to left, they are **Align Left**, **Align Right**, **Align Top**, **Align Bottom**, **Space Across**, **Space Down**, **Make Same Width**, **Make Same Height**, and **Make Same Size**.



Align and Layout Buttons

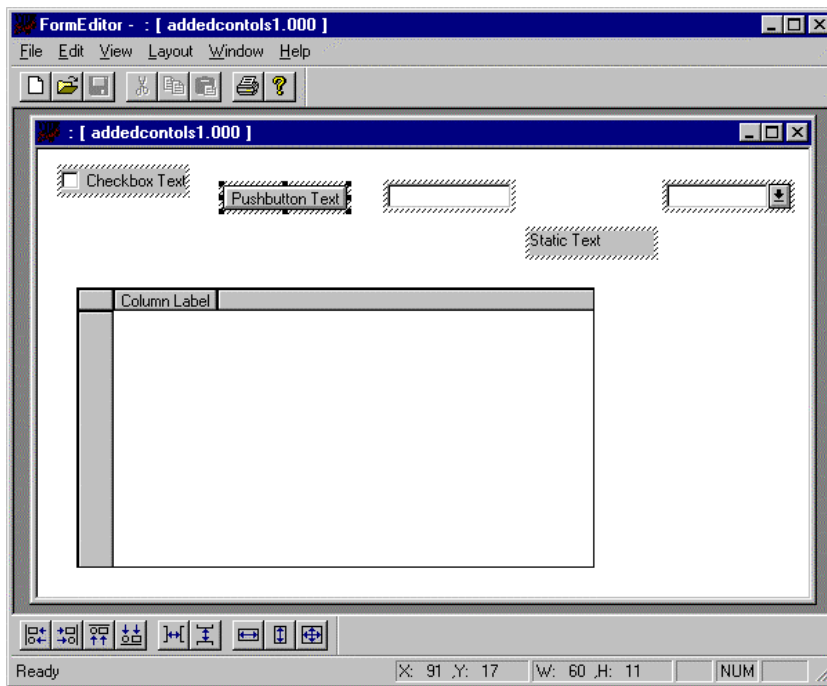
To align controls, first select the controls that you want to align by dragging the mouse over them. Then perform one or more of the following steps:

- **Align Left:** Click **Align Left** or select **Layout, Align Left**. The controls will move to have their left edges aligned.
- **Align Right:** Click **Align Right** button or select **Layout, Align Right**. The controls will move to have their right edges aligned.
- **Align Top:** Click **Align Top** or select **Layout, Align Top**. The controls will move to have their top edges aligned.
- **Align Bottom:** Click **Align Bottom** or select **Layout, Align Bottom**. The controls will move to have their bottom edges aligned.
- **Space Across:** Click **Space Across** or **Layout, Space Across**. The controls will move to be evenly spaced horizontally.
- **Space Down:** Click **Space Down** or select **Layout, Space Down**. The controls will move to be evenly spaced vertically.

To resize more than one control at once, first select the controls that you want to resize by dragging the mouse over them. Then perform one or more of the following steps:

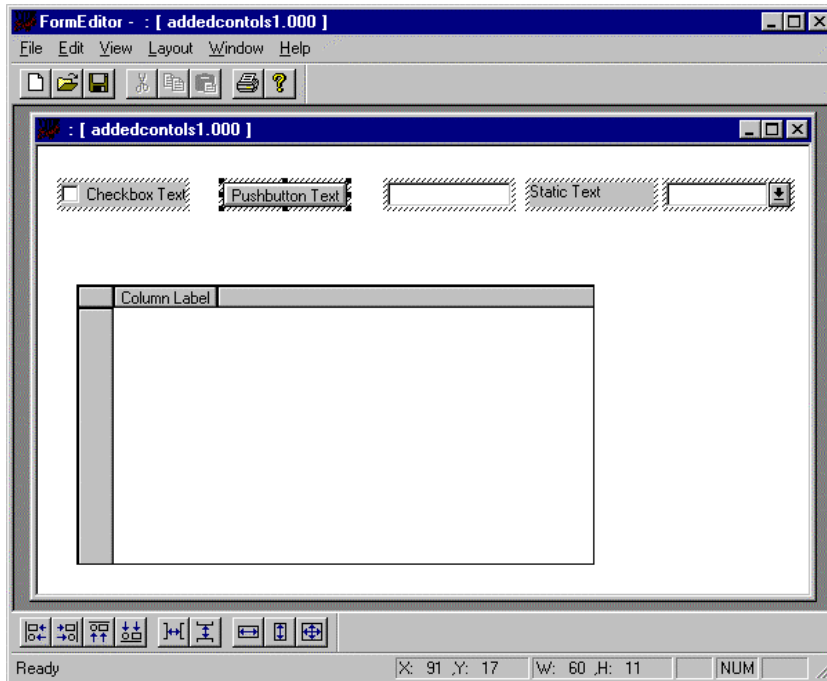
- Click **Make Same Width**: The controls will resize to have the same width.
- Click **Make Same Height**: The controls will resize to have the same height.
- Click **Make Same Size**: The controls will resize to have the same width and height.

Here is a page with some control added, and the controls at the top selected and ready to align.



Page with controls added (shown within the Form Painter)

Here is the same page after the controls at the top of the page (all except the grid) have been aligned vertically.



Page with control aligned vertically (shown within the Form Painter)

CHAPTER 10

Using the Resource Language to Modify Pages

Pages allow users to view and modify data interactively to the Planning system on the server.

Pages contain controls, which allow users to view and modify the data. The controls are usually populated, or get the data that they display, from either an instance of a class or a slot on an instance.



Note: In object-oriented programming, sometimes the terms “object” and “instance” are used interchangeably. This section uses the term “instance” to mean a PepperCode, or SPL, instance.

Menus allow users to select a task to perform, such as adding a unit.

Pages and menus are created with the resource language. You can either create pages and menus from scratch, or you can modify existing pages and menus. Later in this section the resources language is discussed in detail.

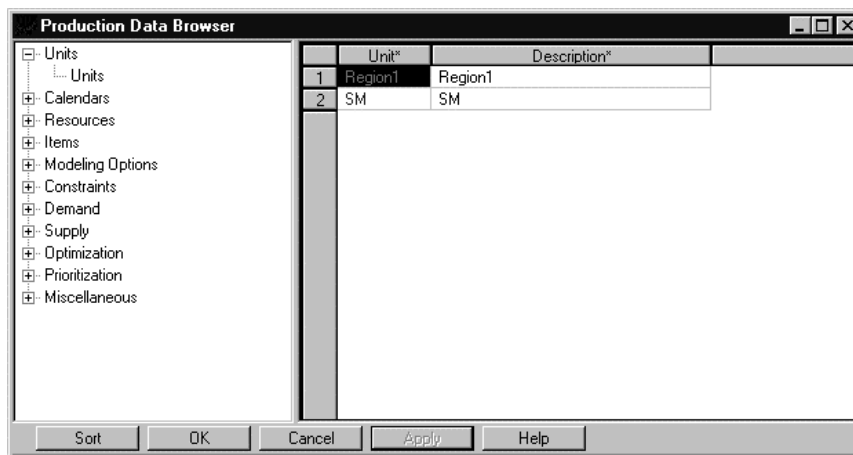
Understanding the Unit Page Code

An example of a page is the Unit page, which displays a unit’s name and description.

The Unit Page is actually a property sheet page. In this documentation, “pages” refers to form pages, property sheet pages, and the data browser. These terms are defined later in this section.

To display this page, do the following:

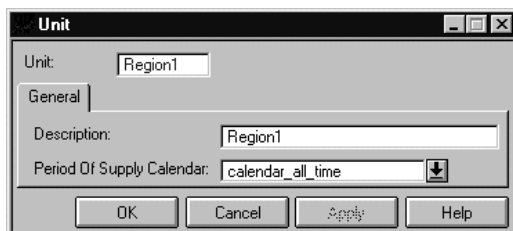
1. Start the Planning Client. The Home Base page appears.
2. Click **Browse Data**. The Data Browser page appears.
3. Click + **Units**. **Units** appears below it and to the right, and + **Units** changes to - **Units**.
4. Click **Units**. A list of units appears on the Data Browser page.



List of Units on the Data Browser page

5. Double-click a unit name. Here, you would double-click Region1 or SM. The Unit page appears.

Here is the Unit page.



Unit page

Following is the resource language code that creates the Unit page. This code is contained in the Site.000 file in the Forms folder.



Note: Units used to be called sites and items used to be called parts, so there will be times in the resource language where the word site appears when you might expect unit, and part appears where you might expect site.

Interspersed in this code are explanations for the tasks that the code performs.

BEGINCLASS and ENDCLASS are not used.

```
[CLASS]

BEGINCLASS

    NAME "Site"

    LABEL "Unit";

ENDCLASS;
```

The BEGINPROPSHEET and ENDPROPSHEET controls create the Unit page. Actually, this is a property sheet page.

```
[PROPERTY SHEET]

BEGINPROPSHEET
```

LABEL places the page label of "Unit" in the top bar of the page.

```
LABEL "Unit";
```

The STATIC control and the EDIT control display the unit name on the page.

Use the STATIC control to display "Unit:".

```
STATIC

    LABEL "Unit:"

    LOCATION 5, 5, 45, 11;
```

Use the EDIT control to display the name of this instance of the site class. The TYPE of this edit field is an instance. When an instance populates an edit field, the value shown in the field is the display_name slot on that instance. NOEDIT means that the user cannot edit the value shown in this field. VALUEPOPULATE populates the edit field with PATH ".", the "." pointing to the topic instance of the edit field.

```
EDIT

    NAME "Unit"

    LOCATION 50, 5, 45, 11

    TYPE "RPS_INSTANCE"

    NOEDIT

    VALUEPOPULATE PATH ".";
```

The BEGINPAGE and ENDPAGE controls place a page on the page.

```
BEGINPAGE
```

LABEL places the page label of "General" in the page's tab.

```
LABEL "General";
```

The STATIC control and the EDIT control show the unit description on the page.

Use the STATIC control to show "Description:".

```
STATIC
```

```
LABEL "Description:"
```

```
LOCATION 5, 5, 45, 11;
```

Use the EDIT control to display the description of this instance of the site class. The TYPE of this edit field is a character string. VALUEPOPULATE populates the edit field with PATH "description", which points to the description slot of the topic instance of the edit field. The user can type a new description into the edit field, and ONAPPLY sends a transaction to the server that updates the description slot of this unit with the new description.

```
EDIT
```

```
NAME "Description"
```

```
LOCATION 50, 5, 135, 11
```

```
TYPE "RPS_STRING"
```

```
VALUEPOPULATE PATH "description"
```

```
ONAPPLY{TRANSACTION "form_set_slot_string(:object %[] .Topic
    :slot_name \"description\"
    :value %[] .Value)"};
```

Use the STATIC control to show "Period Of Supply Calendar:"

```
STATIC
```

```
LABEL "Period Of Supply Calendar:"
```

```
LOCATION 5, 20, 94, 11;
```

Use the COMBOBOX control to display the desired Period Of Supply calendar. The TYPE of this edit field is an instance. VALUEPOPULATE and MENUPOPULATE populate the COMBOBOX with instances of the Calendar class. The user can select a calendar with this COMBOBOX, and ONAPPLY sends a transaction to the server that updates the calendar of this unit with the new calendar.

COMBOBOX

NAME "Period Of Supply Calendar"

TYPE "RPS_INSTANCE"

LOCATION 95, 20, 110, 11

ONAPPLY{TRANSACTION "ui_change_site_period_of_supply_calendar(

:site %[] .Topic

:period_of_supply_calendar %[] .Value)"}

VALUEPOPULATE PATH "period_of_supply_calendar"

MENUPOPULATE TRANSACTION "form_get_instances_of_class(

:class_name \"Calendar\");

ENDPAGE and ENDPROPSHEET mark the end of this page.

ENDPAGE;

ENDPROPSHEET;

Understanding the Unit Page Popup Menu Code

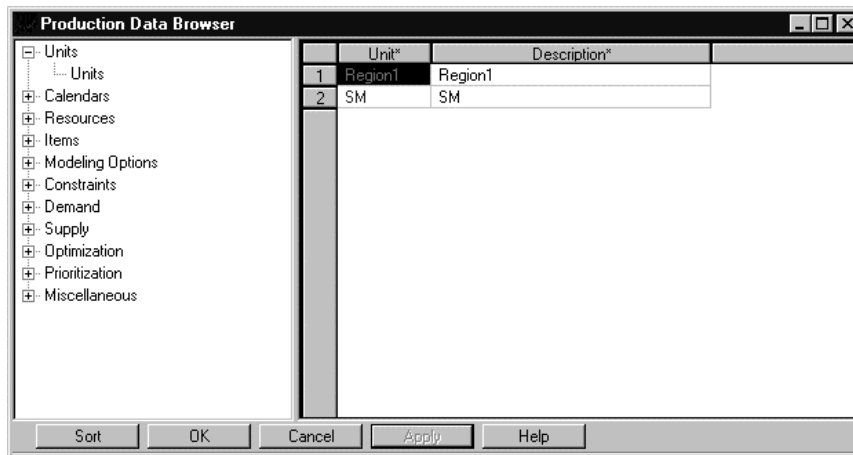
A page that is used to display instances of a class can have a set menu and an instance menu associated with it. The set menu is displayed when you reference a set of instances of this class; an instance menu is displayed when you reference a single instance of this class.

The Unit page is used to display the site class. It has the following menus:

- Unit set menu. This is the set menu for the site class; it allows users to display the Add Unit page, which lets them add a unit.
- Unit instances menu. This is the instance menu for the site class; it allows users to display the Unit page.

To display these menus, do the following:

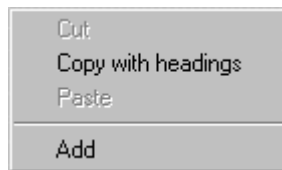
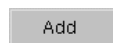
1. Start the Planning Client. The Home Base page appears.
2. Click **Browse Data**. The Data Browser page appears.
3. Double-click + **Units**. **Units** appears below it, and + **Units** changes to - **Units**.
4. To display the Unit set menu, right-click **Units** or on the corner of the Units list.
5. To display the Unit instance menu, right-click one of the unit names. Here, you would right-click Region1 or SM.



Right-click to make menus appear—Data Browser page

Understanding the Unit Set Menu

The Unit set menu can have two appearances:



or Unit Set Menu

Following is the resource language code that creates the Unit set menu. This code is contained in the Site file in the Forms directory.

The BEGINMENU and ENDMENU controls create a pop-up menu to allow users to add a unit.

```
[SETMENU]
```

```
BEGINMENU
```

Use the MENUITEM control to place a menu item within the menu. LABEL places the menu label of **Add** in the menu item. When the user clicks **Add**, ONPICK executes LOADFORM, which loads and displays the Add Unit page.

```
MENUITEM
```

```
    LABEL "Add"
```

```
    ONPICK{LOADFORM("AddSite")};
```

```
ENDMENU;
```

Here is the Unit instance menu.



Unit Instance Menu

Understanding the Unit Instance Menu

Following is the resource language code that creates the Unit instance menu. This code is contained in the Site.000 file in the Forms directory.

The BEGINMENU and ENDMENU control statements create a pop-up menu to allow users to view the unit's properties.

```
[INSTANCEMENU]
```

```
BEGINMENU
```

Use the MENUITEM control to place a menu item within the menu. LABEL places the menu label of "Unit Scorecard" in one menu item, and "Properties" in the other menu item. When the user clicks **Unit Scorecard**, ONPICK executes the first LOADPROPSHEET, which loads and displays the Unit (site) scorecard page. When the user clicks **Properties**, ONPICK executes the other LOADPROPSHEET, which loads and displays the Unit (site) page.

```
MENUITEM
```

```
    LABEL "Unit Scorecard"
```

```
    ALWAYSEDIT
```

```
    ONPICK {LOADPROPSHEET ("Site_Scorecard", :Passed_Site_Or_Site_Node  
%[] .PopupMenuTopic)};
```

```
MENUITEM
```

```
    LABEL "Properties"
```

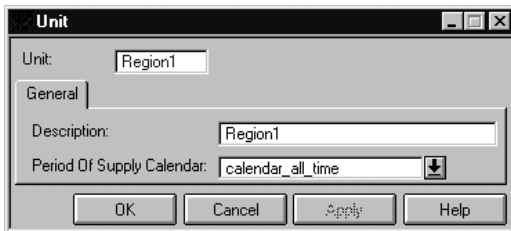
```
    ONPICK{LOADPROPSHEET ("Site", :ParentsTopic %[] .PopupMenuTopic)};
```

```
ENDMENU;
```

Using the Resource Language

An application, such as Planning, contains pages—forms or property sheets—such as the Unit page, which contain controls, such as STATIC and EDIT fields. A page can also be a component by having the page control added to it, and then each page, or page, within the component can

contain controls. Here is the Unit page again, showing the placement of controls on a page. The controls on the Unit page are the Static control, which displays fixed text, the Edit control, which displays an edit field, the Combobox control, which displays a drop-down list, and the Page control, which displays a page.



Page with controls

A page is the window upon which controls are laid; the controls then display data—usually data from the server—and allow users to input and modify data on the server. You create pages, controls, and menus with the resource language.

Populating Pages and Controls

To populate is to associate a page or control with data from the server. A page is generally associated with a topic, and a control is generally associated with a topic and a value.

Populating a Page with a Topic

A page can be populated with a topic. A topic is an instance. An example of this is the About Box page, which is populated with an instance of the About_Box class.



For more information about the About Box page, refer to Understanding the About Box Page Code.

To populate a page with its topic, use the TOPICPOPULATE property.



For the syntax of TOPICPOPULATE, refer to Writing Statement Properties.

Property sheet pages have their topic populated automatically. Property sheet pages are passed a parameter named ParentsTopic when they are loaded—using LOADPROPSHEET—from an instance menu or by double-clicking a instance that is the value of a control. For example, the Unit page code has no TOPICPOPULATE method defined in the resource language; this is because the Unit page is a property sheet page that automatically received its topic from the ParentsTopic parameter.



For more information about LOADPROPSHEET, refer to Writing Event Commands.

Populating a Control with a Topic and Value

A control is populated with a topic and a value. A value is generally the value of a slot on an instance.

When you populate a page with a topic, each control on that page is automatically populated with that topic as well if no TOPICPOPULATE property exists for that control; you can use the slots from the page's topic to populate the control with values. An example of this is the STATIC control in the About Box page that lists the product name, which is populated with the value of the product_name slot from the topic instance of the About Box page. To populate a control with its value, use the VALUEPOPULATE property.



For the About Box page code, refer to Understanding the About Box Page Code. For the syntax of VALUEPOPULATE, refer to Writing Statement Properties.

If you want to display a value from a slot that is not in the page's topic, you can populate the control with another topic in order to get that value. An example of this is the Early Fence and Late Fence EDIT fields on the Score Card page. The Score Card page is populated by the mfg scorecard instance, but this instance does not include early_fence or late_fence, which are slots on the main environment instance. The EDIT controls on the Score Card page that display early_fence and late_fence use TOPICPOPULATE and VALUEPOPULATE: TOPICPOPULATE gets the topic of the control—main environment instance—and VALUEPOPULATE gets the values for the controls: early_fence slot, late_fence slot.

You can also populate a control with a calculated value, when the value that you want to display is not available in a slot. An example of this is the penalty values displayed in the Score Card page. The value to be displayed must be obtained with a transaction because it is not directly available on a slot; the transaction ui_get_penalty_value must calculate it.



For the Score Card page code for these EDIT fields and displaying these penalty values, refer to Understanding the Score Card Page Code.

Displaying Pages and Menus

This section explains how to display pages and menus. To display pages, you, the resource language programmer, use commands such as LOADFORM and LOADPROPSHEET, and the user selects a menu item from the top menu or a pop-up menu; to display menus, you define the menu in the appropriate file and the user right-clicks classes or instances.

Displaying Pages

To display a form page, use the LOADFORM command.



For the code for a page that contains LOADFORM, refer to Understanding the Unit Set Menu. For the syntax of LOADFORM, refer to Writing Event Commands.

To display a property sheet page, use the LOADPROPSHEET command. Property sheet pages differ from form pages in that property sheet pages are used to display data about an instance, and they can contain set and instance menus, which are used to access instance data.



For the code for a page that contains LOADPROPSHEET, refer to Understanding the Purchase Order Page Code. For the syntax of LOADPROPSHEET, refer to Writing Event Commands.

You must also have the filename for the page in the Catalog file.



For instructions about inserting the filenames, refer to Modifying the Catalog File.

If the page is in a Class file, you can display the page by double-clicking an instance of that class. For example, the Unit page is contained in a Class file. From the Data Browser, you clicked first on + **Units**, and you clicked **Units** to display a list of units, and then you double-clicked a unit name to display the page for that unit.

Displaying Menus

There are two types of menus: set menus and instance menus.

Displaying Set Menus

A set menu is a menu that you want displayed when you reference sets of instances of this class. For example, the Unit set menu in the Unit (Site) page is a set menu. To display a list of units, you first clicked + **Units** and then **Units**. If you right-clicked **Units** instead, the Unit set menu is displayed. Where you would double-click to list a set of instances of a class, you can right-click to display the set menu.

Displaying Instance Menus

An instance menu is a menu that you want displayed when you reference a single instance of a class. For example, when you displayed the Unit page from the Data Browser page, you first

clicked + **Units**, and you clicked **Units** to display a list of units, and then you double-clicked a unit name to display the page for that unit. If you right-clicked the name instead, the Unit instance menu is displayed. Where you would double-click an instance to show the page for an instance of that class, you can right-click to display the instance menu.

Writing a Resource Language File

When you write a resource language file, you can use whatever editor you wish. However, you must save the file as an ASCII text file. The file name must conform to the naming conventions supported by Win95 and WinNT. The last 4 characters of a resource file name consist of a period followed by three numbers; for example, resfilename.000. These numbers are used as the version of the file.

You do not have to compile the resource language file. It only needs to be placed in the proper resources directory. Your system has an RPS75 folder, which contains Planing folders. Each Planning folder—PRA, ERA, SRA—contains a file titled “Forms”, and those folders contain the resource language files.

Building Resource Language Files

There are two main types of resource language files that you build for the Planning client: Form files and Class files.

Form Files Contain Form Pages

A form file contains the code that you want in a form page. The structure for a form file is shown below.

```
[FORM]

BEGINFORM

// Here is where you place the statements that build your form.

// Notice that comments start with "//".

;

ENDFORM;
```



For several examples of pages using BEGINFORM, refer to Understanding Examples Of Resource Language Code.

After you create the file, you must enter its file name into the FORMS section of the Catalog file.



For information about where you enter the filename into the Catalog file, refer to Modifying the Catalog File.

Class Files Contain Property Sheet Pages and Menus

Class files are similar to form files, but they have some additional structure for a set menu and an instance menu.

```
BEGINPROPSHEET

// Here is where you place the statements that build your property sheet page.

// Notice that comments start with "//".

;

ENDPROPSHEET;

[SETMENU]

BEGINMENU

// Here is where you place the code for the set menu.

;

ENDMENU;

[INSTANCEMENU]

BEGINMENU

// Here is where you place the code for an instance menu.

;

ENDMENU;
```

For an example of the set and instance menu code, refer to Understanding the Unit Popup Menu Code.

After you create the file, you must enter its file name into the PROPERTYSHEET section of the Catalog file. You must also enter the file name into the CLASS section of the Catalog file, due to the set menu and the instance menu.



For information about where you enter the filename into the Catalog file, refer to Modifying the Catalog File.

Modifying the Catalog File

The Catalog file lists all of the Resource Files used by a Client. It also defines a set of global properties.

The Catalog file has the following format:

[VERSION]

This section contains the version number of the Catalog file.

[OBJECTBROWSER]

This section contains the name of the file that contains the Resource Language for the Data Browser.

[INITIALFORM]

This section contains the file name of the first page displayed when the client connects to a server.

[TOPMENU]

This section contains the file name of the file that contains the Resource Language for the top level menu displayed when the client is connected to a server.

[ENVFORM]

This section contains the file name of the file that contains the Resource Language for the environments page. This page is used when you connect to an empty server, and you wish to create a main environment.

[FLOATFORMAT]

This section contains the default format string to be used when displaying floating point numbers.

[CLASSES]

This section contains the list of all of the class files to be used by the client.

[FORMS]

This section contains the list of all the form files for the form pages to be used by the client.

[PROPERTY SHEETS]

This section contains the list of all class files for the property sheets to be used by the client.

The Catalog file is the only file that should contain resource file name references with version numbers appended; for example, .000. All other resource file name references should use only the base name without the version number, as in LOADFORM commands.

Writing Resource Language Statements

Statements in the resource language begin with a keyword and end with a semicolon. Statements can span multiple lines. Using multiple lines can make a statement easier to read. For example, the following two statements are the same, but the second spans multiple lines:

```
STATIC LABEL "Unit:" LOCATION 5, 5, 45, 11;
```

```
STATIC
```

```
    LABEL "Unit:"
```

```
    LOCATION 5, 5, 45, 11;
```

A statement may contain more than one keyword. Keywords may exist by themselves, such as LEFT, or may require values, such as LABEL "Unit:".

Statements whose keywords start with BEGIN and END—such as BEGINFORM and ENDFORM—mark the beginning and end of structures such as pages and menus. For example:

```
BEGINSUBMENU
```

```
    LABEL "&Reports";
```

```
//Other statements that describe the submenu go here.
```

```
ENDSUBMENU;
```

Statements can contain the following attributes:

- **Properties.** These can be, for example, the location of a control or the type of data shown in a control. Controls are created by statements; these are referred to as control statements.
- **Events.** These have the statement perform an action of some type. For example, a statement that creates a button uses an event to perform an action when the user presses the button. Events can contain commands, which are executed when the event occurs. Commands can contain methods, which perform actions upon control statements. Commands can also contain parameters.

Writing Statement Properties

Statements can contain properties, such as a location or a type. Following are some properties that are used by many statements:

Properties for Resource Language Statements

Property	Description
LOCATION	<p>The LOCATION property determines the location and size of the control placed on a page. It uses the following syntax:</p> <p>LOCATION left, top, length, height</p> <p>where left is the number of units from the left edge of the page or page, top is the number of units from the top edge of the page or page, length is the horizontal length of the control, and height is the height of the control. One character consists of about 4.5 units.</p> <hr/> <p>For a more detailed discussion about setting this property, refer to Understanding Pages Formatting Standards.</p> <hr/>
TYPE	<p>The TYPE property determines the type of data that is going to be displayed in a control such as an EDIT statement. All of these data types display SPL types, not C or C++ data types. It uses the following syntax:</p> <p>TYPE "datatype"</p> <p>where datatype is the type of data to display. The types are:</p> <p>RPS_CLASS displays a class.</p> <p>RPS_DATE displays a date and time.</p> <p>RPS_ENUM displays an enumeration.</p> <p>RPS_FLOAT displays a float.</p> <p>RPS_INSTANCE displays an instance; the default to display for a named instance is the display_name slot for that instance.</p> <p>RPS_INT displays an integer.</p> <p>RPS_STRING displays a character string.</p> <p>RPS_TIME displays a time.</p>
LABEL	<p>The LABEL property displays a fixed character string. Several statements use</p>

Property	Description
	<p>the LABEL property, such as STATIC and BEGINPAGE. It uses the following syntax:</p> <pre>LABEL "Text to display"</pre> <p>The LABEL property in the following STATIC statement causes it to display the text string "Unit:".</p> <pre>STATIC LABEL "Unit:" LOCATION 5, 5, 45, 11;</pre>
NAME	<p>The NAME property allows you to name a statement; usually, this statement is a control. This name must be unique inside the page in which it is used. Naming a control allows you to reference that control from other controls.</p> <p>The NAME of "Load" in the PUSHBUTTON statement is referenced in the ONAPPLY event; when the user presses the Apply or OK button, the METHOD keyword enables the button named "Load".</p> <pre>ONAPPLY { METHOD %[Load].Enable()};</pre> <pre>PUSHBUTTON NAME "Load" LABEL "Load..." LOCATION 182, 5, 45, 14 ONPRESS {COMMAND (32778)};</pre>
INVISIBLE	<p>Makes a control invisible. You can then make it visible with the Show method. Since this can be used with any control, this property is not listed in the later sections that list the properties for each control statement.</p>
NOAPPLY	<p>If the user makes changes to the value shown in this control, the Apply button on</p>

Property	Description
	the page remains disabled.
NOAUTOPOPULATE	<p>When this control is first displayed, this control will not be automatically populated, leaving the control blank. This property is often used with the SENDPARENT; a control in the parent page uses NOAUTOPULATE so that it will not be populated until the child page sends data back to the parent. This property is also often used with the Refresh method; the control is not populated until the user presses a button, whose PUSHBUTTON contains a Refresh for that control.</p> <hr/> <p>For the SENDPARENT syntax, refer to Writing Event Commands. For an example of this used in a page, refer to Understanding the Item Filter Page Code.</p> <hr/>
NOEDIT	Prevents the user from changing the data shown in the control.
REQUIRED	This property prevents the user from leaving the page without entering data in this control. A Required error message box appears if the user tries to leave the page without filling out this control.
DISABLED	Causes the initial state of this control to be disabled.
LEFT/CENTER/RIGHT	LEFT left-justifies the data shown in the control; CENTER centers it; RIGHT right-justifies it.
TOPICPOPULATE	<p>The TOPICPOPULATE property gives a page or control its topic. The topic is an instance on the server. TOPICPOPULATE uses the following attributes: TRANSACTION, CONTROL, PARAMETER, QUERY, FORM, NONE</p> <hr/> <p>For more information about topics and populating, refer to Populating Pages and Controls.</p> <hr/>

Property	Description
	<p>TRANSACTION: This specifies a transaction that, when sent to the server, retrieves the Topic object. It uses the following syntax, where transaction is the transaction to send to the server:</p> <pre>TOPICPOPULATE TRANSACTION "transaction"</pre> <hr/> <p>Refer to Understanding the Control panel Code for an example.</p> <hr/> <p>When you write a transaction to use with TOPICPOPULATE, that transaction must return an instance named "value". For example, the following transaction can populate the Score Card page. It returns an output named "value" that is an instance of Score_Card.</p> <pre>action ui_get_mfg_scorecard (local: action<get_mfg_scorecard> get_mfg_scorecard,</pre>
	<pre> output: instance<Score_Card> value, no_context:) { execute get_mfg_scorecard(); value = GET_INSTANCE_BY_NAME("mfg_score_card") ; succeed(); } "get_oid" retrieves a Topic object: TOPICPOPULATE TRANSACTION get_oid object_name;</pre>

Property	Description
	<p>TOPICPOPULATE TRANSACTION get_oid object_name.slot_name;</p> <p>CONTROL: This specifies that the topic of a control will be the same as another control. It uses the following syntax, where control_name is the name of the other control:</p> <p>TOPICPOPULATE CONTROL "%[control_name]" or</p> <p>TOPICPOPULATE CONTROL "%[control_name].topic"</p> <hr/> <p>For an example of using TOPICPOPULATE CONTROL, refer to Understanding the Score Card Page Code.</p> <hr/>
	<p>PARAMETER: This specifies that in this page, a topic object has been passed in through the parameters list, and the Topic of this control is retrieved from the parameters list. It uses the following syntax, where parameter_name is the name of the passed-in parameter:</p> <p>TOPICPOPULATE PARAMETER :parameter_name</p> <hr/> <p>For an example of using TOPICPOPULATE PARAMETER, refer to Understanding the Inventory Report Page Code.</p> <hr/> <p>QUERY: This specifies that the results of a query will be the topic of this control. It uses the following syntax, where query is the query:</p> <p>TOPICPOPULATE QUERY "query"</p> <hr/> <p>For the syntax of a query, refer to Writing Queries. For an example of TOPICPOPULATE QUERY, refer to Understanding the Item Filter Page Code.</p> <hr/>

Property	Description
	<p>This is used most commonly to populate grids, histograms, and Gantts.</p> <p>When you write a query to use with TOPICPOPULATE, that query returns an oset of instances which will allow the control to be populated with the instances.</p> <hr/> <p>FORM: This specifies that the topic of this control is the topic of the page.</p> <hr/>
	<p>NONE: This specifies that this control does not have a topic. This is often used to collect data that will be sent to the server. It uses the following syntax:</p> <p>TOPICPOPULATE NONE</p> <hr/> <p>For an example, refer to Understanding the Add Unit Page Code.</p> <hr/>
VALUEPOPULATE	<p>The VALUEPOPULATE property gives a control its value. VALUEPOPULATE uses the following attributes: TRANSACTION, PARAMETER, PATH</p> <hr/> <p>For more information about values and populating, refer to Populating Pages and Controls.</p> <hr/> <p>TRANSACTION: This specifies a transaction that, when sent to the server, retrieves the value. It uses the following syntax, where transaction is the transaction to send to the server:</p> <p>VALUEPOPULATE TRANSACTION "transaction"</p> <hr/> <p>For an example of VALUEPOPULATE TRANSACTION, refer to Understanding the Score Card Page Code.</p> <hr/>
	<p>You can write your own transactions, or you can use the transactions listed in</p>

Property	Description
	<p>Using Standard Pages Transactions, that get data from the server. These transactions have as input an instance—which will normally be the topic of the page—and have as output some attribute of that instance—such as an integer slot—named "value". When you write a transaction to use with VALUEPOPULATE, that transaction must return an output named "value".</p> <p>When a VALUEPOPULATE uses a TRANSACTION attribute to populate from a class that is added at run-time, it must also use a PATH attribute so that the client knows the correct class.</p> <p>Use PATH with VALUEPOPULATE TRANSACTION to subscribe to the data values; refer to PATH in this section for a description of subscribe.</p> <p>“get_oid” can be used to retrieve an object or the value of a slot:</p>
	<p>VALUEPOPULATE TRANSACTION get_oid object_name;</p> <p>VALUEPOPULATE TRANSACTION get_oid object_name.slot_name;</p> <p>PARAMETER: This specifies that in this page, a value has been passed in through the parameters list, and the value of this control is retrieved from the parameters list. It uses the following syntax, where parameter_name is the name of the passed-in parameter:</p> <p>VALUEPOPULATE PARAMETER :parameter_name</p> <hr/> <p>For an example of VALUEPOPULATE PARAMETER, refer to Understanding the Inventory Report Page Code.</p> <hr/>
	<p>PATH: This specifies a slot on the Topic for this control to display.</p> <p>VALUEPOPULATE PATH subscribes to</p>

Property	Description
	<p>the slot, allowing the values to be refreshed when they are displayed in more than one place at the same time. When the value changes on the server, the control is refreshed to show the new value. For example, two clients can show the same unit in the Unit page, thus displaying one unit from the server on two Unit pages. The Unit page uses VALUEPOPULATE PATH to populate its edit fields; when one client changes a unit description, the other client will have the unit description refreshed to show the new value.</p> <hr/> <p>For an example of VALUEPOPULATE PATH, refer to Understanding the Control panel Code.</p> <hr/>
	<p>It uses the following syntax, where path_name is the path name:</p> <pre>VALUEPOPULATE PATH "slot_name"</pre> <p>The slot_name is a slot on the instance that is the Topic of this control. slot_name can also consists of several slot names, separated by periods '.', that lead to one slot. For example, the following VALUEPOPULATE property from the Purchase Order page populates with the site (unit) instance, which is a slot on the part (item) instance, which is a slot on the instance that is the topic of this control.</p> <pre>VALUEPOPULATE PATH "part.site";</pre>
	<p>For the code where this is used, refer to Understanding the Purchase Order Page Code.</p> <p>The slot_name can also be a single period, as follows:</p> <pre>VALUEPOPULATE PATH "."</pre> <p>This PATH populates with the value of the default slot for the topic of the control; for an instance, the slot is display_name.</p>

Property	Description
	<p>For the code where this is used, refer to Understanding the Purchase Order Page Code.</p>
	<p>When PATH cannot display the value that you want, you can use the following syntax:</p> <pre>VALUEPOPULATE TRANSACTION "transaction" PATH "slot_name"</pre> <p>where transaction is the transaction to get the actual data value to display. For example, the Score Card page shows a VALUEPOPULATE that uses the ui_get_penalty_value transaction to get the value for display, and uses PATH to subscribe to the penalty slot.</p> <p>For an example of this, refer to Understanding the Score Card Page Code.</p>
MENUPOPULATE	<p>The MENUPOPULATE property gives a COMBOBOX control statement an osset of values to display; an osset of enums, an osset of instances, or an osset of classes.</p> <p>MENUPOPULATE uses the following attributes: TRANSACTION</p> <p>TRANSACTION: This specifies a transaction that, when sent to the server, retrieves the osset of values to display in the drop-down list. It uses the following syntax, where transaction is the transaction to send to the server:</p> <pre>MENUPOPULATE TRANSACTION "transaction"</pre>
	<p>For examples, refer to Understanding the Add Unit Page Code and Understanding the Purchase Order Page Code.</p> <p>Some transactions you can use are listed in Using Standard Pages Transactions:</p>

Property	Description
	<p>form_get_subclasses_of_class, form_get_enums_of_class, and form_get_instances_of_class. Using these transactions will display the display_name slot. If you wish to display something other than the display_name slot within the COMBOBOX drop-down list, make sure that the transaction returns the following:</p> <p>value, where value is an osset of enums, instances, or classes.</p> <p>hr_strings, where hr_strings is an osset of STRING that is displayed in the drop-down list.</p>

Writing Statement Events

A statement can contain events. An event is generated when a certain action is performed, such as pressing a button. An event contains a command, or list of commands, that is executed when the event is generated; this allows you to execute commands when that certain action is performed.

The syntax of an event is:

```
EVENT { command }
```

where EVENT is the name of the event, and command is either a command or a list of commands separated by commas.

The following ONPICK event from the Unit (site) set menu loads the AddSite page—which enables a user to add a unit—when users choose "Add" from that menu.

```
MENUITEM
    LABEL "Add"
    ONPICK{LOADFORM("AddSite")};
```

Events for Resource Language Statements

Event	Description
ONAPPLY	The ONAPPLY event occurs when the Apply or OK button is pressed on a page. When a control has an ONAPPLY event,

Event	Description
	<p>the value in that control has to have changed in order for its ONAPPLY event to occur.</p> <hr/> <p>For an example, refer to Understanding the Add Unit Page Code.</p> <hr/>
ONCHANGE	<p>The ONCHANGE event occurs when a user changes the value of a control.</p> <hr/> <p>For an example, refer to Understanding the Control panel Code.</p> <hr/>
ONCLOSE	<p>The ONCLOSE event occurs when a user closes a page. Unlike the other events, ONCLOSE can only execute a single TRANSACTION command.</p> <hr/> <p>For an example, refer to Understanding the Build Schedule (with Resource) Spreadsheet Page Code.</p> <hr/>
ONOK	<p>The ONOK event occurs when the OK button is pressed on a page.</p>
ONPICK	<p>The ONPICK event occurs when a user chooses an item from a menu.</p> <hr/> <p>For an example, refer to Understanding the Purchase Order Page Code.</p> <hr/>
ONPRESS	<p>The ONPRESS event occurs when a button is pressed. This is used with the PUSHBUTTON control statement.</p> <hr/> <p>For an example, refer to Understanding the Purchase Order Page Code.</p> <hr/>
ONHELP	<p>The ONHELP event occurs when the HELP button is pressed on a page.</p>
ONCANCEL	<p>The ONCANCEL event occurs when the CANCEL button is pressed on a page.</p>

There are also events that are used only with specific controls, such as with the RESOURCE_GANTT and TASK_GANTT controls.

Writing Event Commands

An event can contain commands. These commands are executed when the event occurs.

Commands for Resource Language Events

Command	Description
ASKUSER	<p>ASKUSER is used to display a Yes/No dialog box to the user. If the user answers no, the processing of a list of commands in the current statement is stopped. The syntax is:</p> <pre>ASKUSER "Question", commands</pre> <p>where Question is the question to ask the user, and commands is the command or list of commands that is executed if the user answers yes, and is not executed if the user answers no.</p> <hr/> <p>For an example, refer to Understanding the Purchase Order Page Code.</p> <hr/>
COMMAND	<p>COMMAND is used to execute a series of pre-defined C++ actions. The syntax is:</p> <pre>COMMAND (00000)</pre> <p>where 00000 is a number a specific series of C++ actions.</p> <hr/> <p>For an example, refer to Understanding the Control panel Code.</p> <hr/>
PROGRESS	<p>PROGRESS keyword specifies a transaction text string which is sent to the server thus allowing the use of the Optimization progress dialog box as a generalized progress dialog box.</p> <p>PROGRESS replaces COMMAND (32774).</p>

Command	Description
	<p>The following example is taken from Optimize.000. Formerly, you would use:</p> <pre>PUSHBUTTON</pre> <pre>LABEL "Optimize Allowing New Supply"</pre> <pre>LOCATION 40, 5, 135, 14</pre> <pre>ONPRESS {</pre> <pre>TRANSACTION</pre> <pre>"transaction_set_mfg_master_planning(:</pre> <pre>master_planning \"TRUE\")",</pre> <pre>COMMAND (32774)};</pre> <p>TRANSACTION's text string will be the progress transaction on the server.</p> <p>You can replace COMMAND as in the following example:</p> <pre>PUSHBUTTON</pre> <pre>LABEL "Optimize Allowing New Supply"</pre> <pre>LOCATION 40, 5, 135, 14</pre> <pre>ONPRESS {</pre> <pre>TRANSACTION</pre> <pre>"transaction_set_mfg_master_planning(:</pre> <pre>master_planning \"TRUE\")",</pre> <pre>PROGRESS LABEL "Optimize Allowing New</pre> <pre>Supply "</pre> <pre>TRANSACTION "</pre> <pre>transaction_deconflict()"};</pre>
EXEC APPLICATION	<p>EXEC APPLICATION starts an application program. The syntax is:</p> <pre>EXEC APPLICATION "application name"</pre> <p>where application name is the directory path up to and including the application program.</p> <p>The following example opens the WordPad application, provided that the directory path is accurate:</p> <pre>ONPRESS</pre> <pre>{EXEC APPLICATION "C:\\Program</pre> <pre>Files\\Accessories\\Wordpad.exe"}</pre> <p>The backslash is used as an escape</p>

Command	Description
	<p>character in resource language strings; therefore, when a backslash is used in the path to a document, the backslash must be escaped with another backslash.</p> <p>Note: EXEC APPLICATION currently only works with ONPRESS, which is used with PUSHBUTTON.</p>
EXEC DOCUMENT	<p>EXEC DOCUMENT starts the registered program for the document type and loads the document. The syntax is:</p> <pre>EXEC DOCUMENT "document name"</pre> <p>The following example takes the Monthly.rpt document, start its registered program, and loads Monthly.rpt into that program.</p> <pre>ONPRESS{ EXEC DOCUMENT "C:\MyDocs\Monthly.rpt" }</pre> <p>The following example takes the document name www.peoplesoft.com—the URL for the PeopleSoft home page—and starts your web browser program, and then brings up the PeopleSoft home page in the web browser.</p> <pre>ONPRESS{ EXEC DOCUMENT "WWW.PEOPLESOFT.COM" }</pre> <p>The backslash is used as an escape character in resource language strings; therefore, when a backslash is used in the path to a document, the backslash must be escaped with another backslash.</p> <p>Note: EXEC DOCUMENT currently only works with ONPRESS, which is used with PUSHBUTTON.</p>
LOADFORM	<p>The LOADFORM command loads a form page. The complete filename of the page must be in the [FORM] section of the Catalog file. The syntax is:</p> <pre>LOADFORM("filename")</pre> <p>or</p>

Command	Description
	<p>LOADFORM("filename", :Passed_Parameter)</p> <p>where filename is the filename of the form page, and :Passed_Parameter is optional. Passed_Parameter is a parameter; it is often used to pass the loaded page its topic.</p> <hr/> <p>For an example of LOADFORM being passed its topic, refer to Understanding the Score Card Page Code. For a description of parameters, refer to Writing Method Parameters.</p> <hr/> <p>Do not include the three number extension on this filename. Example:</p> <p>Right:</p> <p>LOADFORM("PartFilter")</p> <p>Wrong:</p> <p>LOADFORM("PartFilter.000")</p>
LOADPROPSHEET	<p>LOADPROPSHEET loads a property sheet page. The complete file name must be in the [PROPERTYSHEETS] section of the Catalog file. Example:</p> <p>LOADPROPSHEET("filename")</p> <p>or</p> <p>LOADPROPSHEET("filename", :Passed_Parameter)</p> <p>where filename is the filename of the page, and :Passed_Parameter is optional. Passed_Parameter is a parameter; it is often used to pass the loaded property sheet page its topic.</p> <hr/> <p>For an example of LOADPROPSHEET being passed its topic, refer to Understanding the Purchase Order Page Code. For a description of parameters, refer to Writing Method Parameters.</p> <hr/>

Command	Description
	<p>Do not include the three number extension on this filename. Example:</p> <p>Right:</p> <pre>LOADPROPSHEET("Purchase_Order")</pre> <p>Wrong:</p> <pre>LOADPROPSHEET("Purchase_Order.000")</pre>
METHOD	<p>METHOD is used to execute a method on a statement. To execute a method upon a statement, use the following syntax:</p> <pre>METHOD method</pre> <p>where method is the method to execute.</p> <hr/> <p>For an example, refer to Understanding the Control panel Code.</p> <hr/>
SENDPARENT	<p>The SENDPARENT command sends data from a page—the child page—to the page that created it—the parent page. The parent page contains a LOADFORM or LOADPROPSHEET that loads the child page. The syntax is:</p> <pre>SENDPARENT (%[child_control].method, %[parent_control].method)</pre> <p>%[child_control].method and %[parent_control].method are methods (refer to "Methods" for a description of methods, where child_control is the NAME of a control on the child page, and parent_control is the NAME of a control on the parent page, and method is the name of a method.</p> <hr/> <p>For a description of methods, refer to Writing Methods. For examples and a discussion of how SENDPARENT sends data from a child page to a parent page, refer to Understanding the Item Filter Page Code.</p> <hr/>
TRANSACTION	<p>The TRANSACTION command executes</p>

Command	Description
	<p>a transaction on the Server. The syntax is:</p> <pre>TRANSACTION "transaction"</pre> <p>where transaction is the transaction to execute.</p> <hr/> <p>For an example, refer to Understanding the Add Unit Page Code.</p> <hr/>

Writing Methods

Methods perform actions upon controls. They can also pass parameters in transactions. The syntax for a method is:

```
%[control_name].method_name
```

where:

- control_name is the name of the control in its NAME property. If there is no control_name, the method is executed on the current control.
- method_name is the name of the method to execute.

For example, the following ONAPPLY event executes the ui_add_site transaction, which uses the Value method to get the values for its parameters:

```
ONAPPLY{TRANSACTION "ui_add_site(:site_name %[Site].Value
:site_class %[Class].Value
:description %[Description].Value) "};
```

In this case, Site, Class, and Description are all edit fields or comboboxes that have been given values by the user.



For the page code where this is used, refer to Understanding the Add Unit Page Code.

The following ONCHANGE event disables a control that has the property NAME "Load".

```
ONCHANGE { METHOD %[Load].Disable() }
```

Common methods are:

Methods for Resource Language Controls

Method	Description
ColumnOrder	<p>The OrderBy attribute in the form_sort_slot_instance_list transaction can be set to ColumnOrder. When this transaction is then used in a BEGINGRID's VALUEPOPULATE or TOPICPOPULATE QUERY, the columns in that grid can then be rearranged by the user, and the columns will be sorted in ascending order according to the order of the columns in the grid.</p> <hr/> <p>For more information about form_sort_slot_instance_list, refer to Using Standard Pages Transactions. For examples of ColumnOrder, refer to Understanding the Purchase Order Page Code and Understanding the Item Filter Page Code.</p> <hr/>
Disable	<p>Disables a control.</p> <hr/> <p>For an example, refer to Understanding the Control panel Code.</p> <hr/>
Enable	<p>Enables a control.</p> <hr/> <p>For an example, refer to Understanding the Control panel Code.</p> <hr/>
FormTopic	<p>Returns the topic of the page.</p> <hr/> <p>For an example, refer to Understanding the About Box Page Code.</p> <hr/>
Hide	<p>Hides the control created by a control statement.</p>
PopupMenuTopic	<p>Returns the instance of whatever the user has just clicked. This is commonly used with instance menus, where the user has clicked an instance to display the instance</p>

Method	Description
	<p>menu.</p> <hr/> <p>For an example, refer to Understanding the Purchase Order Page Code.</p> <hr/>
Refresh	<p>Refreshes a field with the current values from the server.</p> <hr/> <p>For an example, refer to Understanding the Purchase Order Page Code.</p> <hr/>
RowTopic	<p>Gets the current topic of a row on a grid.</p> <hr/> <p>For more information about grids, refer to Grids: BEGINGRID, ENDGRID. For an example of RowTopic, refer to Understanding the Purchase Order Page Code.</p> <hr/>
SelectedRow	<p>Returns the instance of the currently selected row in a grid or spreadsheet.</p> <hr/> <p>For an example, refer to Understanding the Build Schedule (with Resource) Spreadsheet Page Code.</p> <hr/>
Show	<p>Shows the control created by a control statement. Used when a control has been hidden with the Hide method or the INVISIBLE property.</p>
Topic	<p>Returns the current topic of a control.</p> <hr/> <p>For an example, refer to Understanding the Score Card Page Code.</p> <hr/>
TopicForceApply	<p>This is used with the SENDPARENT command, which sends values from a control on a child page to a control on a parent page. When a user changes a control on a page, the Apply button is enabled. When a control on a parent page has its value changed by SENDPARENT from a child page, the TopicForceApply</p>

Method	Description
	<p>in the SENDPARENT forces the Apply button on the parent page to be enabled.</p> <hr/> <p>For an example, refer to Understanding the Item Filter Page Code.</p> <hr/>
Value	<p>Returns the current value of a control.</p> <hr/> <p>For an example, refer to Understanding the Add Unit Page Code.</p> <hr/>
ValueAtRow	<p>Returns the value of a row.</p> <hr/> <p>For an example, refer to Understanding the Build Schedule (with Resource) Spreadsheet Page Code.</p> <hr/>
ValueAtRowColumn	<p>Returns the value of one field in the grid.</p> <hr/> <p>For more information, refer to the events for Grids: BEGINGRID, ENDGRID.</p> <hr/>

Writing Method Parameters

A command can contain parameters. The syntax for parameter is:

```
:parameter_name method
```

where parameter_name is the parameter name, and method is the method used to get the parameter value.

You can define a parameter and assign a value to it. For example, the instance menu for the Inventory Item page contains the following menu item.

```
MENUITEM
```

```
  LABEL "Inventory Report"
```

```
  ONPICK{LOADFORM ("InventoryReport", :Passed_Part %[] .PopupMenuTopic)};
```

The parameter in the LOADFORM, :Passed_Part %[] .PopupMenuTopic, says to pass the Topic of this menu to the Inventory Report page in the parameter Passed_Part . In the InventoryReport page is the following code:

```
[FORM]

BEGINFORM

LABEL "Inventory Report"

TOPICPOPULATE PARAMETER :Passed_Part;
```

This causes the topic of this page to be the Topic in Passed_Part.

Parameters are also used to refer to parameters in transaction calls—which are pre-defined, not defined by the user—as in the following code from the Add Unit (site) page:

```
ONAPPLY{TRANSACTION "ui_add_site(:site_name %[Site].Value

:site_class %[Class].Value

:description %[Description].Value)"};

:period_of_supply_calendar %[Calendar].Value)"};
```

Site, Class, Description, and Calendar are all edit fields or comboboxes that have been given values by the user. These values are passed into the transaction, so that this transaction can add a unit.

Writing Menu Statements

A menu consists several statements: BEGINMENU and ENDMENU, BEGINSUBMENU and ENDSUBMENU, SEPARATOR and MENUITEM.

Menus: BEGINMENU, ENDMENU

The BEGINMENU and ENDMENU statements create a menu.



For examples, refer to Understanding the Unit Set Menu and Understanding the Unit Instance Menu.

The BEGINMENU and ENDMENU statements can contain the following keywords:

- BEGINSUBMENU and ENDSUBMENU
- SEPARATOR
- MENUITEM

The BEGINMENU statement has the LABEL property, which allows you to give a label to the Top Menu. This label appears at the top of the menu. LABEL is not used in the BEGINMENU statement of instance menus or set menus.

Submenus: BEGINSUBMENU, ENDSUBMENU

The BEGINSUBMENU and ENDSUBMENU statements create a submenu from the current menu.

For an example, refer to the Planned Order instance menu—contained in the Planned Order page code—for an example.

As with the BEGINMENU and ENDMENU statements, the BEGINSUBMENU and ENDSUBMENU statements can contain the following keywords:

- BEGINSUBMENU and ENDSUBMENU
- SEPARATOR
- MENUITEM

Menu Items: MENUITEM

The MENUITEM statement specifies a menu item that executes a user action when the user clicks it.



For examples, refer to Understanding the Unit Set Menu and Understanding the Unit Instance Menu.

The MENUITEM statement uses the LABEL property, which provides the label for this item in the menu. You can include a "&" in the label, which will allow an accelerator character for this menu item. For example, the following label allows a user to select the **&New** item from the menu with the **ALT+N** key:

```
LABEL "&New"
```

The MENUITEM statement has the ONPICK event, which executes a command when the user clicks this menu item. The ONPICK event uses the following syntax, where command is the command to be executed:

```
ONPICK { command };
```

SEPARATOR

The SEPARATOR statement causes a line to appear inside a menu or submenu. It can be used to separate menu items into logical groupings.



For examples, refer to Understanding the Unit Set Menu and Understanding the Unit Instance Menu.

Writing Statements That Create Pages

This section contains the statements that create either a form page or a property sheet page. A property sheet page is similar to a form page; a property sheet page is created with a class file. These statements create a blank page upon which a user can place controls.



For more information refer to Class Files Contain Property Sheet Pages and Menus.

Form Page: BEGINFORM, ENDFORM

BEGINFORM and ENDFORM creates a form page upon which you can place controls. All form pages use this.



For examples, refer to Understanding Examples Of Resource Language Code.

Form pages have four buttons: OK, Apply, Cancel, and Help.

The BEGINFORM and ENDFORM statements use the following syntax, required syntax is in bold:

```
BEGINFORM
```

```
  LABEL
```

```
  TOPICPOPULATE topic_attributes
```

```
  ONAPPLY
```

```
  ONCANCEL
```

```
  ONHELP
```

```
  ONOK
```

```

;

// control statements go here

ENDFORM;

```

The BEGINFORM control statement uses the following properties:

BEGINFORM, ENDFORM Properties

Property	Description
LABEL "label"	where label is the text string to show on the top margin of this property sheet page.
TOPICPOPULATE topic_attributes	Optional. topic_attributes are the attributes for TOPICPOPULATE. For more information, refer to TOPICPOPULATE in Writing Statement Properties.

The BEGINFORM control statement can use the following events:

BEGINFORM, ENDFORM Events

Event	Description
ONAPPLY { command }	Optional. command is the command(s) that you wish to perform when the user presses the page's Apply or OK button. This is done after the ONAPPLYs for all the controls on this page .
ONCANCEL {command}	Optional. command is the command(s) that you wish to perform when the user presses the page's CANCEL button.
ONHELP { command }	Optional. command is the command(s) that you wish to perform when the user presses the page's HELP button.
ONOK { command }	Optional. command is the command(s) that you wish to perform when the user presses the page's OK button.



Note: You cannot use the LOADFORM command with any of the events listed above.

Property Sheet Page: BEGINPROPSHEET, ENDPROPSHEET

BEGINPROPSHEET and ENDPROPSHEET creates a property sheet page upon which you can place controls.



For examples, refer to Understanding Examples Of Resource Language Code.

Property sheet pages have four buttons: OK, Apply, Cancel, and Help.

The BEGINPROPSHEET and ENDPROPSHEET statements use the following syntax, required syntax is in bold:

```
BEGINPROPSHEET

    LABEL

    TOPICPOPULATE topic_attributes

    ONAPPLY

    ONCANCEL

    ONHELP

    ONOK

;

// control statements go here

ENDPROPSHEET;
```

The BEGINPROPSHEET control statement uses the following properties:

BEGINPROPSHEET, ENDPROPSHEET Properties

<i>Property</i>	<i>Description</i>
LABEL "label"	where label is the text string to show on the top margin of this property sheet page.
TOPICPOPULATE topic_attributes	Optional. topic_attributes are the attributes for TOPICPOPULATE. Use this when the topic for the property sheet page is not passed in through a parameter. <hr/> For more information, refer to TOPICPOPULATE in Writing Statement Properties. <hr/>

The BEGINPROPSHEET control statement can use the following events:

BEGINPROPSHEET, ENDPROPSHEET Events

<i>Event</i>	<i>Description</i>
ONAPPLY { command }	Optional. command is the command(s) that you wish to perform when the user presses the property sheet page's Apply or OK button. This is done after the ONAPPLYs for all the controls on this property sheet page.
ONCANCEL {command}	Optional. command is the command(s) that you wish to perform when the user presses the property sheet page's CANCEL button.
ONHELP { command }	Optional. command is the command(s) that you wish to perform when the user presses the property sheet page's HELP button.
ONOK { command }	Optional. command is the command(s) that you wish to perform when the user presses the property sheet page's OK button.

Writing Control Statements That Contain Other Controls

This section describes the control statements whose keywords start with BEGIN and END (such as BEGINPAGE and ENDPAGE). These statements can contain other control statements. Control statements create controls, which display something on a page, such as a page or a grid.

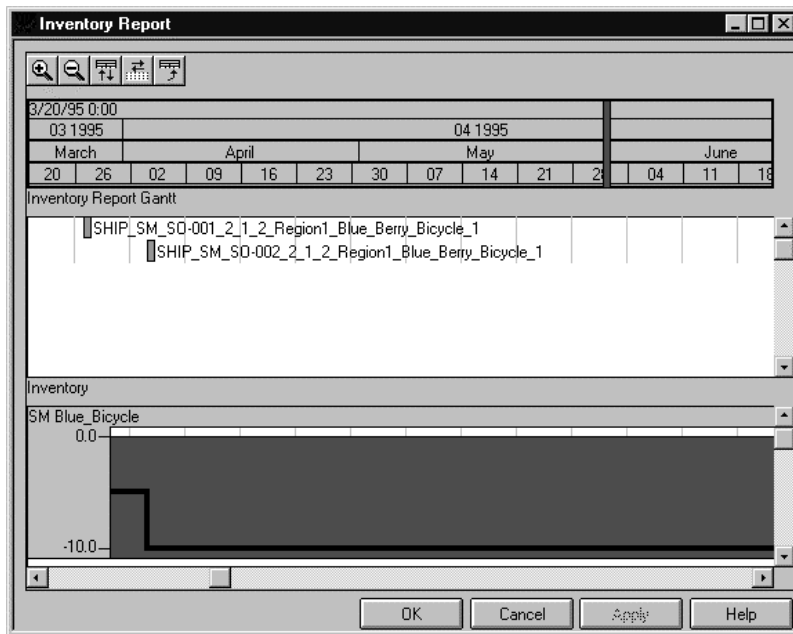
Coordinated Time Control Widget: BEGINCTC, ENDCTC

The CTC statement creates a coordinated time control widget (CTC). A CTC is meant to show histograms and/or tasks over a time period to show information about an object, such as an inventory item.



For an example, refer to Understanding the Inventory Report Page Code.

Here is a CTC that contains a TASK_GANTT and an INVENTORY_HISTOGRAM.



CTC control containing a TASK_GANTT and an INVENTORY_HISTOGRAM

A CTC must contain one or more of the following controls that show behavior over a period of time:

- CAPACITY_HISTOGRAM
- INVENTORY_HISTOGRAM
- CALENDAR_CONTROL
- RESOURCE_GANTT
- TASK_GANTT

The CTC control statement uses the following syntax, required syntax is in bold:

BEGINCTC

LOCATION left, top, length, height

TIME_FENCE time_fence_attributes

INITIAL_GRID_RESOLUTION "resolution"

LEFT_TIME

INITIAL_CALENDAR_SCALE "scale"

INITIAL_ZOOM "zoom"

TOPICPOPULATE topic_attributes

VALUEPOPULATE value_attributes

```
// The following control statements can go here: CAPACITY_HISTOGRAM,
// INVENTORY_HISTOGRAM, CALANDAR_CONTROL, RESOURCE_GANTT, TASK_GANTT.

;

ENDCTC;
```

The BEGINCTC control statement uses the following properties:

BEGINCTC, ENDCTC Properties

Property or Statement	Description
LOCATION left, top, length, height	where left, top, length, and height determine the location and size of this control on the page. <hr/> For more information, refer to Writing Statement Properties. <hr/>
TIME_FENCE time_fence_attributes	where time_fence_attributes are the attributes for TIME_FENCE. <hr/> For more information, refer to TIME_FENCE. <hr/>
INITIAL_GRID_RESOLUTION "resolution"	Optional. resolution is a number representing the resolution in number of hours for the shift length. The numbers are 2, 4, 6, 8, or 12 (2 hours, 4 hours, ...). 8 is the default.
LEFT_TIME	Not used.
INITIAL_CALENDAR_SCALE "scale"	Optional. scale is 0 for hours, 1 for shift, 2 for day, 3 for week, 4 for month, and 5 for quarter. The default is 3.
INITIAL_ZOOM "zoom"	Optional. zoom is a number representing the initial zoom that you want for this CTC. 0 is the maximum zoom in, 1 is zoom in, 2 is normal view, 3 is zoom out, and 4 is maximum zoom out. The default is 2.
TOPICPOPULATE topic_attributes	where topic_attributes are the attributes for TOPICPOPULATE. These attributes are usually TRANSACTION "get_oid main_environment;".

Property or Statement	Description
	<hr/> For more information, refer to TOPICPOPULATE in Writing Statement Properties. <hr/>
VALUEPOPULATE value_attributes	<p>where value_attributes are the attributes for VALUEPOPULATE. These attributes are usually:</p> <pre>TRANSACTION "get_project_times(:env_name \"main_environment\")".</pre> <hr/> For more information, refer to VALUEPOPULATE in Writing Statement Properties and get_project_times. <hr/>

The BEGINCTC control statement uses the following statements:

BEGINCTC, ENDCTC Statements

Statement	Description
CAPACITY_HISTOGRAM	<p>The CTC can use a CAPACITY_HISTOGRAM statement to place a capacity histogram in the CTC.</p> <hr/> For more information, refer to Capacity Histogram inside a CTC: CAPACITY_HISTOGRAM. <hr/>
INVENTORY_HISTOGRAM	<p>The CTC can use an INVENTORY_HISTOGRAM statement to place an inventory histogram in the CTC.</p> <hr/> For more information, refer to Inventory Histogram inside a CTC: INVENTORY_HISTOGRAM. <hr/>
CALENDAR_CONTROL	<p>The CTC can use a CALENDAR_CONTROL statement to place a calendar editor in the CTC.</p>

Statement	Description
	<hr/> For more information, refer to Calendar inside a CTC: CALENDAR_CONTROL. <hr/>
RESOURCE_GANTT	The CTC can use a RESOURCE_GANTT statement to place a resource Gantt in the CTC. <hr/> For more information, refer to Resource Gantt inside a CTC: RESOURCE_GANTT. <hr/>
TASK_GANTT	The CTC can use a TASK_GANTT statement to place a task Gantt in the CTC. <hr/> For more information, refer to Task Gantt inside a CTC: TASK_GANTT. <hr/>

TIME_FENCE

The TIME_FENCE control statement, used only with BEGINCTC-ENDCTC, allows new time fences to be added. If you add fences, the default early and late fences are not added, and you must declare them with TIME_FENCE. If you want to display time fences other than the default early and late fence, use TIME_FENCE to do so.

The TIME_FENCE control statement uses the following syntax, required syntax is in bold:

```

TIME_FENCE

    LABEL

    COLOR

    ONCHANGE

    VALUEPOPULATE value_attributes

;

```

The TIME_FENCE control statement uses the following properties:

TIME_FENCE Properties

Property or Statement	Description
LABEL "label"	where label is the text string to show when you place the cursor over the fence.
COLOR "color_index"	where color_index is one of the following integers to set the time fence color: 0 = green, 1 = yellow, 2 = cyan, 3 = gray, 4 = orange, 5 = red, 6 = magenta, 7 = blue, 8 = teal, 9 = money green, 10 = dark green, 11 = sky blue, 12 = brown, 13 = dark blue, 14 = purple, 15 = olive
VALUEPOPULATE value_attributes	where value_attributes are the attributes for VALUEPOPULATE. Use VALUEPOPULATE to get the osset of instances to populate the rows of the grid. <hr/> For more information, refer to VALUEPOPULATE in Writing Statement Properties. <hr/>

The TIME_FENCE control statement uses the following events:

TIME_FENCE Events

Event	Description
ONCHANGE { command }	Use ONCHANGE rather than ONAPPLY because time fences immediately send their transactions when they are moved. Moving a time fence does not light up the APPLY button. If you do not include ONCHANGE, the user will not be able to drag the time fence.

Here is an example of a definition for the early fence:

```
TIME_FENCE

  LABEL "Early Fence"

  VALUEPOPULATE PATH "early_fence"

  COLOR 6

  ONCHANGE {TRANSACTION "ui_time_change_early_fence(:data %[] .Value)"};
```

Here is an example of a definition for the late fence:

```
TIME_FENCE

LABEL "Late Fence"

VALUEPOPULATE PATH "late_fence"

COLOR 2

ONCHANGE {TRANSACTION "ui_time_change_late_fence(:data %[] .Value)"};
```

Grids: BEGINGRID, ENDGRID

The BEGINGRID and ENDGRID statements create a grid. A grid is used to contain rows and columns of information. A grid always contains an osset of instances. The osset of instances occupy the rows of a grid, while the slots for that instance occupy the columns.



For an example, refer to Understanding the Purchase Order Page Code.

Here is a grid.

	Line*	Item Unit*	Item*	UDM*	Qty Ordered*	Qty Released*	Unrel'd Qty*	Unit Price*
1	1	SM	Red_Paint	Each	5.0	1.0	4.0	0.0
2	2	SM	Blue_Paint	Each	5.0	1.0	4.0	0.0
3	3	SM	Green_Paint	Each	5.0	1.0	4.0	0.0

Grid control

Notice that the BEGINGRID and ENDGRID statements use COLUMN statements to create the columns within the grid.

The BEGINGRID and ENDGRID control statements use the following syntax, required syntax is in bold:

```
BEGINGRID

NAME "name"

LOCATION left, top, length, height

NOEDIT

NOAPPLY
```

```

NOAUTOPOPULATE

DISABLED

LOCKCOLUMNS "number_columns"

TOPICPOPULATE topic_attributes

VALUEPOPULATE value_attributes

COLUMN // one or more COLUMN statements go here.

ONCHANGE { command }

ONREFRESH { command }

;

ENDGRID;

```

The BEGINGRID control statement uses the following properties:

BEGINGRID, ENDGRID Properties

<i>Property</i>	<i>Description</i>
NAME "name"	Optional. name is a unique name for this control. This allows other controls to reference this control, such as in another control's TOPICPOPULATE property.
LOCATION left, top, length, height	<p>where left, top, length, and height determine the location and size of this control on the page.</p> <hr/> <p>For more information, refer to LOCATION in Writing Statement Properties.</p> <hr/>
NOAPPLY	Optional. If the user makes changes to the value shown in this control's field, the Apply button on the page is not enabled.
NOAUTOPOPULATE	<p>Optional. When this control is first displayed, this control will not be automatically populated, leaving the grid blank.</p> <hr/> <p>For more information, refer to NOAUTOPOPULATE in Writing</p>

Property	Description
	Statement Properties.
NOEDIT	Optional. This property prevents the user from changing the data shown in the grid.
DISABLED	Optional. Causes the initial state of this control to be disabled.
LOCKCOLUMNS "number_columns"	Optional. Prevents the user from scrolling the leftmost columns in the grid. number_columns is the number of columns, starting from the left side of the grid, that are not to be scrolled when the user scrolls horizontally.
TOPICPOPULATE topic_attributes	Optional. topic_attributes are the attributes for TOPICPOPULATE. Use TOPICPOPULATE to get the topic object for the grid. For more information, refer to TOPICPOPULATE in Writing Statement Properties.
VALUEPOPULATE value_attributes	where value_attributes are the attributes for VALUEPOPULATE. Use VALUEPOPULATE to get the osset of instances to populate the rows of the grid. A common transaction to use with VALUEPOPULATE is form_sort_slot_instance_list. For more information, refer to VALUEPOPULATE in Writing Statement Properties.

The grid uses COLUMN statements to place columns in the control.



For more information, refer to Columns in a Grid or Spreadsheet: COLUMN.

The BEGINGRID control statement can use the following events:

BEGINGRID, ENDGRID Events

<i>Event</i>	<i>Description</i>
ONCHANGE { command }	Optional. command is the command(s) that you wish to perform when the user changes the values in the grid. For example, you can disable buttons.
ONREFRESH { command }	Optional. command is the command(s) that you wish to perform when the values in this control are refreshed.

When you use the NAME property with this control, another control can use the Disable, Enable, Hide, Show, Refresh, Topic, SelectedRow, RowColumnValue, and ValueAtRow methods on this grid. For example, an ONPRESS event in a PUSHBUTTON control statement can use the Hide method to hide this grid.



For more information, refer to Writing Methods.

The SelectedRow method returns the instance of the currently selected row in the grid.



For an example, refer to Understanding the Build Schedule (with Resource) Spreadsheet Page Code.

The ValueAtRowColumn topic returns the value of one field in the grid. It uses the following syntax, where RowNumber is the number of the row and ColumnNumber is the number of the column where that field is located:

```
%[control].ValueAtRowColumn (RowNumber, ColumnNumber)
```

The ValueAtRow topic returns the value of a row, which is of type RPS_INSTANCE. It uses the following syntax, where RowNumber is the number of the row whose value is returned:

```
%[control].ValueAtRow (RowNumber)
```

Pages or Components: BEGINPAGE, ENDPAGE

BEGINPAGE and ENDPAGE create a tabbed page upon which you can place controls. This changes your page into a component, with each page being a page in the component. You can place several pages on a page, and display each page by clicking the tab for that page. Control

statements for controls on a page can reference controls on the base page, but they cannot reference controls on other pages.



For an example, refer to Understanding the Build Schedule (with Resource) Spreadsheet Page Code.

Here is a page. The page is the Key Setup page; if you wanted to select another page, such as the Spreadsheet page, you would click that page's tab.

Page control

The BEGINPAGE and ENDPAGE statements use the following syntax:

```
BEGINPAGE

    LABEL

    TOPICPOPULATE topic_attributes

;

// other control statements go here

ENDPAGE;
```

The PAGE control statement uses the following properties:

BEGINPAGE, ENDPAGE Properties

<i>Property</i>	<i>Description</i>
LABEL "label"	where label is the text string to show on the tab for this page.
TOPICPOPULATE topic_attributes	where topic_attributes are the attributes for TOPICPOPULATE.

Property	Description
	For more information, refer to TOPICPOPULATE in Writing Statement Properties.

Spreadsheets: BEGINSPREADSHEET, ENDSPREADSHEET

The BEGINSPREADSHEET and ENDSPREADSHEET statements create a spreadsheet. A spreadsheet is used to display a spreadsheet object, which was created to contain rows. Each row is an instance. Each row contains key information, such as a unit name or an inventory item name, and quantities over a set of dates.



For an example, refer to Understanding the Build Schedule (with Resource) Spreadsheet Page Code.

A spreadsheet contains two sets of columns. One set is created with COLUMN statements; these columns are used to display the key information. This information is usually locked information (the user can not scroll that information), such as a unit name or an inventory item name. The other set of columns is automatically created and is used to display the quantities; the user can scroll this information.

	Unit	Inventory Item	Resource	Row Type	Past Due	3/1/95	3/2/95	3/3/95	3/4/95
1	SM	Blue_Bicycle	Assemble_Work_Cent	Production Supply	0.00	0.00	0.00	0.00	0.00
2	SM	Blue_Bicycle	Assemble_Work_Cent	Production Order Supp	0.00	0.00	0.00	0.00	0.00
3	SM	Blue_Bicycle	Assemble_Work_Cent	Non Production Order	0.00	0.00	0.00	0.00	0.00
4	SM	Blue_Bicycle	Assemble_Work_Cent	Production Supply Star	0.00	0.00	0.00	0.00	0.00
5	SM	Blue_Frame	Paint_Work_Center	Production Supply	0.00	0.00	0.00	0.00	0.00
6	SM	Blue_Frame	Paint_Work_Center	Production Order Supp	0.00	0.00	0.00	0.00	0.00
7	SM	Blue_Frame	Paint_Work_Center	Non Production Order	0.00	0.00	0.00	0.00	0.00
8	SM	Blue_Frame	Paint_Work_Center	Production Supply Star	0.00	0.00	0.00	0.00	0.00
9	SM	Blue_Frame	Weld_Work_Center	Production Supply	0.00	0.00	0.00	0.00	0.00
10	SM	Blue_Frame	Weld_Work_Center	Production Order Supp	0.00	0.00	0.00	0.00	0.00

Spreadsheet control

The BEGINSPREADSHEET and ENDSPREADSHEET control statements use the following syntax, required syntax is in bold:

BEGINSPREADSHEET

NAME "name"

LOCATION left, top, length, height

NOAPPLY

NOAUTOPOPULATE

NOEDIT

```

DISABLED

PREAPPLY { command }

POSTAPPLY { command }

LOCKCOLUMNS "number_columns"

VARIABLECOLUMNFORMAT "format"

VARIABLECOLUMNWIDTH "width"

TOPICPOPULATE topic_attributes

VALUEPOPULATE value_attributes

COLUMN // one or more COLUMN statements go here.

ONCHANGE { command }

ONREFRESH { command }

;

ENDSPREADSHEET;

```

The **BEGINSPREADSHEET** control statement uses the following properties:

BEGINSPREADSHEET, ENDSPREADSHEET Properties

<i>Property</i>	<i>Description</i>
NAME "name"	Optional. name is a unique name for this control. This allows other controls to reference this control, such as in a button's ONPRESS event.
LOCATION left, top, length, height	<p>where left, top, length, and height determine the location and size of this spreadsheet.</p> <hr/> <p>For more information, refer to LOCATION in Writing Statement Properties.</p> <hr/>
NOAPPLY	Optional. If the user makes changes to the value shown in this control's field, the Apply button on the page is not enabled.
NOAUTOPOPULATE	Optional. When this control is first displayed, this control will not be automatically populated, leaving the

Property	Description
	spreadsheet blank. <hr/> For more information, refer to NOAUTOPOPULATE in Writing Statement Properties. <hr/>
NOEDIT	Optional. This property prevents the user from changing the data shown in the spreadsheet.
LOCKCOLUMNS "number_columns"	Prevents the user from scrolling the leftmost columns in the spreadsheet. number_columns is the number of columns, starting from the left side of the spreadsheet, that are not to be scrolled when the user scrolls horizontally.
VARIABLECOLUMNFORMAT "format"	Optional. Sets the format for the set of columns that displays quantities, such as the production supply over a set of dates. This data is always floating point data. format is the format used to display data in this field. The digits in format are: # display a digit except for leading zeros 0 display a digit, and display trailing zeros . display the decimal point Example: VARIABLECOLUMNFORMAT "###,##0.00" means display a number of up to eight-digits with two decimal places.
VARIABLECOLUMNWIDTH "width"	Optional. width is the number of units of width for the quantities set of columns. To calculate the units for minimum column width, multiply the number of characters you wish to display by 4.5.
TOPICPOPULATE topic_attributes	Optional. topic_attributes are the attributes for TOPICPOPULATE. Use TOPICPOPULATE to get the topic object if it has not been supplied. For example, the page may not supply the topic for this spreadsheet. <hr/> For more information, refer to

Property	Description
	TOPICPOPULATE in Writing Statement Properties.
VALUEPOPULATE value_attributes	<p>where value_attributes are the attributes for VALUEPOPULATE. Use VALUEPOPULATE to get the oset of instances to populate the rows of the grid.</p> <p>For more information, refer to VALUEPOPULATE in Writing Statement Properties.</p>

The spreadsheet uses COLUMN statements to place columns in the control.



For more information, refer to Columns in a Grid or Spreadsheet: COLUMN.

The BEGINSPREADSHEET control statement uses the following events:

BEGINSPREADSHEET, ENDSPREADSHEET Events

Event	Description
ONCHANGE {command }	Optional. command is the command(s) that you wish to perform when the user changes the values in the spreadsheet. For example, you can disable buttons.
ONREFRESH {command}	Optional. command is the command(s) that you wish to perform when the values in this control are refreshed.
PREAPPLY { command }	Optional. This event prevents a spreadsheet from doing excessive visual updates when several values in the spreadsheet have been changed. Used with POSTAPPLY.
POSTAPPLY { command }	Optional. This event prevents a spreadsheet from doing excessive visual updates when several values in the spreadsheet have been changed. Used with PREAPPLY.

When you use the NAME property with this control, another control can use the Disable, Enable, Hide, Show, Refresh, and Topic methods on this spreadsheet. For example, an ONPRESS event in a PUSHBUTTON control statement can use the Refresh method to refresh this spreadsheet.



For more information, refer to Writing Methods.

Here are examples of the PREAPPLY and POSTAPPLY events.

```
BEGINSPREADSHEET

    TOPICCLASS "Spreadsheet_Row"

    VARIABLECOLUMNFORMAT "###,##0.00"

    LOCATION 5, 25, 510, 120

    PREAPPLY{TRANSACTION "ui_modify_spreadsheet_generate_detailed(:spreadsheet
%[] .Topic

        :generate_detailed 0 :generate_spreadsheet 0)"}

    POSTAPPLY{TRANSACTION "ui_modify_spreadsheet_generate_detailed(:spreadsheet
%[] .Topic

        :generate_detailed 1 :generate_spreadsheet 1)"}

    VALUEPOPULATE PATH "aggregate_spreadsheet_rows";
```

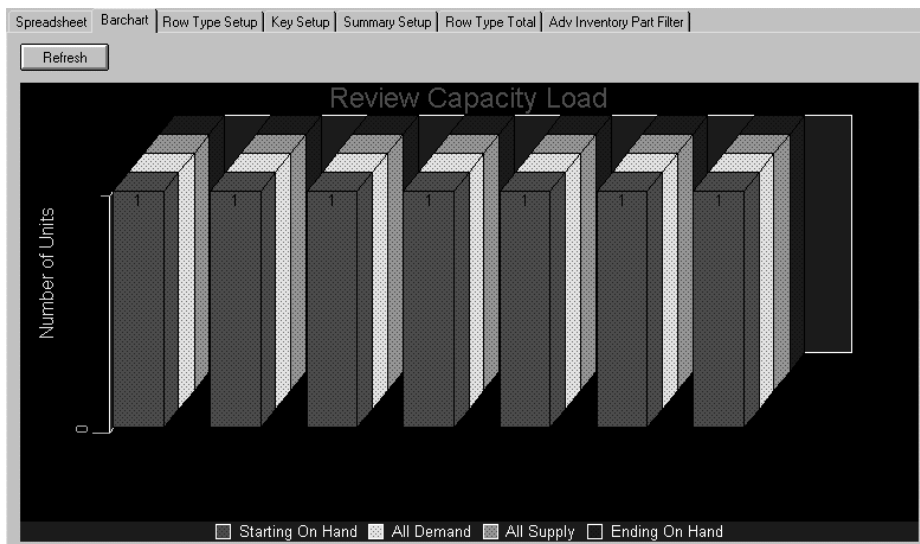
Writing Control Statements

This section discusses the rest of the control statements. Control statements create controls, which display something on a page, such as a PUSHBUTTON, an EDIT field, or a BITMAP. A user can interact with some of these controls, such as pressing a PUSHBUTTON or entering data in an EDIT field.

BARChart

The BARChart control creates a bar chart upon a page.

Here is a barchart.



Barchart

The BARCHART control statement uses the following syntax, required syntax is in bold:

BARCHART

LOCATION left, top, length, height

TOPICPOPULATE topic_attributes

VALUEPOPULATE TRANSACTION "value_foo() "

;

The BARCHART control statement uses the following properties:

BARCHART Properties

<i>Property</i>	<i>Description</i>
LOCATION left, top, length, height	where left, top, length, and height determine the location and size of this control on the page. For more information, refer to LOCATION in Writing Statement Properties.
TOPICPOPULATE topic_attributes	Optional. topic_attributes are the attributes for TOPICPOPULATE. Use TOPICPOPULATE to get the Topic object if it has not been supplied. The topic could be a query instance. <hr/> For more information, refer to

Property	Description
	TOPICPOPULATE in Writing Statement Properties.
VALUEPOPULATE TRANSACTION	The valuepopulate transaction must have certain output variables defined. Refer to the listing below for action sample_barchart_value_populate.

The value populate transaction must have the following output variables defined:

```

action sample_barchart_value_populate

  /* REQUIRED OUTPUT VARIABLES */

  output: int          num_columns, // number of bar columns

  output: oset[int]    bar_values,  // bar values, should be a
                                   // multiple of 'num_columns'

  output: oset[string] bar_labels,  // the label for each bar

  output: oset[int]    bar_colors,  // the color index for each bar

  // 0 = green, 1 = yellow, 2 = cyan, 3 = gray,

  // 4 = orange, 5 = red, 6 = magenta,

  // 7 = blue, 8 = teal, 9 = money green,

  // 10 = dark green, 11 = sky blue, 12 = brown,

  // 13 = dark blue, 14 = purple, 15 = olive


  output: oset[string] bar_legend,  // the bar legend text

  output: string      main_title,

  output: string      sub_title,

  output: string      x_title,

  output: string      y_title,

  output: int         show_y_units, // 0 = don't show, non-zero = show

  /* OPTIONAL OUTPUT VARIABLES */

  output: int         horz_angle, // default: 20

  output: int         vert_angle, // default: 25

```

```

        output: int          bar_placement, // The bar placement has to do with

        // how bars are placed when there

        // are more than one bar per column.

        // default: 0

        // 0 = stacked

        // 1 = behind

        // 2 = beside

    no_context:)

{

// Put the action code here.

}

```

The following is a code example of how to put a Barchart into the Horizontal MRP spreadsheet. It shows the top of the Horizontal_MRP_Spreadsheet.000 file, and then shows code you can add to create a barchart.

```

[PROPERTYSHEET]

BEGINPROPSHEET

    LABEL "Horizontal MRP"

    TOPICPOPULATE TRANSACTION
    "transaction_get_spreadsheet_form_topic(:spreadsheet_template_name
    \"horizontal_mrp_spreadsheet\")"

    ONCLOSE TRANSACTION "transaction_release_spreadsheet_form_topic(:spreadsheet
    %[] .FormTopic)";

// Here is where the Horizontal_MRP_Spreadsheet.000 file has code to define
// controls to show the start date, number of periods, period duration, and a
// small grid to show Unit and Inventory Item. After these definitions, Page
// controls are defined. You can add the following code to have a Barchart
// page on this page.

BEGINPAGE LABEL "Barchart";

    BARCHART

```

```
NAME "Barchart"

LOCATION 5, 25, 510, 240

VALUEPOPULATE TRANSACTION
"ui_generate_barchart_from_spreadsheet (:spreadsheet %[].Topic)";
```

```
PUSHBUTTON

LABEL "Refresh"

LOCATION 5, 5, 50, 14

ONPRESS {METHOD %[Barchart].Refresh};
```

```
ENDPAGE;
```

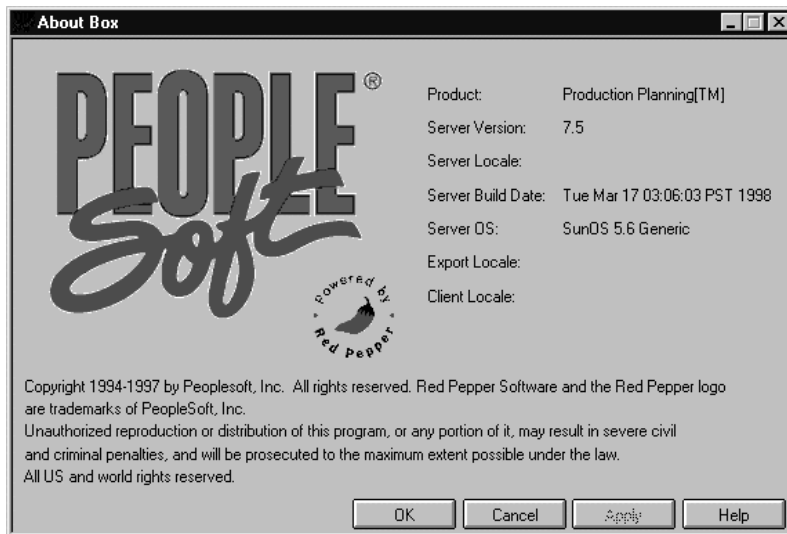
BITMAP

The BITMAP control statement places a bitmap upon a page. A bitmap is a graphic image.



For an example, refer to Understanding the About Box Page Code.

Here is a page with a bitmap placed on it. The bitmap is a PeopleSoft logo.



Bitmap in a page

The BITMAP control statement uses the following syntax, required syntax is in bold:

```

BITMAP

    LOCATION left, top, length, height

    FILE "file_name"

    STRETCH

;
  
```

The BITMAP control statement uses the following properties:

BITMAP Properties

<i>Property</i>	<i>Description</i>
LOCATION left, top, length, height	<p>where left, top, length, and height determine the location and size of this control on the page.</p> <hr/> <p>For more information, refer to LOCATION in Writing Statement Properties.</p> <hr/>
FILE "file_name"	where file_name is the name of the file that contains the bitmap.
STRETCH	Optional. This property causes the bitmap to be sized to fill the rectangle expressed in the location property. If

Property	Description
	STRETCH is not used, the image is placed at the left and top of the LOCATION property, and it is shown at its original size.

When you use the NAME property with this control, you can use the Hide and Show methods. For example, an event in another control can use the Hide method to hide this control.

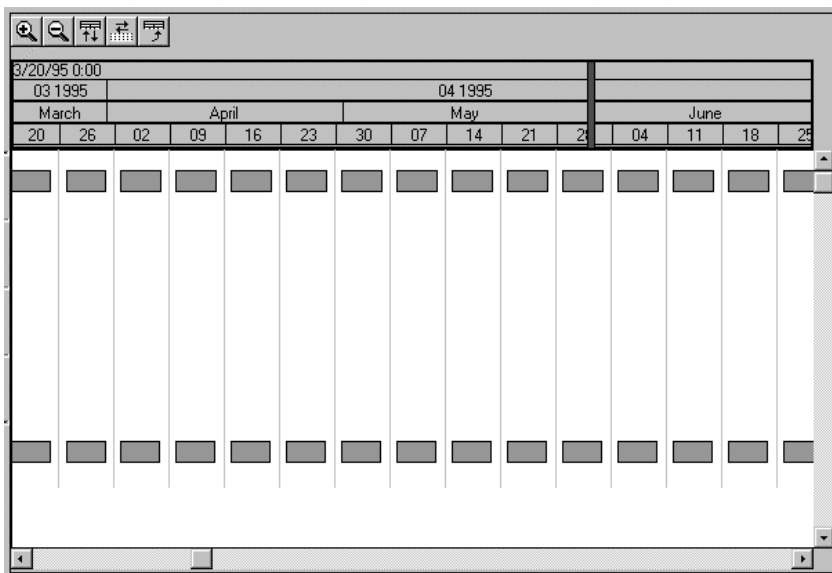


For more information, refer to Writing Methods.

Calendar inside a CTC: CALENDAR_CONTROL

The CALENDAR_CONTROL control statement places a calendar editor into a CTC. Refer to the Calendar resource file for an example.

Here is a calendar inside a CTC.



Calendar control

The CALENDAR_CONTROL statement uses the following syntax, required syntax is in bold:

CALENDAR_CONTROL

LABEL

LABELHEIGHT

PERCENTHEIGHT

;

The CALENDAR_CONTROL control statement uses the following properties:

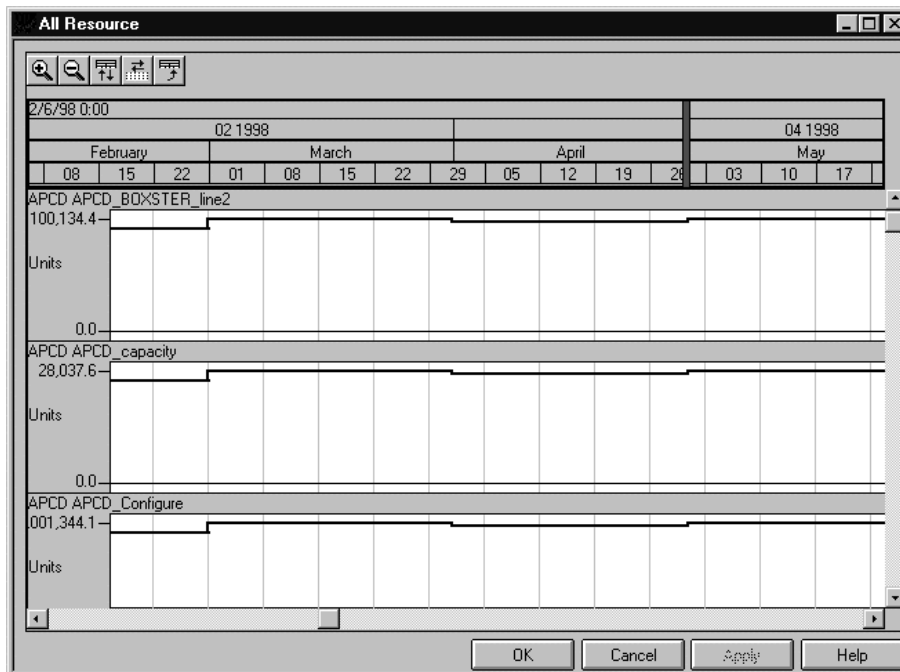
CALENDAR_CONTROL Properties

<i>Property</i>	<i>Description</i>
LABEL "label"	where label is the text string to show on the top of this control.
LABELHEIGHT height	where height is the number of units of height for the label. 12 is a common height.
PERCENTHEIGHT percent	where percent is a number showing the percentage of the CTC that you want this control to cover. For example, if you want the control to cover all the CTC, use 100; if you want it to cover half, use 50.

Capacity Histogram inside a CTC: CAPACITY_HISTOGRAM

The CAPACITY_HISTOGRAM control statement places a capacity histogram into a CTC. For an example, refer to the resource file AllEquipmentHistogram.

Here is a capacity histogram inside a CTC.



Capacity Histogram control

The CAPACITY_HISTOGRAM statement uses the following syntax, required syntax is in bold:

```

CAPACITY_HISTOGRAM

  LABEL "label"

  LABELHEIGHT height

  PERCENTHEIGHT percent

  TOPICPOPULATE topic_attributes

  VALUEPOPULATE value_attributes

  ;
  
```

The CAPACITY_HISTOGRAM control statement uses the following properties:

CAPACITY_HISTOGRAM Properties

Property	Description
LABEL "label"	where label is the text string to show on the top of this control.
LABELHEIGHT height	where height is the number of units of height for the label. 12 is a common height.
PERCENTHEIGHT percent	where percent is a number showing the

Property	Description
	percentage of the CTC that you want this control to cover. For example, if you want the control to cover all the CTC, use 100; if you want it to cover half, use 50.
TOPICPOPULATE topic_attributes	<p>Optional. topic_attributes are the attributes for TOPICPOPULATE. Use TOPICPOPULATE to get the Topic object if it has not been supplied. The topic could be a query instance.</p> <hr/> <p>For more information, refer to TOPICPOPULATE in Writing Statement Properties.</p> <hr/>
VALUEPOPULATE value_attributes	<p>where value_attributes are the attributes for VALUEPOPULATE. The value returned by the value_attributes should be an instance of equipment resource, or an oset of such instances.</p> <hr/> <p>For more information, refer to VALUEPOPULATE in Writing Statement Properties.</p> <hr/>

CHECKBOX

The CHECKBOX control statement places a check box on the page. The user can click in the check box and change a Boolean value.



For an example, refer to Understanding the Build Schedule (with Resource) Spreadsheet Page Code.

Here are two check boxes, labeled Sorting Enabled and Ascending Sort.

Check box control

The CHECKBOX control statement uses the following syntax, required syntax is in bold:

CHECKBOX

NAME "name"

LABEL "label"

LOCATION left, top, length, height

TYPE "datatype"

NOAPPLY

NOAUTOPOPULATE

NOEDIT

REQUIRED

DISABLED

TOPICPOPULATE topic_attributes

VALUEPOPULATE value_attributes

ONAPPLY { command }

ONCHANGE { command }

;

The CHECKBOX control statement uses the following properties:

CHECKBOX Properties

<i>Property</i>	<i>Description</i>
NAME "name"	Optional. name is a unique name for this control. This allows other controls to reference this control, such as in another

Property	Description
	control's events.
LABEL "label"	Optional. label is the text string to show in a Required error message box.
LOCATION left, top, length, height	<p>where left, top, length, and height determine the location and size of this control on the page.</p> <hr/> <p>For more information, refer to LOCATION in Writing Statement Properties.</p> <hr/>
TYPE "data_type"	where data_type is RPS_INT. A check box always uses an integer.
NOAPPLY	Optional. If the user makes changes to the value shown in this control's field, the Apply button on the page is not enabled.
NOAUTOPOPULATE	<p>Optional. When this control is first displayed, the field will not be automatically populated by a value.</p> <hr/> <p>For more information, refer to NOAUTOPOPULATE in Writing Statement Properties.</p> <hr/>
REQUIRED	Optional. This property prevents the user from leaving the page without entering data in this control. A Required error message box appears if the user tries to leave the page without clicking this check box.
DISABLED	Optional. Causes the initial state of this control to be disabled.
TOPICPOPULATE topic_attributes	<p>Optional. topic_attributes are the attributes for TOPICPOPULATE. Use TOPICPOPULATE to get the Topic object if it has not been supplied.</p> <hr/> <p>For more information, refer to TOPICPOPULATE in Writing Statement Properties.</p> <hr/>

Property	Description
VALUEPOPULATE value_attributes	<p>where value_attributes are the attributes for VALUEPOPULATE. Use VALUEPOPULATE to get the value of a Boolean integer that determines whether or not a check appears in the CHECKBOX field.</p> <hr/> <p>For more information, refer to VALUEPOPULATE in Writing Statement Properties.</p> <hr/>

The CHECKBOX control statement can use the following events:

CHECKBOX Events

Event	Description
ONAPPLY { command }	Optional. command is the command(s) that you wish to perform when the user presses the Apply or OK button and the value in the CHECKBOX is changed. For example, you can have a transaction set the slot on the server that populated this CHECKBOX field to the value set by the user.
ONCHANGE {command}	Optional. command is the command(s) that you wish to perform when the user changes the values in the CHECKBOX field. For example, you can disable buttons.

When you use the NAME property with this control, you can use the Disable, Enable, Hide, Show, Refresh, Value, and Topic methods. For example, an event in another control can use the Hide method to hide this control.



For more information, refer to Writing Methods.

Columns in a Grid or Spreadsheet: COLUMN

The COLUMN control statement places a column in a grid or a spreadsheet.



For an example, refer to Understanding the Build Schedule (with Resource) Spreadsheet Page Code and Understanding the Purchase Order Page Code.

Here are several columns inside a grid.

	Line*	Item Unit*	Item*	UOM*	Qty Ordered*	Qty Released*	Unrel'd Qty*	Unit Price*
1	1	SM	Red_Paint	Each	5.0	1.0	4.0	0.0
2	2	SM	Blue_Paint	Each	5.0	1.0	4.0	0.0
3	3	SM	Green_Paint	Each	5.0	1.0	4.0	0.0

Columns inside a grid

The COLUMN statement uses the following syntax, required syntax is in bold:

```

COLUMN

    NAME "name"

    LABEL "label"

    WIDTH "width"

    TYPE "data_type"

    NOEDIT

    LEFT/CENTER/RIGHT

    FORMAT "format"

    SORT "sort_value"

    WIDGET "widget"

    MENUPOPULATE menupopulate_attributes

    VALUEPOPULATE valuepopulate_attribute

    ONAPPLY { command }

;

```

The COLUMN control statement uses the following properties:

COLUMN Properties

<i>Property</i>	<i>Description</i>
NAME "name"	Optional. name is a unique name for this control. This allows other controls to reference this control, such as in a button's ONPRESS event.
LABEL "label"	where label is the text string to show at the top of the column.
WIDTH "width"	where width is the number of units of width for the column. To calculate the units for minimum column width, multiply the number of characters you wish to display by 4.5.
TYPE "data_type"	<p>where data_type is the type of data that you want to display.</p> <hr/> <p>For more information, refer to TYPE in Writing Statement Properties.</p> <hr/>
NOEDIT	Optional. This property prevents the user from changing the data shown in the column.
LEFT/CENTER/RIGHT	Optional. LEFT left-justifies the data in this column; CENTER centers it; RIGHT right-justifies it.
FORMAT "format"	<p>Optional. Used when the TYPE is RPS_FLOAT. format is the format used to display data in this field. The digits in format are:</p> <p># display a digit except for leading zeros</p> <p>0 display a digit, and display trailing zeros</p> <p>. display the decimal point</p> <p>Example: "###,##0.00" means display a number of up to eight-digits with two decimal places.</p>
SORT "slot"	The SORT property defines that this column sorts by using the value of the slot on the instance of the row. For example, SORT "site.display_name" will sort the column on the site (unit) display name slot (you need to have populated the

Property	Description
	SPREADSHEET or GRID so that you can access site.display_name).
WIDGET "widget"	<p>where widget is the control statement that you want inside this column. You can use "CHECKBOX", "EDIT", or "COMBOBOX". For example, "CHECKBOX" would place a check box inside this column, and you can then populate and use that check box as you would a CHECKBOX control statement.</p> <hr/> <p>For an example, refer to Understanding the Build Schedule (with Resource) Spreadsheet Page Code.</p> <hr/>
MENUPOPULATE menu_attributes	<p>where menu_attributes are the attributes for MENUPOPULATE. Use this only when the column uses the WIDGET "COMBOBOX" property.</p> <hr/> <p>For more information, refer to MENUPOPULATE in Writing Statement Properties.</p> <hr/>
VALUEPOPULATE value_attributes	<p>where value_attributes are the attributes for VALUEPOPULATE. Use VALUEPOPULATE to get a value to display in this column.</p> <hr/> <p>For more information, refer to VALUEPOPULATE in Writing Statement Properties.</p> <hr/>

The COLUMN control statement uses the following event:

```
ONAPPLY { command }
```

where command is the command(s) that you wish to perform when the user presses the Apply button and the value in the COLUMN is changed. For example, you can have a transaction set the slot on the server that populated this COLUMN to the value set by the user.

Drop-down List: COMBOBOX

The COMBOBOX control statement creates a field in which the user can select from a drop-down list of items.



For an example, refer to Understanding the Purchase Order Page Code.

Note: The COMBOBOX cannot be the last control on a page.

Here is a combobox, closed and opened.

Combobox Control Closed

Combobox Control Opened

The COMBOBOX control statement uses the following syntax, required syntax is in bold:

COMBOBOX

NAME "name"

LABEL "label"

LOCATION left, top, length, height

TYPE "data_type"

NOAPPLY

```

NOAUTOPOPULATE

NOEDIT

REQUIRED

DISABLED

LEFT/RIGHT

MENUPOPULATE menupopulate_parameters

VALUEPOPULATE value_attributes

ONAPPLY { command }

ONCHANGE { command }

;

```

The COMBOBOX control statement uses the following properties:

COMBOBOX Properties

<i>Property</i>	<i>Description</i>
NAME "name"	Optional. name is a unique name for this control. This allows other controls to reference this control, such as in another control's TOPICPOPULATE property.
LABEL "label"	where label is the text string to show in a Required error message box.
LOCATION left, top, length, height	<p>where left, top, length, and height determine the location and size of this control on the page. Note: Height must be at least 24.</p> <hr/> <p>For more information, refer to LOCATION in Writing Statement Properties.</p> <hr/>
TYPE "data_type"	<p>where data_type is RPS_CLASS, RPS_ENUM, or RPS_INSTANCE.</p> <hr/> <p>For more information, refer to TYPE in Writing Statement Properties.</p> <hr/>
NOAPPLY	Optional. If the user makes changes to the value shown in this control's field, the

Property	Description
	Apply button on the page is not enabled.
NOAUTOPOPULATE	<p>Optional. When this control is first displayed, this control will not be automatically populated, leaving the field blank.</p> <hr/> <p>For more information, refer to NOAUTOPOPULATE in Writing Statement Properties.</p> <hr/>
NOEDIT	Optional. This property prevents the user from changing the value shown in the COMBOBOX field.
REQUIRED	Optional. This property prevents the user from leaving the page without selecting an item for this COMBOBOX menu. A Required error message box appears if the user tries to leave the page without selecting an item.
DISABLED	Optional. Causes the initial state of this control to be disabled.
LEFT/RIGHT	Optional. LEFT left-justifies the data in this field; RIGHT right-justifies it.
MENUPOPULATE menu_attributes	<p>where menu_attributes are the attributes for MENUPOPULATE. Use MENUPOPULATE to get the oset of values to display in the COMBOBOX list.</p> <hr/> <p>For more information, refer to MENUPOPULATE in Writing Statement Properties.</p> <hr/>
VALUEPOPULATE value_attributes	<p>Optional. value_attributes are the attributes for VALUEPOPULATE. Use VALUEPOPULATE to get a value to display in the COMBOBOX field before the user clicks the COMBOBOX; this allows a value to be displayed in the closed COMBOBOX before MENUPOPULATE populates it.</p> <hr/> <p>For an example, refer to Understanding the Purchase Order Page Code. For an</p>

Property	Description
	<p>example of VALUEPOPULATE not being used to display a value in the closed COMBOBOX, refer to Understanding the Add Unit Page Code.</p> <p>For more information, refer to VALUEPOPULATE in Writing Statement Properties.</p>

The COMBOBOX control statement can use the following events:

COMBOBOX Events

Events	Description
ONAPPLY { command }	Optional. command is the command(s) that you wish to perform when the user presses the Apply or OK button and the value in the COMBOBOX is changed. For example, you can have a transaction set the slot on the server that populated this COMBOBOX field to the value set by the user.
ONCHANGE {command}	Optional. command is the command(s) that you wish to perform when the user changes the values in the COMBOBOX field. For example, you can disable buttons.

When you use the NAME property with this control, you can use the Disable, Enable, Hide, Show, Refresh, Value, and Topic methods. For example, an event in another control can use the Hide method to hide this control.



For more information, refer to Writing Methods.

CONTROLSLIDER

The CONTROLSLIDER control statement places a control slider upon a page. A control slider displays a slider control and a field; the user can change the value displayed in the control slider field by adjusting the slider control.



For an example, refer to Understanding the Control panel Code.

Here is a controlslider.



Control Slider control

The CONTROLSLIDER control statement uses the following syntax, required syntax is in bold:

```
CONTROLSLIDER

  NAME "name"

  LOCATION left, top, length, height

  TYPE "datatype"

  NOAPPLY

  NOAUTOPOPULATE

  TOPICPOPULATE topic_attributes

  VALUEPOPULATE value_attributes

  ONAPPLY { command }

  ONCHANGE { command }

;
```

The CONTROLSLIDER control statement uses the following properties:

CONTROLSLIDER Properties

<i>Property</i>	<i>Description</i>
NAME "name"	Optional. name is a unique name for this control. This allows other controls to reference this control, such as in another control's TOPICPOPULATE property.
LOCATION left, top, length, height	where left, top, length, and height determine the location and size of this control on the page. <hr/> For more information, refer to LOCATION in Writing Statement

Property	Description
	Properties.
TYPE "datatype"	where datatype is RPS_FLOAT.
NOAPPLY	Optional. When the user makes changes to the value shown in this control's field, the Apply button on the page is not enabled.
NOAUTOPOPULATE	Optional. When this control is first displayed, this control will not be automatically populated, leaving the field blank. For more information, refer to NOAUTOPOPULATE in Writing Statement Properties.
TOPICPOPULATE topic_attributes	Optional. topic_attributes are the attributes for TOPICPOPULATE. Use TOPICPOPULATE to get the topic object for the CONTROLSLIDER control statement. For more information, refer to TOPICPOPULATE in Writing Statement Properties.
VALUEPOPULATE value_attributes	where value_attributes are the attributes for VALUEPOPULATE. Use VALUEPOPULATE to get the value to display in the CONTROLSLIDER field. A value of 0 shows the control slider to be OFF. For more information, refer to VALUEPOPULATE in Writing Statement Properties.

The CONTROLSLIDER control statement uses the following events:

CONTROLSLIDER Events

Event	Description
ONAPPLY { command }	Optional. command is the command(s) that you wish to perform when the user presses the Apply or OK button. For example, you can have a transaction set the slot on the server that populated this CONTROLSLIDER to the value set by the user.
ONCHANGE {command}	Optional. command is the command(s) that you wish to perform when the user changes the values in the CONTROLSLIDER field. For example, you can disable buttons.

When you use the NAME property with this control, you can use the Hide, Show, Refresh, Value, and Topic methods. For example, an event in another control can use the Refresh method to refresh the value shown in this control.



For more information, refer to Writing Methods.

Field to Enter Data: EDIT

The EDIT control statement creates a field in which the user can enter new data. This field can also be populated with data from the server, allowing the user to display and modify that data.



For an example, refer to Understanding the Add Unit Page Code.

Here are two edit controls on the Unit (site) page, labeled Unit: and Description:.

The screenshot shows a dialog box titled "Unit". It has a "General" tab. Inside the tab, there are three input fields: "Unit:" with the value "Region1", "Description:" with the value "Region1", and "Period Of Supply Calendar:" with the value "calendar_all_time" and a dropdown arrow. At the bottom, there are four buttons: "OK", "Cancel", "Apply", and "Help".

Edit controls

The EDIT control statement uses the following syntax, required syntax is in bold:

```

EDIT

    NAME "name"

    LABEL "label"

    LOCATION left, top, length, height

    TYPE "data_type"

    NOAPPLY

    NOAUTOPOPULATE

    NOEDIT

    REQUIRED

    DISABLED

    LEFT/CENTER/RIGHT

    FORMAT "format"

    TOPICPOPULATE topic_attributes

    ONAPPLY { command }

    ONCHANGE { command }

;

```

The EDIT control statement uses the following properties:

EDIT Properties

<i>Property</i>	<i>Description</i>
NAME "name"	Optional. name is a unique name for this control. This allows other controls to reference this control, such as in another control's TOPICPOPULATE property.
LABEL "label"	Optional. label is the text string to show in a Required error message box.
LOCATION left, top, length, height	<p>where left, top, length, and height determine the location and size of this control on the page.</p> <hr/> <p>For more information, refer to LOCATION in Writing Statement</p>

Property	Description
	Properties.
TYPE "data_type"	<p>where data_type is the type of data that you want to display.</p> <p>For more information, refer to TYPE in Writing Statement Properties.</p>
NOAPPLY	Optional. If the user makes changes to the value shown in this control's field, the Apply button on the page is not enabled.
NOAUTOPOPULATE	<p>Optional. When this control is first displayed, this control will not be automatically populated, leaving the field blank.</p> <p>For more information, refer to NOAUTOPOPULATE in Writing Statement Properties.</p>
NOEDIT	Optional. This property prevents the user from editing the data in the EDIT field.
REQUIRED	Optional. This property prevents the user from leaving the page without entering data in this EDIT field. A Required error message box appears if the user tries to leave the page without filling out this EDIT field.
LEFT/CENTER/RIGHT	Optional. LEFT left-justifies the data in this field; CENTER centers it; RIGHT right-justifies it.
FORMAT format	<p>Optional. Used when the TYPE is RPS_FLOAT. format is the format used to display data in this field. The digits in format are:</p> <p># display a digit except for leading zeros</p> <p>0 display a digit, and display trailing zeros</p> <p>. display the decimal point</p> <p>Example: "###,##0.00" means display a number of up to eight-digits with two decimal places.</p>

Property	Description
TOPICPOPULATE topic_attributes	<p>Optional. topic_attributes are the attributes for TOPICPOPULATE. Use TOPICPOPULATE to get the Topic object if it has not been supplied.</p> <hr/> <p>For more information, refer to TOPICPOPULATE in Writing Statement Properties.</p> <hr/>
VALUEPOPULATE value_attributes	<p>where value_attributes are the attributes for VALUEPOPULATE. Use VALUEPOPULATE to get the value to display in the EDIT field.</p> <hr/> <p>For more information, refer to VALUEPOPULATE in Writing Statement Properties.</p> <hr/>

The EDIT control statement can use the following events:

EDIT Events

Event	Description
ONAPPLY { command }	Optional. command is the command(s) that you wish to perform when the user presses the Apply or OK button after changing the value in the EDIT field. For example, you can execute a transaction to set the slot on the server that populated this EDIT field to the value set by the user.
ONCHANGE {command}	Optional. command is the command(s) that you wish to perform when the user changes the values in the EDIT field. For example, you can disable buttons.

When you use the NAME property with this control, you can use the Disable, Enable, Refresh, Value, Topic, Hide and Show methods. For example, an event in another control can use the Hide method to hide this control.



For more information, refer to Writing Methods.

Grouping Controls Together: GROUPBOX

The GROUPBOX control statement creates a box in which the user can group controls together. This allows the user of a page to see what controls are related to each other.

Here are several GROUPBOX controls which are on the General page on the Export to RDBMS component. In this page, the GROUPBOX controls are labeled Unit, Environment, Vendor, Order Promising, Calendar, Class Info, and Attribute.

Groupbox controls

The GROUPBOX control statement uses the following syntax, required syntax is in bold:

```
GROUPBOX

    NAME "name"

    LABEL "label"

    LOCATION left, top, length, height

;
```

The GROUPBOX control statement uses the following properties:

GROUPBOX Properties

<i>Property</i>	<i>Description</i>
NAME "name"	Optional. name is a unique name for this control.
LABEL "label"	Optional. A text string placed in the upper left corner of the group box..
LOCATION left, top, length, height	<p>where left, top, length, and height determine the location and size of this control on the page.</p> <hr/> <p>For more information, refer to LOCATION in Writing Statement Properties.</p> <hr/>

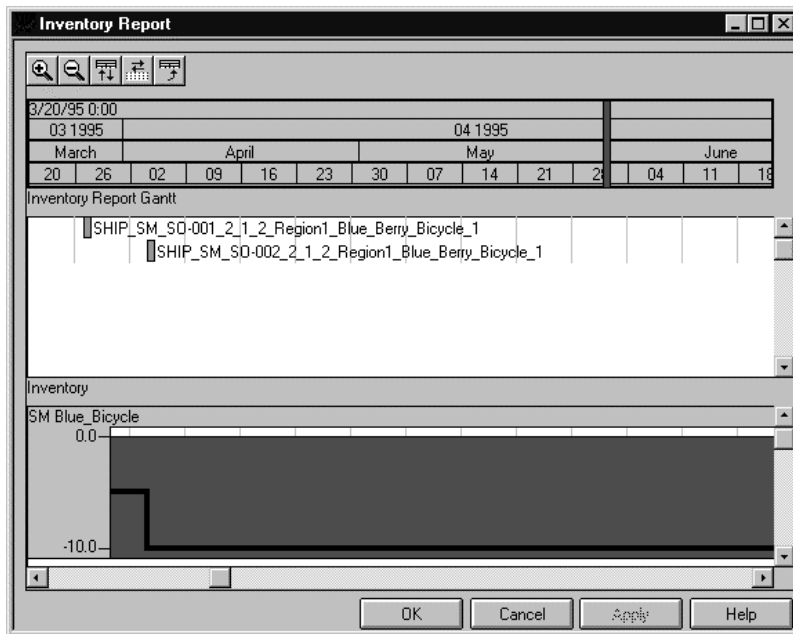
Inventory Histogram inside a CTC: INVENTORY_HISTOGRAM

The INVENTORY_HISTOGRAM control statement places an inventory histogram into a CTC.



For an example, refer to Understanding the Inventory Report Page Code.

Here is an inventory histogram on the lower half of a CTC for the Inventory Report page. It is labeled "SM Blue_Bicycle."



Inventory Histogram control

The INVENTORY_HISTOGRAM statement uses the following syntax, required syntax is in bold:

```

INVENTORY_HISTOGRAM

  LABEL "label"

  LABELHEIGHT height

  PERCENTHEIGHT percent

  TOPICPOPULATE topic_attributes

  VALUEPOPULATE value_attributes

;
  
```

The INVENTORY_HISTOGRAM control statement uses the following properties:

INVENTORY_HISTOGRAM Properties

<i>Property</i>	<i>Description</i>
LABEL "label"	where label is the text string to show on the top of this control.
LABELHEIGHT height	where height is the number of units of height for the label. 12 is a common height.
PERCENTHEIGHT percent	where percent is a number showing the

Property	Description
	percentage of the CTC that you want this control to cover. For example, if you want the control to cover all the CTC, use 100; if you want it to cover half, use 50.
TOPICPOPULATE topic_attributes	<p>Optional. topic_attributes are the attributes for TOPICPOPULATE. Use TOPICPOPULATE to get the Topic object if it has not been supplied. The topic could be a query instance.</p> <hr/> <p>For more information, refer to TOPICPOPULATE in Writing Statement Properties.</p> <hr/>
VALUEPOPULATE value_attributes	<p>where value_attributes are the attributes for VALUEPOPULATE. The value returned by the value_attributes should be a part (item) instance or an oset of part instances.</p> <hr/> <p>For more information, refer to VALUEPOPULATE in Writing Statement Properties.</p> <hr/>

PENALTY

The PENALTY control statement places a penalty field upon a page. This field displays using different colors, depending upon the value that is displayed in the field.



For an example, refer to Understanding the Score Card Page Code.

Here are penalty controls on the Score Card page, below the Score label.

Scorecard

Query Print Report Early Fence: 3/1/95 0:00:00 Late Fence: 6/1/95 0:00:00

	Count	Score		Count	Score
Request Dates	3	0.05	Safety Stock	0	0.00
Promise Dates	0	0.00	Excess Stock	0	0.00
BI Shortages	11	1.00	RM Shortages	0	0.00
Aggregate Capacity	0	0.00			

Safety Stock Excess Stock RM Shortages

Request Dates Promise Dates BI Shortages Aggregate Capacity

Sort

	Request Date*	Customer*	Shipment Start*	Shipment End*	
1	4/3/95 0:00:00	Cycles And More	4/4/95 23:59:58	4/5/95 0:00:00	S
2	4/3/95 0:00:00	Cycles And More	4/4/95 23:59:58	4/5/95 0:00:00	S
3	4/3/95 0:00:00	Cycles And More	4/4/95 23:59:58	4/5/95 0:00:00	S

OK Cancel Apply Help

Penalty control

The PENALTY statement currently displays the following colors:

- -1: grey (OFF)
- 0: green
- 1-10: yellow
- > 10: red

The PENALTY control statement uses the following syntax, required syntax is in bold:

PENALTY

NAME "name"

LOCATION left, top, length, height

TYPE "datatype"

NOEDIT

LEFT/RIGHT

TOPICPOPULATE topic_attributes

VALUEPOPULATE value_attributes

;

The PENALTY control statement uses the following properties:

PENALTY Properties

Property	Description
NAME "name"	Optional. name is a unique name for this control. This allows other controls to reference this control, such as in another control's TOPICPOPULATE property.
LOCATION left, top, length, height	<p>where left, top, length, and height determine the location and size of this control on the page.</p> <hr/> <p>For more information, refer to LOCATION in Writing Statement Properties.</p> <hr/>
TYPE "datatype"	where datatype is RPS_FLOAT.
LEFT/RIGHT	Optional. The LEFT property causes the data in the penalty field to be left-justified; RIGHT causes it to be right-justified. Default is LEFT.
TOPICPOPULATE topic_attributes	<p>Optional. topic_attributes are the attributes for TOPICPOPULATE. Use TOPICPOPULATE to get the Topic object if it has not been supplied.</p> <hr/> <p>For more information, refer to VALUEPOPULATE in Writing Statement Properties.</p> <hr/>
VALUEPOPULATE value_attributes	<p>where value_attributes are the attributes for VALUEPOPULATE. Use VALUEPOPULATE to get the value to display in the penalty field.</p> <hr/> <p>For more information, refer to VALUEPOPULATE in Writing Statement Properties.</p> <hr/>
NOEDIT	This statement prevents the user from editing the PENALTY control.

When you use the NAME property with this control, you can use the Hide, Show, and Refresh methods. For example, an event in another control can use the Refresh method to refresh the value shown in this control.



For more information, refer to Writing Methods.

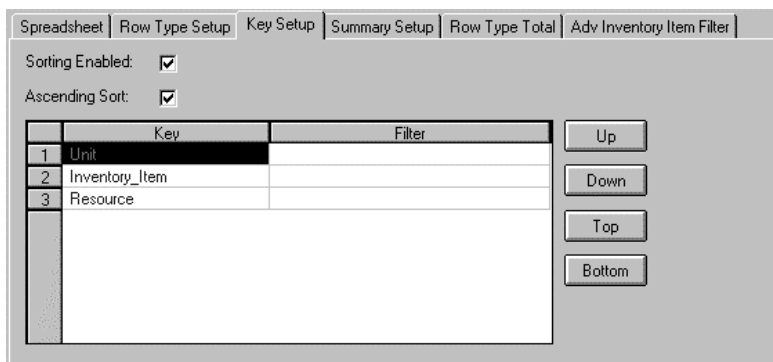
Buttons: PUSHBUTTON

The PUSHBUTTON places a button on the page. When the user pushes the button, the ONPRESS event executes a command(s).



For more information, refer to Understanding the Build Schedule (with Resource) Spreadsheet Page Code.

Here are buttons on a page. The buttons are labeled Up, Down, Top, and Bottom.



Buttons

The PUSHBUTTON control statement uses the following syntax, required syntax is in bold:

```
PUSHBUTTON
```

```
NAME "name"
```

```
LABEL "label"
```

```
LOCATION left, top, length, height
```

```
ONPRESS { command }
```

```
;
```

The PUSHBUTTON control statement uses the following properties:

PUSHBUTTON Properties

<i>Property</i>	<i>Description</i>
NAME "name"	Optional. name is a unique name for this control. This allows other controls to reference this control.
LABEL "label"	where label is the text string to show on the button.
LOCATION left, top, length, height	<p>where left, top, length, and height determine the location and size of this control on the page.</p> <hr/> <p>For more information, refer to VALUEPOPULATE in Writing Statement Properties.</p> <hr/>

The PUSHBUTTON control statement uses the following event:

```
ONPRESS { command }
```

where command is the command(s) that you wish to perform when the user pushes the button.



For a list of the commands, refer to Writing Event Commands.

The ONPRESS event occurs when a button is pressed.



For an example of ONPRESS, refer to Understanding the Build Schedule (with Resource) Spreadsheet Page Code.

When you use the NAME property with this control, you can use the Disable, Enable, Hide, and Show methods. For example, an event in another control can use the Disable and Enable methods to disable and enable this button.

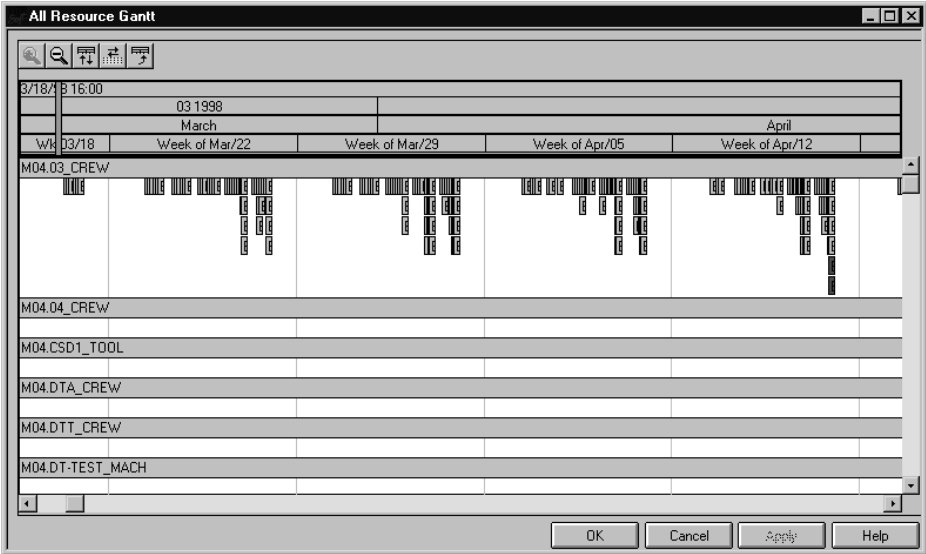


For more information, refer to Writing Methods.

Resource Gantt inside a CTC: RESOURCE_GANTT

The RESOURCE_GANTT control statement places a resource Gantt on a CTC. Refer to the AllEquipmentResourceGantt resource file for an example.

Here is a resource Gantt on the CTC for the All Resource Gantt page.



Resource Gantt

The RESOURCE_GANTT statement uses the following syntax, required syntax is in bold:

```
RESOURCE_GANTT
    LABEL "label"
    LABELHEIGHT height
    PERCENTHEIGHT percent
    TOPICPOPULATE topic_attributes
    VALUEPOPULATE value_attributes
;
```

The RESOURCE_GANTT control statement uses the following properties:

RESOURCE_GANTT Properties

Property	Description
LABEL "label"	where label is the text string to show on the top of this control.

Property	Description
LABELHEIGHT height	where height is the number of units of height for the label. 12 is a common height.
PERCENTHEIGHT percent	where percent is a number showing the percentage of the CTC that you want this control to cover. For example, if you want the control to cover all the CTC, use 100; if you want it to cover half, use 50.
TOPICPOPULATE topic_attributes	<p>Optional. topic_attributes are the attributes for TOPICPOPULATE. Use TOPICPOPULATE to get the Topic object if it has not been supplied. The topic could be a query instance.</p> <hr/> <p>For more information, refer to TOPICPOPULATE in Writing Statement Properties..</p> <hr/>
VALUEPOPULATE value_attributes	<p>where value_attributes are the attributes for VALUEPOPULATE. The value returned by the value_attributes should be an instance of equipment resource or an osset of such instances.</p> <hr/> <p>For more information, refer to VALUEPOPULATE in Writing Statement Properties.</p> <hr/>

The RESOURCE_GANTT control statement can use the following events:

RESOURCE_GANTT Events

Event	Description
MODIFY_MOVE _TASK_FORWARD	<p>Allows a task Gantt chart to move a task forward in time. The following transaction is its default value:</p> <pre>MODIFY_MOVE_TASK_FORWARD{TRANSACTION "ui_task_change_start_time(:data %[] .StartTime :task %[] .Task)"} </pre>
MODIFY_MOVE _TASK_BACKWARD	<p>Allows a task Gantt chart to move a task backward in time. The following</p>

Event	Description
	transaction is its default value: MODIFY_MOVE_TASK_FORWARD{TRANSACTION "ui_task_change_end_time(:data %[] .EndTime :task %[] .Task)"} }
MODIFY_MOVE _SHUFFLE_FORWARD	Allows a task Gantt chart to shuffle a task forward in time. The following transaction is its default value: MODIFY_SHUFFLE_TASK_FORWARD{TRANSACTION "ui_task_shuffle_start_time(:data %[] .StartTime :task %[] .Task)"} }
MODIFY_MOVE _SHUFFLE_BACKWARD	Allows a task Gantt chart to shuffle a task backward in time. The following transaction is its default value: MODIFY_SHUFFLE_TASK_BACKWARD{TRANSACTION "ui_task_shuffle_end_time(:data %[] .EndTime :task %[] .Task)"} }
MODIFY_CHANGE _DURATION	Allows a task Gantt chart to change a task's duration. The following transaction is its default value: MODIFY_CHANGE_DURATION{TRANSACTION "ui_task_change_duration(:data %[] .Duration :task %[] .Task)"} }
MODIFY_CHANGE _RESOURCE	Allows a resource Gantt chart to change a task's resource. Only valid on resource Gantt charts. The following transaction is its default value: MODIFY_CHANGE_RESOURCE{TRANSACTION "ui_task_change_duration(:task %[] .Task :resource %[] .Resource :data %[] .NewResource)"} }

The RESOURCE_GANTT events can use the following methods:

RESOURCE_GANTT Methods

Method	Description
Task	The oid of the selected task.
StartTime	The start time of the selected task.

Method	Description
EndTime	The end time of the selected task.
Duration	The duration of the selected task.
Resource	The resource that the selected task is currently using; the "from" resource.
NewResource	The resource that the selected task was dropped onto; the "to" resource.

Here is an example using the extended syntax:

```
[FORM]

BEGINFORM

    LABEL "All Equipment Resource Gantt";

BEGINCTC

    LABEL "All Equipment Resource"

    LOCATION 5, 5, 520, 270

    TOPICPOPULATE TRANSACTION "get_oid main_environment;"

    VALUEPOPULATE TRANSACTION "get_project_times(:env_name
\"main_environment\")";

    RESOURCE_GANTT

        NAME "All_Equipment_Resource"

        PERCENTHEIGHT 100

        VALUEPOPULATE TRANSACTION "ui_equipment_resource_gantt_data()"

        MODIFY_CHANGE_RESOURCE{TRANSACTION "my_new_task_change_resource(:task
%[] .Task

                                :oldresource %[] .Resource

                                :newresource %[] .NewResource)"};

        // I'm using the default transactions for all of the task move actions.

ENDCTC
```

ENDFORM;

Non-editable Text Strings: STATIC

The STATIC control statement places, on a page, a text string or other data that the user cannot edit.



For examples, refer to any of the examples in Understanding Examples Of Resource Language Code.

The Unit (site) page contains STATIC controls that display “Unit:”, “Description:”, and “Period Of Supply Calendar:”.

Static controls

The STATIC control statement most commonly uses the following syntax , which displays a supplied text string in the LABEL.

```

STATIC

    LABEL "label"

    LOCATION left, top, length, height

;
  
```

The STATIC control statement can also use the following syntax, which gets data to display from the server.



For an example, refer to Understanding the About Box Page Code.

```

STATIC

    LABEL "label"

    LOCATION left, top, length, height

    TYPE "data_type"
  
```

```

TOPICPOPULATE topic_attributes

VALUEPOPULATE value_attributes

;

```

The STATIC control statement uses the following properties:

STATIC Properties

Property	Description
LABEL "label"	where label is the text string to show on the page.
LOCATION left, top, length, height	<p>where left, top, length, and height determine the location and size of this control on the page.</p> <hr/> <p>For more information, refer to VALUEPOPULATE in Writing Statement Properties.</p> <hr/>
TYPE "data_type"	<p>where data_type is the type of data that you want to display. The user cannot edit this data.</p> <hr/> <p>For more information, refer to TYPE in Writing Statement Properties.</p> <hr/>
TOPICPOPULATE topic_attributes	<p>where topic_attributes are the attributes for TOPICPOPULATE. Use TOPICPOPULATE to get the Topic object for the STATIC control statement.</p> <hr/> <p>For more information, refer to TOPICPOPULATE in Writing Statement Properties.</p> <hr/>
VALUEPOPULATE value_attributes	<p>where value_attributes are the attributes for VALUEPOPULATE. Use VALUEPOPULATE to get the value to display in the STATIC field.</p>

<i>Property</i>	<i>Description</i>
	For more information, refer to VALUEPOPULATE in Writing Statement Properties.

When you use the NAME property with this control, you can use the Refresh, Value, Topic, Hide and Show methods. For example, an event in another control can use the Hide method to hide this control.



For more information, refer to Writing Methods.

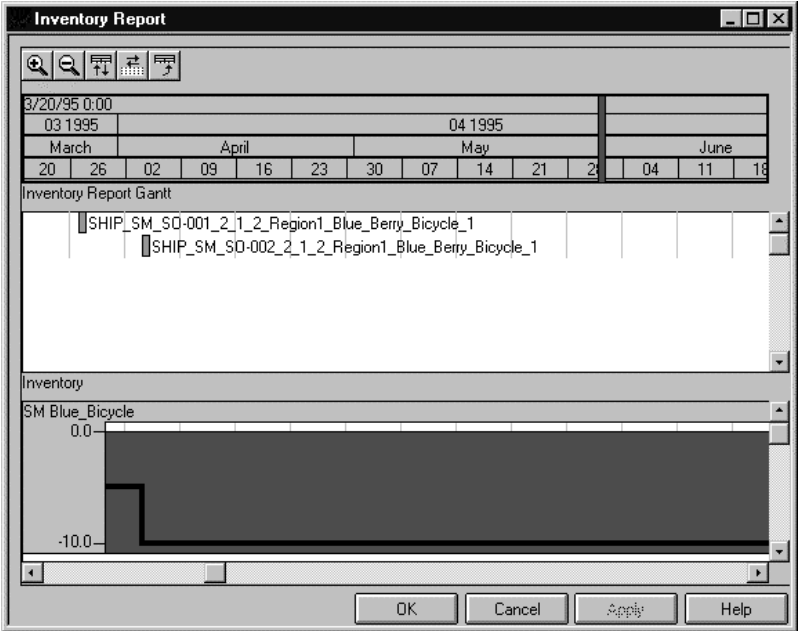
Task Gantt inside a CTC: TASK_GANTT

The TASK_GANTT control statement places a task Gantt on a CTC.



For an example, refer to Understanding the Inventory Report Page Code.

Here is a task Gantt on the upper half of a CTC for the Inventory Report page. It is labeled “Inventory Report Gantt.”



Task Gantt

The TASK_GANTT statement uses the following syntax, required syntax is in bold:

```
TASK_GANTT

    LABEL "label"

    LABELHEIGHT height

    PERCENTHEIGHT percent

    TOPICPOPULATE topic_attributes

    VALUEPOPULATE value_attributes

;
```

The TASK_GANTT control statement uses the following properties:

TASK_GANTT Properties

Property	Description
LABEL "label"	where label is the text string to show on the top of this control.
LABELHEIGHT height	where height is the number of units of height for the label. 12 is a common height.
PERCENTHEIGHT percent	where percent is a number showing the percentage of the CTC that you want this

Property	Description
	control to cover. For example, if you want the control to cover all the CTC, use 100; if you want it to cover half, use 50.
TOPICPOPULATE topic_attributes	<p>Optional. topic_attributes are the attributes for TOPICPOPULATE. Use TOPICPOPULATE to get the Topic object if it has not been supplied. The topic could be a query instance.</p> <hr/> <p>For more information, refer to TOPICPOPULATE in Writing Statement Properties.</p> <hr/>
VALUEPOPULATE value_attributes	<p>where value_attributes are the attributes for VALUEPOPULATE. The value returned by the value_attributes should be an instance of equipment resource or an osset of such instances.</p> <hr/> <p>For more information, refer to VALUEPOPULATE in Writing Statement Properties.</p> <hr/>

The TASK_GANTT control statement can use the following events:

TASK_GANTT Events

Event	Description
MODIFY_MOVE _TASK_FORWARD	<p>Allows a task Gantt chart to move a task forward in time. The following transaction is its default value:</p> <pre>MODIFY_MOVE_TASK_FORWARD{TRANSACTION "ui_task_change_start_time(:data %[] .StartTime :task %[] .Task)"} </pre>
MODIFY_MOVE _TASK_BACKWARD	<p>Allows a task Gantt chart to move a task backward in time. The following transaction is its default value:</p> <pre>MODIFY_MOVE_TASK_FORWARD{TRANSACTION "ui_task_change_end_time(:data %[] .EndTime :task %[] .Task)"} </pre>

Event	Description
MODIFY_MOVE _SHUFFLE_FORWARD	Allows a task Gantt chart to shuffle a task forward in time. The following transaction is its default value: MODIFY_SHUFFLE_TASK_FORWARD{TRANSACTION "ui_task_shuffle_start_time(:data %[] .StartTime :task %[] .Task)"} }
MODIFY_MOVE _SHUFFLE_BACKWARD	Allows a task Gantt chart to shuffle a task backward in time. The following transaction is its default value: MODIFY_SHUFFLE_TASK_BACKWARD{TRANSACTION "ui_task_shuffle_end_time(:data %[] .EndTime :task %[] .Task)"} }
MODIFY_CHANGE _DURATION	Allows a task Gantt chart to change a task's duration. The following transaction is its default value: MODIFY_CHANGE_DURATION{TRANSACTION "ui_task_change_duration(:data %[] .Duration :task %[] .Task)"} }

The TASK_GANTT events can use the following methods:

TASK_GANTT Methods

Method	Description
Task	The oid of the selected task.
StartTime	The start time of the selected task.
EndTime	The end time of the selected task.
Duration	The duration of the selected task.
Resource	The resource that the selected task is currently using; the "from" resource.

Here is an example using the extended syntax:

```
[FORM]

BEGINFORM

    LABEL "All Equipment Task Gantt";
```

```

BEGINCTC

    LABEL "All Equipment Resource"

    LOCATION 5, 5, 520, 270

    TOPICPOPULATE TRANSACTION "get_oid main_environment;"

    VALUEPOPULATE TRANSACTION "get_project_times(:env_name
\"main_environment\");

RESOURCE_GANTT

    NAME "All_Equipment_Resource"

    PERCENTHEIGHT 100

    VALUEPOPULATE TRANSACTION "ui_equipment_resource_gantt_data()"

    MODIFY_MOVE_TASK_FORWARD{TRANSACTION "my_new_task_change_start_time(

        :mystarttime %[].StartTime

        :task %[].Task)"};

// I'm using the default transactions for all of the other task move actions.

ENDCTC

ENDFORM;

```

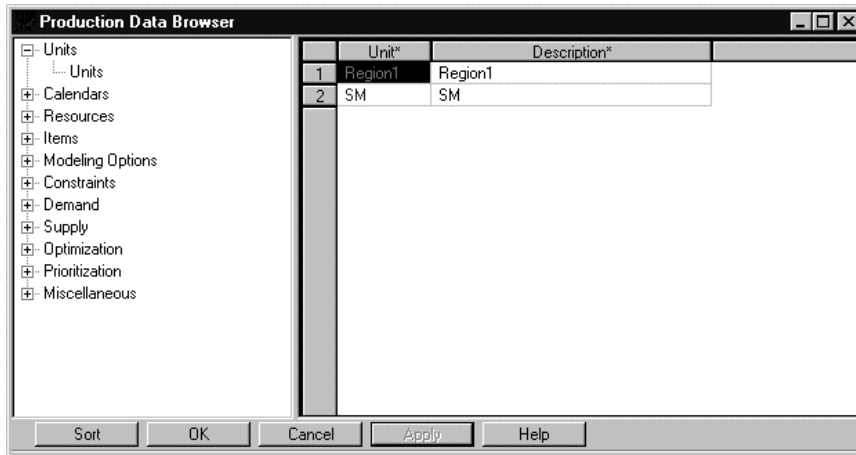
Writing a Data Browser

The BEGINBROWSER and ENDBROWSER statements create the data browser. The data browser shows a windows with two panes. The left pane is a tree pane; this tree pane allows you to display several folder names in the right pane. By clicking a folder name, you can then display class names. And by clicking a class name, you can then display the instances of that class in a grid on the right side of the data browser.



For an example, refer to Understanding the Data Browser Page Code.

Here is the data browser, displaying the site (unit) class.



Data Browser showing site (unit) class

BEGINBROWSER, ENDBROWSER

BEGINBROWSER and ENDBROWSER creates the page for the data browser.



For more information, refer to VALUEPOPULATE in Writing Statement Properties.

The BEGINBROWSER and ENDBROWSER statements use the following syntax, required syntax is in bold:

```
BEGINBROWSER

  LABEL "label";

  BEGINFOLDER // there is one or more

      // BEGINFOLDER, ENDFOLDER;

  // BEGINFOLDER, ENDFOLDER syntax goes here

  ENDFOLDER;

ENDBROWSER;
```

The BEGINBROWSER control statement uses the following properties and statements:

BEGINBROWSER, ENDBROWSER Properties

<i>Property</i>	<i>Description</i>
LABEL "label"	where label is the text string to show on the top margin of the data browser.

Property	Description
BEGINFOLDER, ENDFOLDER	Refer to the following section, BEGINFOLDER, ENDFOLDER, for a description of these statements.

BEGINFOLDER, ENDFOLDER

BEGINFOLDER and ENDFOLDER creates a folder. A folder represents a indented node on the tree pane—the left pane—in the OBJECTBROWSER. It displays one class.



For more information, refer to VALUEPOPULATE in Writing Statement Properties.

The BEGINFOLDER and ENDFOLDER statements use the following syntax, required syntax is in bold:

```

BEGINFOLDER

    LABEL "label";

    BEGINGRID // there is one or more

        //   BEGINGRID, ENDGRID;

    LABEL "label"

    TOPICCLASS "class_name"

    COLUMN; // there is one or more COLUMN

    ENDGRID;

ENDFOLDER;

```

The BEGINFOLDER control statement uses the following properties and statements:

BEGINFOLDER, ENDFOLDER Properties and Statements

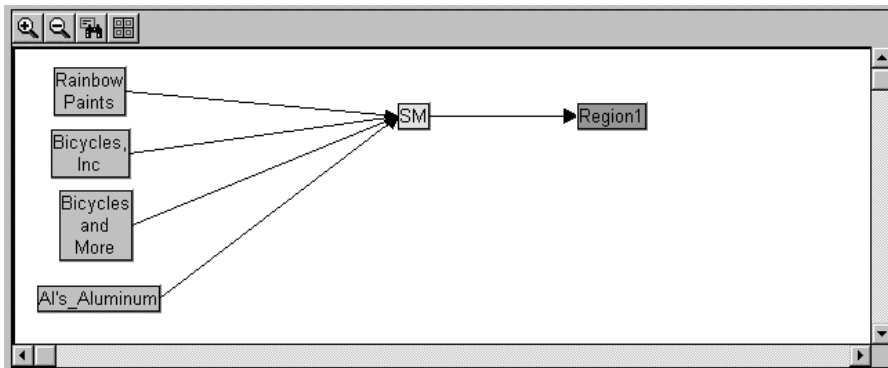
Properties and Statements			Description
LABEL "label"			where label is the text string to show for this folder in the left side of the data browser.
BEGINGRID, ENDGRID;			The BEGINGRID and ENDGRID statements create a grid within the folder that is displayed in the right pane of the OBJECTBROWSER. The rows of this

Properties and Statements			Description
			<p>grid contain the instances of the class being displayed. The columns contain the slots for this class that you want to display.</p> <p>The BEGINGRID, ENDGRID statements are the same as the BEGINGRID, ENDGRID statements discussed earlier.</p> <hr/> <p>For more information about grids, refer to Grids: BEGINGRID, ENDGRID.</p> <hr/> <p>These BEGINGRID, ENDGRID statements use only the following properties:</p>
	LABEL "label"		where label is the text to display for this class in the left side of the data browser.
	TOPICCLASS "class_name"		where class_name is the name of the class for which you want to display the instances in this grid.
	COLUMN		<p>The grid—BEGINGRID, ENDGRID—uses COLUMN statements to place columns in the control.</p> <hr/> <p>For more information about COLUMN, refer to Columns in a Grid or Spreadsheet: COLUMN.</p> <hr/> <p>The COLUMN statements are the same as the COLUMN statement for Grids and Spreadsheets, and can use the same properties and events. This COLUMN statement uses the following properties in a slightly more specialized way:</p>
		LABEL "label"	where label is the text string to show at the top of the column.
		VALUEPOPULATE attributes	where attributes are the attributes for VALUEPOPULATE. Use this to determine the values that you want to display. For example, the Site (unit) class uses VALUEPOPULATE PATH

Properties and Statements			Description
			"." to display the display_name for the site (unit) instance.

Writing a Diagramming Control

This control may be added to any applicable Client page using the following Client resource language statements. This control adds a diagramming control to your page. The diagramming control allows you to use boxes—nodes—and lines and arrows—arcs—to graphically display relationships between planning data. For example, the diagram below shows the purchase and transfer options between vendors such as Rainbow Paints and Bicycles, Inc, and the SM and Region1 units.



Diagramming control



For an example, refer to Understanding the Supply Chain Viewer Page Code.

BEGINDIAGRAM, ENDDIAGRAM

BEGINDIAGRAM and ENDDIAGRAM create a diagramming control. A diagramming control consists of a toolbar and drawing area. The toolbar has buttons which are managed by the diagramming control. The drawing area contains nodes and arcs. Nodes show such objects as units or vendors. Arcs show the relationships between nodes, such as transfer orders and purchase orders.



For an example, refer to Understanding the Supply Chain Viewer Page Code.

The syntax of BEGINDIAGRAM contains properties which:

- set the location of the diagramming control within a page or tab dialog,

- set drawing properties of the drawing area, nodes, and arcs,
- set the transactions to be run to retrieve nodes and arcs from the server,
- set the instance menus for the nodes and arcs.

The BEGINDIAGRAM and ENDDIAGRAM statements use the following syntax, required syntax is in bold:

BEGINDIAGRAM

NAME "name"

LOCATION left, top, length, height;

DRAWINGAREA drawing_attributes

NODE node_attributes

ARC arc_attributes

VALUEPOPULATE TRANSACTION transaction_attributes

NOAUTOPOPULATE

BEGINDIAGRAMMENU

diagram_menu_items

ENDMENU;

BEGINNODEMENU

node_menu_items

ENDMENU;

BEGINARCSMENU

arc_menu_items

ENDMENU;

ENDDIAGRAM;

The BEGINDIAGRAM control statement uses the following properties and statements:

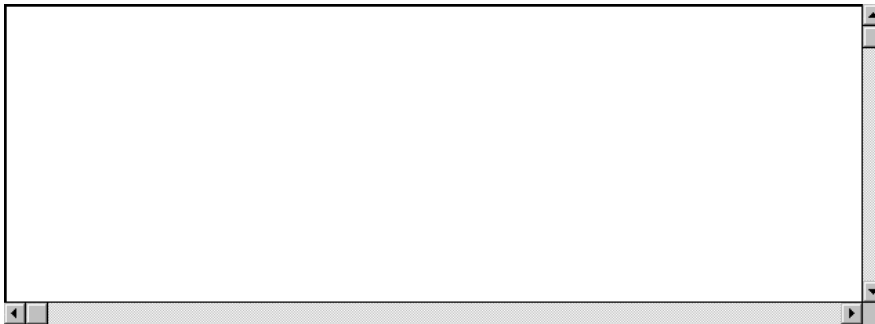
BEGINDIAGRAM, ENDDIAGRAM Properties

<i>Properties and Statements</i>	<i>Description</i>
NAME "name"	name is a unique name for this control. This allows other controls to reference this control, such as in another control's TOPICPOPULATE property.
LOCATION left, top, length, height	<p>where left, top, length, and height determine the location and size of this control on the page.</p> <hr/> <p>For more information, refer to LOCATION in Writing Statement Properties.</p> <hr/>
DRAWINGAREA drawing_attributes	<p>where drawing_attributes define the properties of the drawing area. A default DRAWINGAREA will be supplied if this is not used.</p> <hr/> <p>For more information, refer to DRAWINGAREA.</p> <hr/>
NODE node_attributes	<p>where node_attributes define the properties of entities displayed in the drawing area. A default NODE will be supplied if this is not used.</p> <hr/> <p>For more information, refer to NODE.</p> <hr/>
ARC arc_attributes	<p>where arc_attributes define the properties of arcs displayed in the drawing area. A default ARC will be supplied if this is not used.</p> <hr/> <p>For more information, refer to ARC.</p> <hr/>
VALUEPOPULATE TRANSACTION transaction_attributes	<p>Optional. transaction_attributes are the attributes for VALUEPOPULATE TRANSACTION. This transaction should be written to retrieve values that are used by nodes and arcs.</p> <hr/> <p>For more information, refer to VALUEPOPULATE in Writing</p>

Properties and Statements	Description
	Statement Properties.
NOAUTOPOPULATE	<p>Optional. When the diagram is first displayed, it will not be automatically populated, leaving it blank.</p> <p>For more information, refer to NOAUTOPOPULATE in Writing Statement Properties.</p>
BEGINNODEMENU node_menu_items ENDMENU	<p>The BEGINNODEMENU and ENDMENU statements identify the menu items to be displayed in a popup menu when a node is right-clicked.</p> <p>For more information, refer to Node and Arc Instance Menus: BEGINNODEMENU and BEGINARCSMENU.</p>
BEGINARCSMENU arc_menu_items ENDMENU	<p>The BEGINARCSMENU and ENDMENU statements identify the menu items to be displayed in a popup menu when a arc is right-clicked.</p> <p>For more information, refer to Node and Arc Instance Menus: BEGINNODEMENU and BEGINARCSMENU.</p>

DRAWINGAREA

DRAWINGAREA defines the properties of the drawing area, printer settings, and initial zoom level. Following is a drawing area without nodes and arcs displayed.



Drawing Area Without Nodes and Arcs

DRAWINGAREA use the following syntax, required syntax is in bold:

```
DRAWINGAREA

    BACKCOLOR "color-index"

    NOEDITMODE

    GRIDHEIGHT "grid-value"

    GRIDLINECOLOR "color-value"

    GRIDLINES

    GRIDWIDTH "grid-value"

;
```

The DRAWINGAREA control statement uses the following properties:

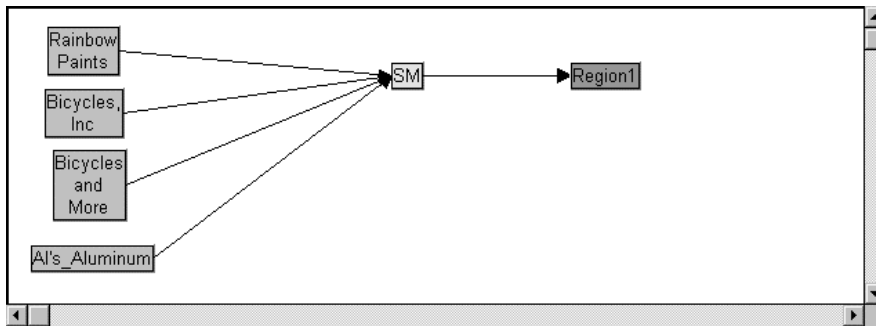
DRAWINGAREA Properties

Property	Description
BACKCOLOR "color_index"	Optional. color_index is a number that determines the background color of the drawing area. The numbers are 2 for cyan, 3 for gray, 4 for orange, 6 for magenta, 7 for blue, 8 for teal, 9 for money green, 10 for dark green, 11 for sky blue, 12 for brown, 13 for dark blue, 14 for purple, 15 for olive, 16 for white, and 17 for black. The default is 3 for gray.
NOEDITMODE	Optional. Entities and arcs cannot be modified—moved or resized—and right-clicking the drawing area will not display popup menus.

Property	Description
GRIDHEIGHT "grid_value"	Optional. Used with GRIDLINES. This property determines the vertical spacing of the gridlines. grid_value is an integer from 1 to 999 indicating the vertical spacing of the gridlines. The number is approximately the number of pixels. The default is 100.
GRIDLINECOLOR "color_index"	Optional. GRIDLINES must be used with this property. color-index is a number that determines the color used to paint gridlines in the drawing area. The numbers are 0 for green, 1 for yellow, 2 for cyan, 3 for gray, 4 for orange, 5 for red, 6 for magenta, 7 for blue, 8 for teal, 9 for money green, 10 for dark green, 11 for sky blue, 12 for brown, 13 for dark blue, 14 for purple, 15 for olive, 16 for white, and 17 for black. The default is 17 for black.
GRIDLINES	Optional. Causes gridlines to be displayed in the drawing area.
GRIDWIDTH "grid_value"	Optional. Used with GRIDLINES. This property determines the horizontal spacing of gridlines in IDO. grid_value is an integer from 1 to 999 indicating the vertical spacing of the grid lines. The number is approximately the number of pixels. The default is 100.

NODE

NODE defines the attributes for nodes displayed in the drawing area. The following diagram shows nodes for the Rainbow Paints vendor, the Bicycles Inc. Vendor, the Bicycles and More vendor, the Al's Aluminum vendor, the SM unit, and the Region1 unit.



Nodes in a Drawing Area

NODE use the following syntax, required syntax is in bold:

NODE

NOAUTORESIZE

FILLCOLOR "color-index"

NOCANMOVE

NOCANRESIZE

GRAPHIC "graphic-index"

FRAME "frame"

TEXTORIENTATION "text-orientation"

;

The NODE control statement uses the following properties:

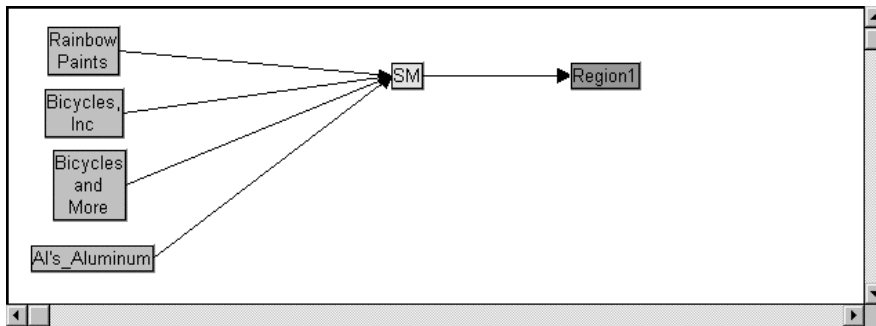
NODE Properties

<i>Property</i>	<i>Description</i>
NOAUTORESIZE	Optional. Causes a node to not automatically resize to fit the text of the node. If the node is a bitmap this property has no effect.
FILLCOLOR "color_index"	color-index is a number that determines the color used to paint the node's background. The numbers are 0 for green, 1 for yellow, 2 for cyan, 3 for gray, 4 for orange, 5 for red, 6 for magenta, 7 for blue, 8 for teal, 9 for money green, 10 for dark green, 11 for sky blue, 12 for brown, 13 for dark blue, 14 for purple, 15 for olive, 16 for white, and 17 for black.

Property	Description
	You should avoid using 0 for green, 1 for yellow, and 5 for red, since these are reserved for scorecards. The default is 3 for gray.
NOCANMOVE	Optional. Causes a node to not be movable by the user with the mouse.
NOCANRESIZE	Optional. Causes a node to not be resizable by the user with the mouse.
GRAPHIC "graphic_index"	Assigns the display bitmaps to the nodes. Use this property to assign drawing shapes and bitmaps to the node. The values are RECTANGLE or 0, ROUNDRECT or 1, and CIRCLE or 2. If you do not use these values, you must use an absolute pathname to a bitmap the will be displayed. The default is RECTANGLE.
FRAME "frame"	The property determines the type of 3D frame drawn around a node. The values are NONE or 0, HEAVYOUTDENT or 1, OUTDENT or 2, HEAVYINDENT or 3, INDENT or 4, and SHADOW or 5. The default is HEAVYOUTDENT.
TEXTORIENTATION "text_orientation"	This property determines the orientation of the nodes text if the node is displaying a graphic. For rectangles and ellipses the text is always CENTERed. The values are BOTTOM or 0, TOP or 1, LEFT or 2, RIGHT or 3, and CENTER or 4.

ARC

ARC defines the attributes for arcs displayed in the drawing area. The arcs are the lines and arrows shown in the following drawing area. The following diagram shows the arcs for the purchase options between the vendors—Rainbow Paints, Bicycles, Inc, Bicycles and More, Al's Aluminum—and the SM unit, and the transfer option between the SM unit and the Region1 unit.



Arcs in a Drawing Area

ARC use the following syntax, required syntax is in bold:

ARC

LINECOLOR "color-index"

NOCANMOVEMIDPOINTS

DESTINATIONARROW "arrow-type"

SOURCEARROW "arrow-type"

THICKNESS "arrow-thickness"

TYPE "arrow-line-type"

;

The ARC control statement uses the following properties:

ARC Properties

Property	Description
LINECOLOR "color_index"	color-index is a number that determines the color used to paint the arc—a line and arrow. The numbers are 0 for green, 1 for yellow, 2 for cyan, 3 for gray, 4 for orange, 5 for red, 6 for magenta, 7 for blue, 8 for teal, 9 for money green, 10 for dark green, 11 for sky blue, 12 for brown, 13 for dark blue, 14 for purple, 15 for olive, 16 for white, and 17 for black. Default is 17 for black.
NOCANMOVEMIDPOINTS	Optional. Causes an arc's midpoints to not be moveable by the user with the mouse. This property has no effect if the arc has no midpoints.

Property	Description
DESTINATIONARROW	Optional. Determines the type of arrow, if any, to draw at the destination point of a arc. Possible values: NONE or 0, STANDARD or 1, NARROW or 2, WIDE or 3, SWEPT or 4, WHITECIRCLE or 5, BLACKCIRCLE or 6. Default is NONE.
SOURCEARROW "arrow_type"	Optional. Determines the type of arrow, if any, to draw at the source point of an arc. Possible values: NONE or 0, STANDARD or 1, NARROW or 2, WIDE or 3, SWEPT or 4, WHITECIRCLE or 5, BLACKCIRCLE or 6. Default is STANDARD.
THICKNESS	Optional. This property determines the thickness of a line drawn representing a arc. A value from 1 to 5—5 being thickest—which specifies the thickness of the line drawn to represent the arc. Default is 1.
TYPE "arrow_line_type"	Optional. Determines the type of line drawn for the arc. Possible values: STRAIGHT, 3PT (3 point), 4PT (4 point), 90DEGREE (elbow). Default is STRAIGHT.

Node and Arc Instance Menus: BEGINNODEMENU and BEGINARCSMENU

The instance menus for arcs and nodes are contained in the property sheet pages for the node or arc. In the case of the Supply Chain Viewer, the node class has two subclasses: vendor node and unit (site) node, and the arc class has two subclasses: transfer order node and purchase order node. The property sheet pages for these subclasses appear when you double-click a node or arc, and the instance menu for these property sheet pages appear when you right-click a node or arc.

BEGINNODEMENU, ENDMENU

The BEGINNODEMENU and ENDMENU statements can either create an instance menu for all nodes, or if there are instance menus for the node subclasses, they can add menu items to the node subclass instance menu. For example, if a node for a vendor is right-clicked, the menu items listed in BEGINNODEMENU will be displayed, followed by a separator, and then followed by

the vendor node instance menu items. This allows you to add a menu item to all node instance menus from the BEGINDIAGRAM/ENDDIAGRAM code, rather than adding that menu item to every node instance menu. To use these statements, place them between BEGINDIAGRAM and ENDDIAGRAM.

The BEGINNODEMENU and ENDMENU statements can contain the following keywords, which are the same keywords as for BEGINMENU and ENDMENU.



For more information, refer to Writing Menu Statements.

- BEGINSUBMENU and ENDSUBMENU
- SEPARATOR
- MENUITEM

An optional menu item attribute, CLASSNAME, can be used with MENUITEM. This supports menu item selections for specific classes, such as SITE and SUPPLIER, from the same pop-up menu.

BEGINARCSMENU, ENDMENU

The BEGINARCSMENU and ENDMENU statements can either create an instance menu for all arcs, or if there are instance menus for the arc subclasses, they add menu items to the arc subclass instance menu. For example, if an arc for a transfer option is right-clicked, the menu items listed in BEGINARCSMENU will be displayed, followed by a separator, and then followed by the transfer order arc instance menu items. This allows you to add a menu item to all arc instance menus from the BEGINDIAGRAM/ENDDIAGRAM code, rather than adding that menu item to every arc instance menu. To use these statements, place them between BEGINDIAGRAM and ENDDIAGRAM.

The BEGINARCSMENU and ENDMENU statements can contain the following keywords, which are the same keywords as for BEGINMENU and ENDMENU.



For more information, refer to Writing Menu Statements.

- BEGINSUBMENU and ENDSUBMENU
- SEPARATOR
- MENUITEM

An optional menu item attribute, CLASSNAME, can be used with MENUITEM. This supports menu item selections for specific classes, such as SITE and SUPPLIER, from the same pop-up menu.

BEGINDIAGRAMMENU, ENDMENU

The BEGINDIAGRAMMENU and ENDMENU statements create a popup menu that appears when a diagram area is right-clicked.

The BEGINDIAGRAMMENU and ENDMENU statements can contain the following keywords, which are the same keywords as for BEGINMENU and ENDMENU.



For more information, refer to Writing Menu Statements.

- BEGINSUBMENU and ENDSUBMENU
- SEPARATOR
- MENUITEM

An optional menu item attribute, CLASSNAME, can be used with MENUITEM. This supports menu item selections for specific classes, such as SITE and SUPPLIER, from the same pop-up menu.

Understanding Examples Of Resource Language Code

This section shows several pages. For each page, it lists the code that creates the page, and comments explaining some of the statements in the code. The code can be found in the Resources folder; these pages are PROduction Planning pages. The following pages are shown:

- Add Unit page
 - Contains comments for the following statements: BEGINFORM, STATIC, EDIT, COMBOBOX
- Build Schedule (with Resource) Spreadsheet page
 - Contains comments for the following statements: BEGINPROPSHEET, STATIC, EDIT, BEGINPAGE, PUSHBUTTON, BEGINSPREADSHEET, COLUMN, CHECKBOX
- Control panel
 - Contains comments for the following statements: BEGINFORM, STATIC, PUSHBUTTON, CONTROLSLIDER
- Score Card
 - Contains comments for the following statements: BEGINPROPSHEET, STATIC, EDIT, PUSHBUTTON, PENALTY
- Purchase Order: contains a set menu and an instance menu

Contains comments for the following statements: BEGINPROPSHEET, STATIC, EDIT, BEGINPAGE, COMBOBOX, PUSHBUTTON, BEGINGRID, COLUMN, BEGINMENU, MENUITEM, SEPARATOR

- Inventory Report

Contains comments for the following statements: BEGINFORM, BEGINCTC, TASK_GANTT, INVENTORY_HISTOGRAM

- About Box

Contains comments for the following statements: BEGINFORM, BITMAP

- Item Filter Page

Contains comments for the following command: SENDPARENT.

- Data Browser

Contains comments for the following statements: BEGINBROWSER, BEGINFOLDER

- Supply Chain Viewer

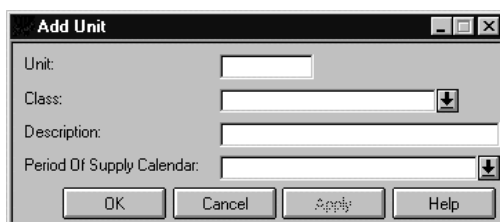
Contains comments for the following statements: BEGINDIAGRAM, ENDDIAGRAM

Understanding the Add Unit Page Code

Here is the Add Unit page. Following it is the code that creates it, from the AddSite file.

To display this page, perform the following steps:

1. Start the Planning Client. The Home Base page appears.
2. Click **Browse Data**. The Data Browser page appears.
3. Double-click + **Units**. **Units** appears below it, and + **Units** changes to - **Units**.
4. Right-click **Units**. The Add Units menu appears.
5. Click **Add** on the Add Unit menu.



Add Unit page

[FORM]

The BEGINFORM statement defines the beginning of the Add Unit page. The ENDFORM statement, later in the code, defines the end of the page. The LABEL property places the label "Add Unit" at the top of the page. The TOPICPOPULATE NONE property means that this page does not have a topic. This page is used to collect data that will be sent to the server and used to add a unit.

The commands in the ONAPPLY event are executed when the user presses the APPLY button on the Add Unit page. The TRANSACTION command executes a transaction that uses the data input by the user to add a unit.

```
BEGINFORM
```

```
  LABEL "Add Unit"
```

```
  TOPICPOPULATE NONE
```

```
  ONAPPLY{TRANSACTION "ui_add_site(:site_name %[Site].Value
                        :site_class %[Class].Value
                        :description %[Description].Value)";
           :period_of_supply_calendar %[Calendar].Value)"};
```

These STATIC and EDIT controls display the input field for the user to enter the new unit's id.

This STATIC control statement displays the label "Unit:" on the page.

```
  STATIC
```

```
    LABEL "Unit:"
```

```
    LOCATION 5, 5, 45, 11;
```

This EDIT control statement displays an edit field where the user enters a unit id. The TYPE of this EDIT is a character string. The REQUIRED property means that the user must enter a unit id in this field.

```
  EDIT
```

```
    LABEL "Unit"
```

```
    NAME "Site"
```

```
    LOCATION 100, 5, 45, 11
```

```
    TYPE "RPS_STRING"
```

```
    REQUIRED;
```

These STATIC and COMBO controls display the list for the user to choose the new unit's (site's) class.

This STATIC control displays the label "Class:" on the page.

```

STATIC

    LABEL "Class:"

    LOCATION 5, 20, 45, 11;

```

This COMBOBOX control statement displays the list of subclasses for the site class. The user can then choose one of these subclasses for the new site (unit) class. The MENUPOPULATE property executes a transaction that gathers all the instances of the Site class to be displayed in a menu combobox.



For more information about that transaction, refer to `form_get_subclasses_of_class`.

```

COMBOBOX

    NAME "Class"

    LOCATION 50, 20, 115, 11

    TYPE "RPS_CLASS"

    MENUPOPULATE TRANSACTION "form_get_subclasses_of_class(:class_name
\"Site\)");

STATIC

    LABEL "Description:"

    LOCATION 5, 35, 45, 11

    TYPE "RPS_STRING";

EDIT

    NAME "Description"

    LOCATION 50, 35, 135, 11

    TYPE "RPS_STRING";

ENDFORM;

```

The following STATIC and COMBOBOX controls work in a similar fashion to the ones listed above. STATIC lists "Period Of Supply Calendar:" on the page, and the COMBOBOX control statement displays the list of calendars for the site (unit) class. The user can then choose one of these calendars for the new site class. The MENUPOPULATE property executes a transaction that gathers all the instances of the Calendar class to be displayed in a menu combobox.

```

STATIC

```

```

    LABEL "Period Of Supply Calendar:"

    LOCATION 5, 50, 95, 11;

COMBOBOX

    NAME "Calendar"

    LOCATION 100, 50, 135, 11

    TYPE "RPS_CLASS"

    MENUPOPULATE TRANSACTION "form_get_instances_of_class(

                                :class_name \"Calendar\");

```

The following code is the PepperCode from the site_ui_actions.spl file. This code shows you the transactions that are called by the Add Unit page. This documentation shows only this example, and does not explain it.



For more information about writing PepperCode programs, refer to PeopleSoft PepperCode.

```

// Forward Declaration

#include <cpp/cpp_utility.spl>
#include "site_transactions.spl"

class Site;

action ui_add_site

    (input: string site_name,

     input: class<Site> site_class,

     input: string description,

     input: instance<Calendar> period_of_supply_calendar = 0,

     local: string class_name,

     local: string calendar_name,

     local: action<transaction_create_site> transaction_create_site,

     output: string exit_msg = "",

```

```
no_context:)

{
  if (NOT(site_class)) {
    class_name = "";
  }
  else {
    class_name = GET_NAME_OF_CLASS(site_class);
  }

  if (NOT(period_of_supply_calendar)) {
    calendar_name = "";
  }
  else {
    calendar_name = period_of_supply_calendar.name;
  }

  execute transaction_create_site(:site_name site_name,
                                  :class_name class_name,
                                  :description description,
                                  :calendar_name calendar_name);

  exit_msg = transaction_create_site.exit_msg;

  if (EQ(transaction_create_site.status, SUCCEED)) {
    succeed();
  }
  else {
    fail();
  }
}
```

```

action ui_change_site_period_of_supply_calendar

(input: instance<Site> site,

input: instance<Calendar> period_of_supply_calendar,

local: action<transaction_change_site_period_of_supply_calendar>
transaction_change_site_period_of_supply_calendar,

output: string exit_msg,

no_context:)
{
    execute transaction_change_site_period_of_supply_calendar(
        :site_name site.name,
        :period_of_supply_calendar_name period_of_supply_calendar.name);

    exit_msg = transaction_change_site_period_of_supply_calendar.exit_msg;

    if(EQ(transaction_change_site_period_of_supply_calendar.status, SUCCEED))
        succeed();

    else
        fail();
}

```

Understanding the Build Schedule (with Resource) Spreadsheet Page Code

Here is the Build Schedule (with Resource) Spreadsheet page. Following it is the code that creates it, from the Build_Schedule_Equipment_Spreadsheet file.

To display this page, perform the following steps:

1. Start the Planning Client. The Home Base page appears.
2. Click **Build Schedule**. The Build Schedule (with Resource) page appears.

Build Schedule (with Resource)

Start Date: 3/1/95 0:00:00
 Number Of Periods: 7
 Period Duration: S_DAY

Key	Filter
1 Unit	
2 Inventory_Item	
3 Resource	

Spreadsheet | Row Type Setup | Key Setup | Summary Setup | Row Type Total | Adv Inventory Item Filter

Query | Print Report

	Unit	Inventory Item	Resource	Row Type	Past Due	3/1/95	3/2/95	3/3/95	3/4/95
1	SM	Blue_Bicycle	Assemble_Work_Cent	Production Supply	0.00	0.00	0.00	0.00	0.00
2	SM	Blue_Bicycle	Assemble_Work_Cent	Production Order Supp	0.00	0.00	0.00	0.00	0.00
3	SM	Blue_Bicycle	Assemble_Work_Cent	Non Production Order	0.00	0.00	0.00	0.00	0.00
4	SM	Blue_Bicycle	Assemble_Work_Cent	Production Supply Star	0.00	0.00	0.00	0.00	0.00
5	SM	Blue_Frame	Paint_Work_Center	Production Supply	0.00	0.00	0.00	0.00	0.00
6	SM	Blue_Frame	Paint_Work_Center	Production Order Supp	0.00	0.00	0.00	0.00	0.00
7	SM	Blue_Frame	Paint_Work_Center	Non Production Order	0.00	0.00	0.00	0.00	0.00
8	SM	Blue_Frame	Paint_Work_Center	Production Supply Star	0.00	0.00	0.00	0.00	0.00
9	SM	Blue_Frame	Weld_Work_Center	Production Supply	0.00	0.00	0.00	0.00	0.00
10	SM	Blue_Frame	Weld_Work_Center	Production Order Supp	0.00	0.00	0.00	0.00	0.00

OK Cancel Apply Help

Build Schedule (with Resource) Spreadsheet page

Build Schedule (with Resource)

Start Date: 3/1/95 0:00:00
 Number Of Periods: 7
 Period Duration: S_DAY

Key	Filter
1 Unit	
2 Inventory_Item	
3 Resource	

Spreadsheet | Row Type Setup | Key Setup | Summary Setup | Row Type Total | Adv Inventory Item Filter

Sorting Enabled: ☒
 Ascending Sort: ☒

Key	Filter
1 Unit	
2 Inventory_Item	
3 Resource	

Up
Down
Top
Bottom

OK Cancel Apply Help

Build Schedule (with Resource) page, showing key setup

Build Schedule (with Resource)

Start Date: 3/1/95 0:00:00
 Number Of Periods: 7
 Period Duration: S_DAY

Key	Filter
1 Unit	
2 Inventory_Item	
3 Resource	

Spreadsheet | Row Type Setup | Key Setup | Summary Setup | Row Type Total | Adv Inventory Item Filter

Key	Total	Average	MAX	MIN
1 Unit				
2 Inventory_Item				
3 Resource				

OK Cancel Apply Help

Build Schedule (with Resource) page, showing summary setup

[PROPERTY SHEET]

The BEGINPROPSHEET statement defines the beginning of the Build Schedule with Resource property sheet page. The ENDPROPSHEET statement, later in the code, defines the end of the property sheet page. The LABEL property places the label "Build Schedule (with Resource)" at the top of the page. The TOPICPOPULATE property executes a transaction to create a temporary page backing object. This object is created to use as the topic of this page, and contains data to be displayed in this page. The ONCLOSE event deletes the temporary page backing object.

```
BEGINPROPSHEET

  LABEL "Build Schedule (with Resource) "

  TOPICPOPULATE TRANSACTION

    "transaction_get_spreadsheet_form_topic(:spreadsheet_template_name
      \"build_schedule_spreadsheet_with_equipment\") "

  ONCLOSE TRANSACTION "transaction_release_spreadsheet_form_topic
    (:spreadsheet %[] .FormTopic)";
```

These STATIC and EDIT controls display the start date.

This STATIC control displays the label "Start Date:" on the page.

```
STATIC

  LABEL "Start Date:"

  LOCATION 5, 5, 50, 11;
```

This EDIT control statement displays the start date for the spreadsheet. The TYPE of this EDIT is date. The NOAPPLY property means that when the user makes changes to the date in this field, the APPLY button is not enabled when the date is changed; if the user presses the APPLY or OK buttons, any changes made to this field are not sent back to the server. The VALUEPOPULATE property populates the edit field with the start_date slot on the temporary page backing object. Later in the code, the page backing object for this spreadsheet uses its start_date slot to determine the starting time for the variable columns in the following spreadsheet.

```
EDIT

  NAME "Start_Date"

  LOCATION 80, 5, 70, 11

  TYPE "RPS_DATE"

  ALWAYSEDIT

  NOAPPLY
```

```
VALUEPOPULATE PATH "start_date";
```

Later in the code, the page backing object for this spreadsheet uses its `number_of_periods` slot to determine the number of columns for the variable columns in the following spreadsheet.

```
STATIC
```

```
  LABEL "Number Of Periods:"
```

```
  LOCATION 5, 20, 70, 11;
```

```
EDIT
```

```
  NAME "Number_Of_Periods"
```

```
  LOCATION 80, 20, 40, 11
```

```
  TYPE "RPS_INT"
```

```
  ALWAYSEDIT
```

```
  NOAPPLY
```

```
  RIGHT
```

```
  VALUEPOPULATE PATH "number_of_periods";
```

Later in the code, the page backing object for this spreadsheet uses its `period_duration` slot to determine the time duration for the variable columns in the following spreadsheet.

```
STATIC
```

```
  LABEL "Period Duration:"
```

```
  LOCATION 5, 35, 65, 11;
```

```
COMBOBOX
```

```
  NAME "Period_Duration"
```

```
  LOCATION 80, 35, 65, 11
```

```
  TYPE "RPS_ENUM"
```

```
  ALWAYSEDIT
```

```
  NOAPPLY
```

```
  VALUEPOPULATE PATH "period_duration"
```

```
  MENUPOPULATE TRANSACTION "form_get_enums_of_class(:class_name  
  \"Period_Duration\")";
```

```
BEGINGRID
```

```

NAME "Keys"

TOPICCLASS "Key"

NOAPPLY

LOCATION 205, 5, 220, 41

VALUEPOPULATE PATH "keys";

COLUMN

    NAME "Key"

    LABEL "Key"

    WIDTH "100"

    TYPE "RPS_CLASS"

    NOEDIT

    VALUEPOPULATE TRANSACTION "form_get_slot_class (:topic_object %[] .RowTopic
                                :slot_name \"key_class\")" PATH "key_class";

COLUMN

    NAME "Filter"

    LABEL "Filter"

    WIDTH "135"

    TYPE "RPS_STRING"

    VALUEPOPULATE PATH "filter"

ENDGRID;

```

The BEGINPAGE statement defines the beginning of the spreadsheet page on this component, or spreadsheet page on this page. The ENDPAGE statement, later in the code, defines the end of the spreadsheet page. The LABEL property places the label "Spreadsheet" on the tab of the page.

```

BEGINPAGE

    LABEL "Spreadsheet";

```

The PUSHBUTTON statement defines a button on the page/page. The LABEL property places the label "Query" on the button. The ONPRESS event calls the transaction that fills the spreadsheet with values when the user presses this button.

```

PUSHBUTTON

    LABEL "Query"

```

```
LOCATION 5, 5, 50, 14
```

```
ALWAYS EDIT
```

```
ONPRESS {TRANSACTION "ui_generate_spreadsheet(:spreadsheet %[] .Topic
                                     :start_date %[Start_Date] .Value
                                     :number_of_periods %[Number_Of_Periods] .Value
                                     :period_duration %[Period_Duration] .Value)"};
                                     :key1 %[Keys] .ValueAtRow(0)
                                     :filter1 %[Keys] .RowColumnValue(0,1)
                                     :key2 %[Keys] .ValueAtRow(1)
                                     :filter2 %[Keys] .RowColumnValue(1,1)
                                     :key3 %[Keys] .ValueAtRow(2)
                                     :filter3 %[Keys] .RowColumnValue(2,1))"};
```

```
PUSHBUTTON
```

```
LABEL "Print Report"
```

```
LOCATION 60, 5, 50, 14
```

```
ONPRESS {LOADFORM ("PrintEditor", :Passed_Object %[] .Topic)};
```

The **BEGINSPREADSHEET** statement defines the beginning of a spreadsheet on the page/page. The **ENDSPREADSHEET** statement, later in the code, defines the end of the spreadsheet. The **TOPICCLASS** property is not used. The **LOCKCOLUMNS** property indicates that the first 4 columns on the left are not moved when the user scrolls the spreadsheet horizontally. The **VALUEPOPULATE** property populates the spreadsheet rows with the spreadsheet_rows osset on the temporary page backing object—the topic object of this page. **VARIABLECOLUMNFORMAT** sets the format for the columns to the right of the first 4 columns. The values for these columns—the columns with the date headers—are determined by the temporary page backing object. In the case of this page backing object, the date on those columns starts at `start_date`, the duration from column to column is set by `period_duration`, and the number of columns is set by `number_of_periods`.

```
BEGINSPREADSHEET
```

```
TOPICCLASS "Spreadsheet_Row"
```

```
LOCATION 5, 25, 510, 120
```

```
VARIABLECOLUMNFORMAT "###,##0.00"
```

```
LOCKCOLUMNS "4"
```

```

PREAPPLY { TRANSACTION
    "ui_modify_spreadsheet_generate_detailed
    (:spreadsheet %[] .Topic
    :generate_detailed 0
    :generate_spreadsheet 0)" }

POSTAPPLY { TRANSACTION
    "ui_modify_spreadsheet_generate_detailed
    (:spreadsheet %[] .Topic
    :generate_detailed 1
    :generate_spreadsheet 1)" }

VALUEPOPULATE PATH "spreadsheet_rows";

```

This COLUMN control statement creates a column with the LABEL "Unit". The WIDTH property defines how wide the column is. This column is of TYPE instance. The NOEDIT property means that the value of this column is not changeable by the user on this page. This VALUEPOPULATE property executes a transaction to populate this column with the key1_instance slot from the temporary page backing object—the topic object of this page.

```

COLUMN

NAME "Site"

LABEL "Unit"

WIDTH "45"

TYPE "RPS_INSTANCE"

NOEDIT

VALUEPOPULATE PATH "key1_instance";

```

The remaining three locked columns on this spreadsheet all behave similarly to the previous COLUMN.

```

COLUMN

NAME "Inventory_Part"

LABEL "Inventory Item"

WIDTH "80"

TYPE "RPS_INSTANCE"

NOEDIT

VALUEPOPULATE PATH "key2_instance";

```

```

COLUMN

```

```

NAME "Equipment_Resource"

LABEL "Resource"

WIDTH "80"

TYPE "RPS_INSTANCE"

NOEDIT

VALUEPOPULATE PATH "key3_instance";

```

COLUMN

```

NAME "Row_Type"

LABEL "Row Type"

WIDTH "80"

TYPE "RPS_INSTANCE"

NOEDIT

VALUEPOPULATE PATH "row_type";

```

```
ENDSPREADSHEET;
```

```
ENDPAGE;
```

The BEGINPAGE statement defines the beginning of the Row Type Setup page on the page, or page in this component. The ENDPAGE statement, later in the code, defines the end of the Row Type Setup page. The LABEL property places the label "Row Type Setup" on the tab of the page.

BEGINPAGE

```

LABEL "Row Type Setup" ALWAYSEDIT;

PUSHBUTTON

LABEL "Add"

LOCATION 5, 5, 40, 14

ONPRESS {LOADFORM("AddRowType", :Passed_Spreadsheet %[] .Topic)};

```

This button is labeled "Delete". The ONPRESS event for this button is executed when this button is pressed, and it executes the ASKUSER commands. The ASKUSER command creates a pop-up window to ask the user "Delete Row Type?". If the user clicks **OK**, the TRANSACTION command executes the transaction to delete the selected row in the "Row Types" grid. The SelectedRow method passes the instance of the selected row to the transaction, so that the transaction can delete that row. If the user clicks **Cancel**, that delete transaction is not executed.

PUSHBUTTON

LABEL "Delete"

LOCATION 50, 5, 40, 14

ONPRESS{ASKUSER "Delete Row Type?",

TRANSACTION "ui_modify_spreadsheet_row_type_position

(:spreadsheet %[].Topic

:row_type %[Row Types].SelectedRow

:modification \"DELETE\\")\"};

BEGINGRID

NAME "Row Types"

TOPICCLASS "Row_Type"

LOCATION 5, 25, 160, 100

VALUEPOPULATE PATH "actual_row_types";

COLUMN

NAME "Row_Type"

LABEL "Row Type"

WIDTH "140"

TYPE "RPS_INSTANCE"

NOEDIT

VALUEPOPULATE PATH ".";

ENDGRID;

PUSHBUTTON

LABEL "Up"

LOCATION 170, 25, 40, 14

ONPRESS{TRANSACTION "ui_modify_spreadsheet_row_type_position


```
(:spreadsheet %[] .Topic
:row_type %[Row Types].SelectedRow
:modification \"MOVE_UP_ONE\");
```

PUSHBUTTON

```
LABEL "Down"

LOCATION 170, 45, 40, 14

ONPRESS{TRANSACTION "ui_modify_spreadsheet_row_type_position
(:spreadsheet %[] .Topic
:row_type %[Row Types].SelectedRow
:modification \"MOVE_DOWN_ONE\");
```

PUSHBUTTON

```
LABEL "Top"

LOCATION 170, 65, 40, 14

ONPRESS{TRANSACTION "ui_modify_spreadsheet_row_type_position
(:spreadsheet %[] .Topic
:row_type %[Row Types].SelectedRow
:modification \"MOVE_TO_HEAD\");
```

PUSHBUTTON

```
LABEL "Bottom"

LOCATION 170, 85, 40, 14

ONPRESS{TRANSACTION "ui_modify_spreadsheet_row_type_position
(:spreadsheet %[] .Topic
:row_type %[Row Types].SelectedRow
:modification \"MOVE_TO_TAIL\");
```

STATIC

```

    LABEL "Maximum Rows:"

    LOCATION 5, 130, 75, 11;

EDIT

    NAME "Maximum_Rows"

    LOCATION 85, 130, 75, 11

    TYPE "RPS_INT"

    RIGHT

    VALUEPOPULATE PATH "max_rows"

    ONAPPLY{TRANSACTION "ui_modify_spreadsheet_max_rows

        (:spreadsheet %[] .Topic

            :max_rows %[] .Value)"};

ENDPAGE;

```

The BEGINPAGE statement defines the beginning of the key setup page on the page, or page in this component. The ENDPAGE statement, later in the code, defines the end of the key setup page. The LABEL property places the label "Key Setup" on the tab of the page.

```

BEGINPAGE

    LABEL "Key Setup" ALWAYSEDIT;

```

These STATIC and CHECKBOX control statements display the Sorting Enabled check box on the key setup page.

This STATIC control statement creates the label "Sorting Enabled" for the following check box.

```

STATIC

    LABEL "Sorting Enabled:"

    LOCATION 5, 5, 60, 11;

```

This CHECKBOX control statement creates a check box. This check box is of TYPE integer. This VALUEPOPULATE property populates the check box with the sorting_enabled slot on temporary page backing object—the topic object of this page. The ONAPPLY event sends a transaction to the server that sets the sorting_enabled slot on the temporary page backing object.

```

CHECKBOX

    NAME "Sorting_Enabled"

    LOCATION 70, 5, 100, 11

    TYPE "RPS_INT"

```

```

VALUEPOPULATE PATH "sorting_enabled"

ONAPPLY{TRANSACTION "ui_modify_spreadsheet_sorting_enabled

    (:spreadsheet %[] .Topic

        :sorting_enabled %[] .Value)"};

STATIC

    LABEL "Ascending Sort:"

    LOCATION 5, 20, 60, 11;

CHECKBOX

    NAME "Ascending_Sort"

    LOCATION 70, 20, 100, 11

    TYPE "RPS_INT"

    VALUEPOPULATE PATH "ascending_sort"

    ONAPPLY{TRANSACTION "ui_modify_spreadsheet_ascending_sort

        (:spreadsheet %[] .Topic

            :ascending_sort %[] .Value)"};

BEGINGRID

    NAME "Keys"

    TOPICCLASS "Key"

    LOCATION 5, 35, 255, 100

    VALUEPOPULATE PATH "keys";

COLUMN

    NAME "Key"

    LABEL "Key"

    WIDTH "100"

    TYPE "RPS_CLASS"

    NOEDIT

    VALUEPOPULATE TRANSACTION "form_get_slot_class

        (:topic_object %[] .RowTopic

            :slot_name \"key_class\");

```

```

COLUMN

    NAME "Filter"

    LABEL "Filter"

    WIDTH "135"

    TYPE "RPS_STRING"

    VALUEPOPULATE PATH "filter"

    ONAPPLY{TRANSACTION "ui_modify_spreadsheet_key_filter

        (:key %[] .RowTopic

        :filter %[] .Value)"};

ENDGRID;

PUSHBUTTON

    LABEL "Up"

    LOCATION 265, 35, 40, 14

    ONPRESS{TRANSACTION "ui_modify_spreadsheet_key_position

        (:spreadsheet %[] .Topic

        :key %[Keys] .SelectedRow

        :modification \"MOVE_UP_ONE\")"};

PUSHBUTTON

    LABEL "Down"

    LOCATION 265, 55, 40, 14

    ONPRESS{TRANSACTION "ui_modify_spreadsheet_key_position

        (:spreadsheet %[] .Topic

        :key %[Keys] .SelectedRow

        :modification \"MOVE_DOWN_ONE\")"};

PUSHBUTTON

    LABEL "Top"

    LOCATION 265, 75, 40, 14

    ONPRESS{TRANSACTION "ui_modify_spreadsheet_key_position

        (:spreadsheet %[] .Topic

```

```

:key %[Keys].SelectedRow

:modification \"MOVE_TO_HEAD\")\"};

PUSHBUTTON

LABEL "Bottom"

LOCATION 265, 95, 40, 14

ONPRESS{TRANSACTION "ui_modify_spreadsheet_key_position

(:spreadsheet %[] .Topic

:key %[Keys].SelectedRow

:modification \"MOVE_TO_TAIL\")\"};

ENDPAGE;

```

The BEGINPAGE statement defines the beginning of the summary setup page on the page, or summary setup page in this component. The ENDPAGE statement, later in the code, defines the end of the summary setup page. The LABEL property places the label "Summary Setup" on the tab of the page.

```

BEGINPAGE

LABEL "Summary Setup" ALWAYSEDIT;

```

The BEGINGRID statement defines the beginning of a grid on the summary setup page. The ENDGRID statement, later in the code, defines the end of the grid. The TOPICCLASS property is not used. The VALUEPOPULATE property populates the grid with the keys osct on the temporary page backing object—the topic object of this page.

```

BEGINGRID

NAME "Summary_Setup"

TOPICCLASS "Key"

LOCATION 5, 5, 311, 100

VALUEPOPULATE PATH "keys";

COLUMN

NAME "Key"

LABEL "Key"

WIDTH "150"

TYPE "RPS_CLASS"

```

```
NOEDIT
```

```
VALUEPOPULATE TRANSACTION "form_get_slot_class
```

```
(:topic_object %[].RowTopic :slot_name \"key_class\");
```

The following COLUMN uses a WIDGET property to insert a check box into each field in the column. The VALUEPOPULATE property populates each of the check boxes with the total_flag slot on the instance of this grid's topic, "key".

```
COLUMN
```

```
NAME "Total"
```

```
LABEL "Total"
```

```
WIDTH "35"
```

```
TYPE "RPS_INT"
```

```
WIDGET "CHECKBOX"
```

```
VALUEPOPULATE PATH "total_flag"
```

```
ONAPPLY{TRANSACTION "ui_modify_key_total_flag
```

```
(:key %[].RowTopic :total_flag %[].Value)"};
```

```
COLUMN
```

```
NAME "Average"
```

```
LABEL "Average"
```

```
WIDTH "35"
```

```
TYPE "RPS_INT"
```

```
WIDGET "CHECKBOX"
```

```
VALUEPOPULATE PATH "average_flag"
```

```
ONAPPLY{TRANSACTION "ui_modify_key_average_flag
```

```
(:key %[].RowTopic :average_flag %[].Value)"};
```

```
COLUMN
```

```
NAME "MAX"
```

```
LABEL "MAX"
```

```
WIDTH "35"
```

```

TYPE "RPS_INT"

WIDGET "CHECKBOX"

VALUEPOPULATE PATH "max_flag"

ONAPPLY{TRANSACTION "ui_modify_key_max_flag

    (:key %[].RowTopic :max_flag %[].Value)"};

COLUMN

NAME "MIN"

LABEL "MIN"

WIDTH "35"

TYPE "RPS_INT"

WIDGET "CHECKBOX"

VALUEPOPULATE PATH "min_flag"

ONAPPLY{TRANSACTION "ui_modify_key_min_flag

    (:key %[].RowTopic :min_flag %[].Value)"};

ENDGRID;

ENDPAGE;

```

The remaining code up to the ENDPROPSHEET statements for the Build Schedule (with Resource) is not shown up. It contains controls statements that have examples elsewhere in this section.

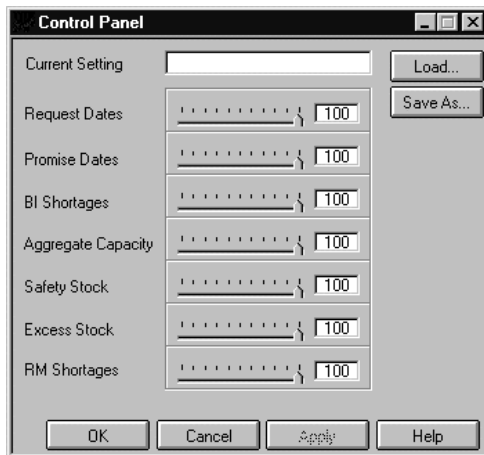
```
ENDPROPSHEET;
```

Understanding the Control panel Code

Here is the Control panel page. Following it is the code that creates it, from the ControlPage file.

To display this page, perform the following steps:

1. Start the Planning Client. The Home Base page appears.
2. Click **Control panel**. The Control panel appears.



Control panel page

FORM]

The BEGINFORM statement defines the beginning of the Control panel page. The ENDFORM statement, later in the code, defines the end of the page. The commands in the ONAPPLY event are executed when the user presses the APPLY button on the Control panel page. The METHOD commands enable the buttons labeled "Load..." and "Save As...". The TRANSACTION command executes the specified transaction.

```
BEGINFORM
```

```
    LABEL "Control panel"
```

```
    ONAPPLY { METHOD %[Load].Enable(),
```

```
              METHOD %[Save].Enable(),
```

```
              TRANSACTION "transaction_clear_control_page_setting_name()" };
```

This STATIC control statement displays "Current Setting".

```
    STATIC
```

```
        LABEL "Current Setting"
```

```
        LOCATION 6, 7, 70, 25;
```

This EDIT control statement displays the current control setting name for the control panel. The TYPE of this EDIT is a character string. The NOEDIT property means that the user can not change the character string displayed in this control. The TOPICPOPULATE property gets the instance whose key is the object deconflict_env, and the VALUEPOPULATE property populates this EDIT with the control_page_setting_name slot on that instance.

```
        EDIT
```

```
            LOCATION 73, 5, 100, 11
```

```
            TYPE "RPS_STRING"
```



```
NOEDIT
```

```
TOPICPOPULATE TRANSACTION "get_oid deconflict_env;"
```

```
VALUEPOPULATE PATH "control_page_setting_name";
```

This PUSHBUTTON control statement creates a button with the label "Load...". The NAME allows the earlier BEGINFORM statement to enable this button in its ONAPPLY event. The ONPRESS event executes the command COMMAND (32778) when the user presses the "Load..." button. This invokes a C++ action defined by the message number 32778.

```
PUSHBUTTON
```

```
NAME "Load"
```

```
LABEL "Load..."
```

```
LOCATION 182, 5, 45, 14
```

```
ONPRESS {COMMAND ( 32778 )};
```

This PUSHBUTTON control statement creates a button with the label "Save As...". The NAME allows the earlier BEGINFORM statement to enable this button in its ONAPPLY event. The ONPRESS event executes the command COMMAND (32780) when the user presses the "Save As..." button. This invokes a C++ action defined by the message number 32780.

```
PUSHBUTTON
```

```
LABEL "Save As..."
```

```
NAME "Save"
```

```
LOCATION 182, 21, 45, 14
```

```
ONPRESS {COMMAND ( 32780 )};
```

These STATIC and CONTROLSLIDER statements create the request dates slider.

This STATIC control displays the label "Request Dates".

```
STATIC
```

```
LABEL "Request Dates"
```

```
LOCATION 5, 29, 70, 29 ;
```

This CONTROLSLIDER control statement creates a slider which displays the current class weight for the request date constraint, and it also creates a field that shows the numeric value of the weight. The TOPICPOPULATE property gets the Request Milestone Constraint, and the VALUEPOPULATE property populates the field with the class_weight slot from the constraint. The commands in the ONCHANGE event are executed when the user changes the value for the request date weight. The two METHODS disable the buttons named "Load" and "Save As...".

The TRANSACTION command in the ONAPPLY event is executed when the user presses the APPLY button. That TRANSACTION sets the class_weight slot from the constraint.

```

CONTROLSLIDER

LOCATION 73, 18, 100, 24

TYPE "RPS_FLOAT"

TOPICPOPULATE TRANSACTION "get_oid Request_Milestone_Constraint;"

VALUEPOPULATE PATH "class_weight"

ONCHANGE { METHOD %[Load].Disable(), METHOD %[Save].Disable() }

ONAPPLY { TRANSACTION "transaction_set_class_weight

                (:constraint_class \"Request_Milestone_Constraint\"

                :weight %[.Value])" };

```

The remainder of the code up to the ENDFORM statement for the Control panel is not shown. It contains control statements are CONTROLSLIDERS and their respective STATICS. They all behave similarly to the previous CONTROLSLIDER.

```
ENDFORM;
```

Understanding the Score Card Page Code

Here is the Score Card page. Following it is the code that creates it, from the ScoreCard file.

To display this page, perform the following steps:

1. Start the Planning Client. The Home Base page appears.
2. Click **Score Card**. The Score Card Page appears.

Score Card page

[PROPERTY SHEET]

The BEGINPROPSHEET statement defines the beginning of the Score Card page. Actually, this is a property sheet page. The ENDPROPSHEET statement, later in the code, defines the end of the page. The LABEL property places the label "Scorecard" at the top of the page. The TOPICPOPULATE property executes a transaction to get the mfg scorecard instance.

```
BEGINPROPSHEET
```

```
    LABEL "Scorecard"
```

```
    TOPICPOPULATE TRANSACTION "ui_get_scorecard()";
```

This PUSHBUTTON control statement creates a button with the label "Query". The ONPRESS event executes the transaction specified in the TRANSACTION command when this button is pressed. This transaction gets score card data that is used to fill in fields on this page.

```
PUSHBUTTON
```

```
    LABEL "Query"
```

```
    LOCATION 5, 5, 40, 14
```

```
    ALWAYS EDIT
```

```
    ONPRESS { TRANSACTION "ui_fill_in_score_card(:score_card %[] .FormTopic) "};
```

This PUSHBUTTON control statement creates a button with the label "Print Report". The ONPRESS event loads the page specified in the LOADFORM command when this button is pressed.

```
PUSHBUTTON
```

```
  LABEL "Print Report"
```

```
  LOCATION 49, 5, 55, 14
```

```
  ONPRESS { LOADFORM ("PrintEditor", :Passed_Object %[] .Topic) };
```

These STATIC and EDIT controls display the early fence.

This STATIC control displays the label "Early Fence:" on the page.

```
STATIC
```

```
  LABEL "Early Fence:"
```

```
  LOCATION 110, 5, 50, 11;
```

This EDIT control statement displays the current value of the early fence. The TYPE property of this edit field is date. The TOPICPOPULATE property gets the topic of the Early Fence field, which is the main environment instance. The VALUEPOPULATE property populates this field with the early_fence slot on the main environment. The ONAPPLY event is executed when the APPLY or OK button is pressed, and it executes the transaction to modify the value of the early fence.

```
EDIT
```

```
  NAME "Early Fence"
```

```
  TYPE "RPS_DATE"
```

```
  LOCATION 153, 5, 70, 11
```

```
  TOPICPOPULATE TRANSACTION "get_oid main_environment;"
```

```
  VALUEPOPULATE PATH "early_fence"
```

```
  ONAPPLY{TRANSACTION "ui_time_change_early_fence(:data %[] .Value)"};
```

These STATIC and EDIT control statements display the late fence.

This STATIC control statement displays the label "Late Fence:" on the page.

```
STATIC
```

```
  LABEL "Late Fence:"
```

```
  LOCATION 230, 5, 50, 11;
```

This EDIT control statement displays the current value of the late fence. The TYPE property of this edit field is date. The TOPICPOPULATE property gets the topic of the Late Fence field, which is the main environment instance. The VALUEPOPULATE property populates this field with the late_fence slot on the main environment. The ONAPPLY event is executed when the APPLY or OK button is pressed, and it executes the transaction to modify the value of the late fence.

EDIT

NAME "Late Fence"

LOCATION 273, 5, 70, 11

TYPE "RPS_DATE"

TOPICPOPULATE CONTROL "%[Early Fence]"

VALUEPOPULATE PATH "late_fence"

ONAPPLY{TRANSACTION "ui_time_change_late_fence(:data %[] .Value)"};

This STATIC control statement displays the label "Count".

STATIC

LABEL "Count"

LOCATION 82, 24, 30, 11;

This STATIC control statement displays the label "Score".

STATIC

LABEL "Score"

LOCATION 132, 24, 30, 11;

STATIC

LABEL "Count"

LOCATION 257, 24, 30, 11;

STATIC

LABEL "Score"

LOCATION 307, 24, 30, 11;

These STATIC and EDIT control statements display the request date penalty.

This STATIC control statement displays the label "Request Dates" on the page.

STATIC

LABEL "Request Dates"

LOCATION 5, 35, 75, 11;

This EDIT control statement displays the current value of the request date penalty. The TYPE property of this edit field is integer. The TOPICPOPULATE property executes a transaction to get the topic of this control, which is the scorecard element. The VALUEPOPULATE property populates this field with the value of the number_of_violations slot on the scorecard element.

```

EDIT

NAME "Request Dates"

LOCATION 80, 35, 40, 11

TYPE "RPS_INT"

NOEDIT

RIGHT

TOPICPOPULATE TRANSACTION "ui_get_scorecard_element

    (:score_card %[] FormTopic

        :constraint_class_name \"Requested_Milestone_Constraint\") \"

VALUEPOPULATE PATH "number_of_violations";

```

This PENALTY control statement displays the value of the request date penalty. The TYPE property of this field is float. The NOEDIT property means the value of this field is not changeable by the user on this page. RIGHT means the field is displayed right-justified. The TOPICPOPULATE property gets the topic of the Request Date field, which is the scorecard element. The VALUEPOPULATE property executes a transaction to get the value of the request date penalty, and uses PATH to subscribe to this value.



For a description of subscribe, refer to VALUPOPULATE in Writing Statement Properties.

```

PENALTY

LOCATION 130, 35 , 40, 11

TYPE "RPS_FLOAT"

NOEDIT

RIGHT

TOPICPOPULATE CONTROL "%[Request Dates] "

VALUEPOPULATE TRANSACTION "ui_get_penalty_value

    (:element %[].Topic)" PATH "penalty";

```

The rest of the code up to the ENDPROPSHEET statements for the Score Card is not shown. The remainder of the STATIC and PENALTY control statements on this page are similar to the previous STATIC and PENALTY. The rest of the code also contains other control statements, which have examples elsewhere in this section.

```

ENDPROPSHEET;

```

Understanding the Purchase Order Page Code

Here are the Purchase Order page and menus. Following them is the code that creates them, from the Purchase_Order file.

To display this page, perform the following steps:

1. Start the Planning Client. The Home Base page appears.
2. Click **Browse Data**. The Data Browser page appears.
3. Click + **Supply. Purchase Orders**—along with several other items you can choose—appears below it and to the right.
4. Click **Purchase Orders**. A list of purchase orders, along with a list of Units, Items, and other information appears on the Data Browser page.
5. Double-click a purchase order number. The Purchase Order page appears.

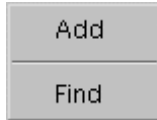
Line*	Item Unit*	Item*	UOM*	Qty Ordered*	Qty Released*	Unrel'd Qty*	Unit Price*
1	SM	Red_Paint	Each	5.0	1.0	4.0	0.0
2	SM	Blue_Paint	Each	5.0	1.0	4.0	0.0
3	SM	Green_Paint	Each	5.0	1.0	4.0	0.0

Purchase Order page showing Lines page

Purchase Order page showing General page

Following is the set menu for the Purchase Order class. To display this menu, perform the following steps:

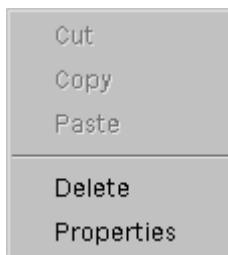
1. Start the Planning Client. The Home Base page appears.
2. Click **Browse Data**. The Data Browser page appears.
3. Click + Supply. **Purchase Orders** and **Planned Orders** appear below it and to the right.
4. Right-click **Purchase Orders**. The Purchase Order set menu appears.



Purchase Order set menu

Following is the instance menu for the Purchase Order class. To display this menu, perform the following steps:

1. Start the Planning Client. The Home Base page appears.
2. Click **Browse Data**. The Data Browser page appears.
3. Click + Supply. **Purchase Orders** and **Planned Orders** appear below it and to the right.
4. Click **Purchase Orders**. A list of purchase orders, along with a list of Units, Items, and other information appears on the Data Browser page.
5. Right-click a purchase order name. The Purchase Order instance menu appears.



Purchase Order instance menu

The BEGINCLASS and ENDCLASS control statements are no longer required.

```
[CLASS]

BEGINCLASS

    NAME "Purchase_Order"

    LABEL "Purchase_Order";
```



```
ENDCLASS;
```

The BEGINPROPSHEET statement defines the beginning of the Purchase Order page. Actually, this is a property sheet page. The ENDPROPSHEET statement, later in the code, defines the end of the page. The LABEL property places the label "Purchase Order" at the top of the page. The topic of this purchase order is its Parents Topic, which is a parameter that was passed in when this property sheet page was loaded. This topic is an instance of the purchase order class.

```
[PROPERTYSHEET]
```

```
BEGINPROPSHEET
```

```
  LABEL "Purchase Order";
```

These STATIC and EDIT control statements display the purchase order unit (site).

This STATIC control displays the label "Unit:" on the page.

```
  STATIC
```

```
    LABEL "Unit:"
```

```
    LOCATION 5, 5, 23, 11;
```

This EDIT control statement displays the purchase order's unit (site). The TYPE property of this edit field is on instance. The NOEDIT property means this field is not changeable by the user on this page. This VALUEPOPULATE property populates this field with the value of the site slot on the purchase order instance.

```
  EDIT
```

```
    NAME "Site"
```

```
    LOCATION 27, 5, 45, 11
```

```
    TYPE "RPS_INSTANCE"
```

```
    NOEDIT
```

```
    VALUEPOPULATE PATH "site";
```

These STATIC and EDIT control statements display the purchase order name.

This STATIC control statement displays the label "Purchase Order:" on the page.

```
  STATIC
```

```
    LABEL "Purchase Order:"
```

```
    LOCATION 89, 5, 68, 11;
```

This EDIT control statement displays the purchase order number. The TYPE property of this edit field is instance. The NOEDIT property means this field is not changeable by the user on this page. This VALUEPOPULATE property, which uses a PATH ".", populates this field with the value of the display_name slot on the purchase order instance. PATH "." means to populate with

the value of the default slot for the topic of the control. The value of this control is the same as the topic of the page: the purchase order instance.

```

EDIT

NAME "Purchase Order"

LOCATION 156, 5, 90, 11

TYPE "RPS_INSTANCE"

NOEDIT

VALUEPOPULATE PATH ".";

```

The BEGINPAGE statement defines the beginning of the General page on the Purchase Order component. The ENDPAGE statement, later in the code, defines the end of the General page. This LABEL property places the label "General" on this page's tab.

```

BEGINPAGE

LABEL "General";

```

These STATIC and COMBOBOX control statements displays the purchase order's vendor.

This STATIC control statement displays the label "Vendor:" on the page.

```

STATIC

LABEL "Vendor:"

LOCATION 5, 5, 50, 11;

```

This COMBOBOX control statement displays the current vendor for this purchase order. The TYPE property of this field is instance. The MENUPOPULATE property executes a transaction that gathers all the instances of the Vendor class to be displayed in a menu combobox. The VALUEPOPULATE property serves as a pointer to the vendor slot on the Purchase Order Instance. The Purchase Order instance is the topic of this page, so by default, it is also the topic of this control. The ONAPPLY event is executed when the APPLY or OK button is pressed, and will execute a transaction to modify the value of the vendor with the vendor that the user selected via the combobox.

```

COMBOBOX

NAME "Vendor"

LOCATION 55, 5, 90, 11

TYPE "RPS_INSTANCE"

MENUPOPULATE TRANSACTION "form_get_instances_of_class

(:class_name \"Vendor\") "

VALUEPOPULATE PATH "vendor"

```

```
ONAPPLY{TRANSACTION "ui_modify_purchase_order_vendor
(:purchase_order %[] .Topic
:data %[] .Value)"};
```

These STATIC and EDIT control statements display the purchase order's order date.

This STATIC control statement displays the label "Order Date:" on the page.

```
STATIC

LABEL "Order Date:"

LOCATION 5, 20, 50, 11;
```

This EDIT control statement displays the purchase order's order date. The TYPE property of this edit field is date. This VALUEPOPULATE property populates this field with the value of the order_date slot on the purchase order instance. The ONAPPLY event is executed when the APPLY or OK button is pressed, and will execute the transaction to modify the purchase order's order date.

```
EDIT

NAME "Order Date"

LOCATION 55, 20, 70, 11

TYPE "RPS_DATE"

VALUEPOPULATE PATH "order_date"

ONAPPLY{TRANSACTION "ui_modify_purchase_order_order_date
(:purchase_order %[] .Topic
:data %[] .Value)"};

ENDPAGE;
```

The BEGINPAGE statement defines the beginning of the Lines page on the Purchase Order component. The ENDPAGE statement, later in the code, defines the end of the Lines page. The LABEL property places the label "Lines" on this page's tab.

```
BEGINPAGE

LABEL "Lines";
```

This PUSHBUTTON control statement creates a button with the label "Add". The ONPRESS event loads the page specified in the LOADFORM command when this button is pressed.

```
PUSHBUTTON

LOCATION 5, 5, 50, 14

LABEL "Add"
```

```
ONPRESS{LOADFORM("AddPurchaseOrderLine", :Passed_PO %[] .Topic)}
```

This PUSHBUTTON control statement creates a button with the label "Sort". The ONPRESS event is executed when this button is pressed, and refreshes the grid named "Purchase Order Line". This grid is defined in the code following the button.

```
PUSHBUTTON

LABEL "Sort"

LOCATION 430, 5, 50, 14

ONPRESS {METHOD %[Purchase Order Line].Refresh};
```

The BEGINGRID statement defines the beginning of a grid on the page, or Line page in the component. The ENDGRID statement, later in the code, defines the end of the grid. The grid is named Purchase Order Line. This VALUEPOPULATE property executes a transaction that populates this grid with the list of instances from the purchase_order_lines slot on the purchase order instance, and uses PATH to subscribe to the instances. These instances are sorted in ascending order according to the order of the columns in the grid, which can be rearranged. Each COLUMN control statement defines a column of values to be displayed.



For a description of subscribe, refer to VALUPOPULATE in Writing Statement Properties.

```
BEGINGRID

NAME "Purchase Order Line"

TOPICCLASS "Purchase_Order_Line"

LOCATION 5, 24, 475, 110

VALUEPOPULATE TRANSACTION "form_sort_slot_instance_list

(:topic_object %[] .Topic

:slot_name \"purchase_order_lines\"

:orderby \"%[] .ColumnOrder\") "

PATH "purchase_order_lines";
```

This COLUMN control statement creates a column with the LABEL "Line*". The "*" indicates to the user that this column can be sorted on; this column has a SORT property. The WIDTH property defines how wide the column is. This column is of TYPE instance. The NOEDIT property means that the user cannot change the value of this column on this page. The RIGHT property means the value of this column is displayed right-justified. The SORT property defines that this column uses the value of the line_number slot on the purchase order line instance for sorting. This VALUEPOPULATE property executes a transaction to populate this column with the actual purchase order line instance, using the line_number slot as the display value.

```

COLUMN

    LABEL "Line*"

    WIDTH "40"

    TYPE "RPS_INSTANCE"

    NOEDIT

    RIGHT

    SORT "line_number"

    VALUEPOPULATE TRANSACTION "form_get_topic_instance

        (:topic_object %[] .RowTopic

        :display_slotname \"line_number\"

        :display_slottype \"INT\")";

```

This COLUMN statement creates a column with the LABEL "Item Unit*". The "*" indicates to the user that this column can be sorted on; this column has a SORT property. The WIDTH property defines how wide the column is. This column is of TYPE instance. The NOEDIT property means that the value of this column is not changeable by the user on this page. The SORT property defines that this column sorts by using the value of the display_name slot on the site (unit) instance, which is a slot on the part (item) instance, which is a slot on the purchase order line instance. This VALUEPOPULATE property populates this column with the site instance, which is a slot on the part instance, which is a slot on the purchase order line instance.

```

COLUMN

    LABEL "Item Unit*"

    WIDTH "45"

    TYPE "RPS_INSTANCE"

    NOEDIT

    SORT "part.site.display_name"

    VALUEPOPULATE PATH "part.site";

```

This COLUMN statement creates a column with the LABEL "Item*", with the "*" indicating this column can be sorted on. The WIDTH property defines how wide the column is. This column is of TYPE instance. The NOEDIT property means that the value of this column is not changeable by the user on this page. The SORT property defines that this column sorts by using the value of the display_name, which is a slot on the part (item) instance, which is a slot on the purchase order line instance. This VALUEPOPULATE property populates this column with the part instance, which is a slot on the purchase order line instance.

```

COLUMN

    LABEL "Item*"

    WIDTH "135"

    TYPE "RPS_INSTANCE"

    NOEDIT

    SORT "part.display_name"

    VALUEPOPULATE PATH "part";

```

This COLUMN statement creates a column with the LABEL "UOM*", with the "*" indicating this column can be sorted on. The WIDTH property defines how wide the column is. This column is of TYPE string. The NOEDIT property means that the value of this column is not changeable by the user on this page. The SORT property defines that this column sorts by using the value of the uom, which is a slot on the part (item) instance, which is a slot on the purchase order line instance. This VALUEPOPULATE property populates this column with the uom, which is a slot on the part instance, which is a slot on the purchase order line instance.

```

COLUMN

    LABEL "UOM*"

    WIDTH "25"

    TYPE "RPS_STRING"

    NOEDIT

    SORT "part.uom"

    VALUEPOPULATE PATH "part.uom";

```

This COLUMN statement creates a column with the LABEL "Qty Ordered*", with the "*" indicating this column can be sorted on. The WIDTH property defines how wide the column is. This column is of TYPE float. The RIGHT property means that the value of this column is displayed right-justified. The SORT property defines that this column sorts by using the value of the total_quantity slot on the purchase order line instance. This VALUEPOPULATE property populates this column with the total_quantity, which is a slot on the purchase order line instance. The ONAPPLY event is executed when the APPLY or OK button is pressed, and executes the transaction to modify the value of the purchase order's total quantity.

```

COLUMN

    LABEL "Qty Ordered*"

    WIDTH "50"

    TYPE "RPS_FLOAT"

    RIGHT

```

```

SORT "total_quantity"

VALUEPOPULATE PATH "total_quantity"

ONAPPLY{TRANSACTION "ui_modify_po_line_total_quantity

    (:purchase_order_line %[] .RowTopic

    :data %[] .Value)"};

```

This COLUMN statement creates a column with the LABEL "Qty Released*", with the "*" indicating this column can be sorted on. The WIDTH property defines how wide the column is. This column is of TYPE float. The NOEDIT property means that the value of this column is not changeable by the user on this page. The SORT property defines that this column sorts by using the value of the released_quantity, which is a slot on the purchase order line instance. This VALUEPOPULATE property populates this column with the released_quantity, which is a slot on the purchase order line instance.

```

COLUMN

    LABEL "Qty Released*"

    WIDTH "50"

    TYPE "RPS_FLOAT"

    NOEDIT

    RIGHT

    SORT "released_quantity"

    VALUEPOPULATE PATH "released_quantity";

```

This COLUMN statement creates a column with the LABEL "Unrel'd Qty*", with the "*" indicating this column can be sorted on. The WIDTH property defines how wide the column is. This column is of TYPE float. The NOEDIT property means that the value of this column is not changeable by the user on this page. The SORT property defines that this column sorts by using the value of the unreleased_quantity, which is a slot on the purchase order line instance. This VALUEPOPULATE property populates this column with the unreleased_quantity, which is a slot on the purchase order line instance.

```

COLUMN

    LABEL "Unrel'd Qty*"

    WIDTH "50"

    TYPE "RPS_FLOAT"

    NOEDIT

    RIGHT

    SORT "unreleased_quantity"

```

```
VALUEPOPULATE PATH "unreleased_quantity";
```

This COLUMN statement creates a column with the LABEL "Unit Price*", with the "*" indicating this column can be sorted on. The WIDTH property defines how wide the column is. This column is of TYPE float. The RIGHT property means that the value of this column is displayed right-justified. The SORT property defines that this column sorts by using the value of the unit_price, which is a slot on the purchase order line instance. This VALUEPOPULATE property populates this column with the unit_price, which is a slot on the purchase order line instance. The ONAPPLY event is executed when the APPLY or OK button is pressed, and it executes the transaction to modify the value of the purchase order's unit price.

```
COLUMN

    LABEL "Unit Price*"

    WIDTH "50"

    TYPE "RPS_FLOAT"

    RIGHT

    SORT "unit_price"

    VALUEPOPULATE PATH "unit_price"

    ONAPPLY{TRANSACTION "ui_modify_po_line_unit_price

        (:purchase_order_line %[] .RowTopic

        :data %[] .Value)"};

ENDGRID;

ENDPAGE;

ENDPROPSHEET;
```

[SETMENU] defines the section for the set menu for this class.

```
[SETMENU]
```

The BEGINMENU statement defines the beginning of the set menu for the Purchase Order class. The ENDMENU statement, later in the code, defines the end of the menu.

```
BEGINMENU

    LABEL "Purchase Order";
```

This MENUITEM control statement creates a menu item on this menu with the LABEL "Add". The ONPICK event is executed when this menu item is selected, and it executes the LOADFORM command to load the page to add a purchase order.

```
MENUITEM

    LABEL "Add"
```



```
ONPICK{LOADFORM("AddPurchaseOrder")};
```

This SEPARATOR control statement places a separator—a horizontal grey line—on the menu. A separator helps to group menu items.

```
SEPARATOR;
```

This MENUITEM control statement creates a menu item on this menu with the LABEL "Find". The ONPICK event is executed when this menu item is selected, and will execute the LOADFORM command to load the purchase order filter page.

```
MENUITEM

    LABEL "Find"

    ALWAYSEDIT

    ONPICK{LOADFORM("PurchaseOrderFilter")};

ENDMENU;
```

The [INSTANCEMENU] defines the section for the instance menu of this class.

```
[INSTANCEMENU]
```

The BEGINMENU statement defines the beginning of the instance menu for the Purchase Order class. The ENDMENU statement, later in the code, defines the end of the menu.

```
BEGINMENU

    LABEL "Purchase Order";
```

This MENUITEM control statement creates a menu item on this menu with the LABEL "Delete". The ONPICK event is executed when this menu item is selected, and it executes the two commands specified. The ASKUSER command creates a pop-up window to ask the user "Delete Purchase Order?". If the user clicks **OK**, the TRANSACTION command executes the transaction to delete the selected purchase order. If the user clicks **Cancel**, that delete transaction is not executed.

```
MENUITEM

    LABEL "Delete"

    ONPICK{ASKUSER "Delete Purchase Order?",

        TRANSACTION "ui_delete_po(:purchase_order %[].PopupMenuTopic)"};
```

This MENUITEM control statement creates a menu item on this menu with the LABEL "Properties". The ONPICK event is executed when this menu item is selected, and it executes the LOADPROPSHEET command to load the purchase order property sheet page. The :ParentsTopic %[].PopupMenuTopic passes the purchase order property sheet page the purchase order instance, allowing it to have that instance for its ParentsTopic parameter.

```
MENUITEM  
  
    LABEL "Properties"  
  
    ONPICK{LOADPROPSHEET ("Purchase_Order", :ParentsTopic %[].PopupMenuTopic)};  
  
ENDMENU;
```

Understanding the Inventory Report Page Code

Here is the Inventory Report page. Following it is the code that creates it, from the InventoryReport file.

To display this page, perform the following steps:

1. Start the Planning Client. The Home Base page appears.
2. Click the Browse Data button. The Data Browser page appears.
3. Click + Items. Inventory Items, Products, Planning Items, and Phantom Items appear below it and to the right.
4. Click **Inventory Items**. A list of parts appears on the Data Browser page.
5. Right-click an item name. A pop-up menu appears, with **Inventory Report** as one of the menu items.
6. Click **Inventory Report**. The Inventory Report page appears.

Inventory Report page

The BEGINFORM statement defines the beginning of the Inventory Report page. The ENDFORM statement, later in the code, defines the end of the page. The LABEL property places the label "Inventory Report" at the top of the page. The TOPICPOPULATE property populates the page with the passed parameter Passed_Part, which is a part (item) instance.

```
[FORM]
```

```
BEGINFORM
```

```
    LABEL "Inventory Report"
```

```
    TOPICPOPULATE PARAMETER :Passed_Part;
```

The BEGINCTC statement defines the beginning of this coordinated time control widget. The ENDCTC statement, later in the code, defines the end of this coordinated time control widget. The TOPICPOPULATE property executes a transaction to get the main environment. The VALUEPOPULATE property executes a transaction to get the time fences for this environment.

```
BEGINCTC
```

```
    LOCATION 5, 5, 520, 240
```

```
    TOPICPOPULATE TRANSACTION "get_oid main_environment;"
```

```
    VALUEPOPULATE TRANSACTION "get_project_times(:env_name  
    \"main_environment\")";
```

The TASK_GANTT control statement defines a task Gantt with the label "Inventory Report Gantt". The LABELHEIGHT property defines the height of this label. The PERCENTHEIGHT property defines what percentage of this ctc this Gantt will use. The VALUEPOPULATE property executes a transaction to get the inventory information used to populate this Gantt chart, and uses PATH to subscribe to this information.



For a description of subscribe, refer to VALUPOPULATE in Writing Statement Properties.

```
TASK_GANTT

NAME "Inventory_Report_Gantt"

LABEL "Inventory Report Gantt"

LABELHEIGHT 12

PERCENTHEIGHT 50

VALUEPOPULATE TRANSACTION "ui_inventory_report_data(:part %:Passed_Part)"

PATH "resource_history";
```

This INVENTORY_HISTOGRAM control statement defines a histogram with the label "Inventory". The LABELHEIGHT property defines the height of this label. The PERCENTHEIGHT property defines what percentage of this ctc that this histogram will use. This VALUEPOPULATE property uses the passed parameter Passed_Part—which is a part (item) instance—to populate the histogram.

```
INVENTORY_HISTOGRAM

LABEL "Inventory"

LABELHEIGHT 12

PERCENTHEIGHT 50

VALUEPOPULATE PARAMETER :Passed_Part;

ENDCTC;

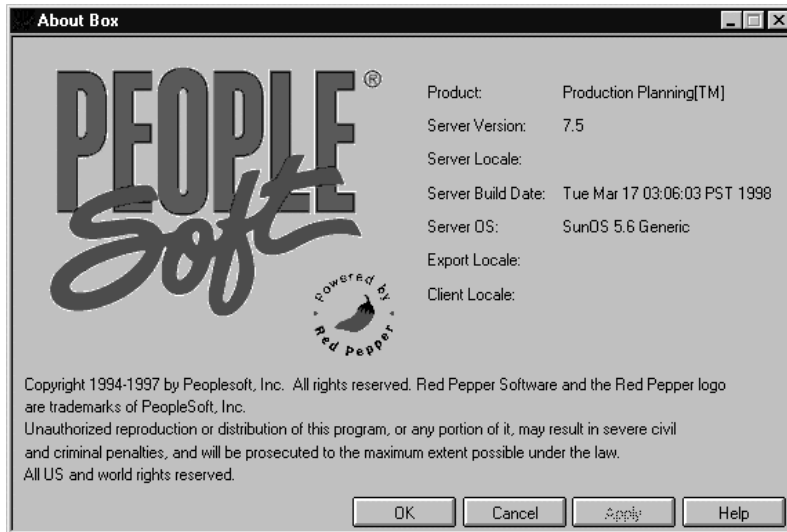
ENDFORM;
```

Understanding the About Box Page Code

Here is the About Box page. Following it is the code that creates it, from the AboutBox file.

To display this page, perform the following steps:

1. Start the Planning Client.
2. Select **Help, About**. The About Box page appears.



About Box page

The BEGINFORM statement defines the beginning of the About Box page. The ENDFORM statement, later in the code, defines the end of the page. The LABEL property places the label "Inventory Report" at the top of the page. The TOPICPOPULATE property executes a transaction to create a temporary page backing object. This object is created to use as the topic of this page, and contains data to be displayed in this page. The ONCLOSE event deletes the temporary page backing object.

```
[FORM]

BEGINFORM

    LABEL "About Box"

    TOPICPOPULATE TRANSACTION "transaction_get_form_topic (:form_topic_class_name
\ "About_Box\ ") "

    ONCLOSE TRANSACTION "transaction_release_form_topic (:form_topic
% [.FormTopic) ";
```

The BITMAP statement places a bitmap on the page. The FILE property uses the file ABOUTBOX.BMP for the bitmap to place on this page.

```
BITMAP

    FILE "ABOUTBOX.BMP"

    LOCATION 5, 5, 190, 140;

STATIC
```

```
    LABEL "Product:"

    LOCATION 200, 20, 70, 11;

STATIC

    LOCATION 265, 20, 110, 11

    TYPE "RPS_STRING"

    NOEDIT

    VALUEPOPULATE PATH "product_name";

STATIC

    LABEL "Server Version:"

    LOCATION 200, 35, 70, 11;

STATIC

    LOCATION 265, 35, 175, 11

    TYPE "RPS_STRING"

    NOEDIT

    VALUEPOPULATE PATH "server_version";
```

The remainder of the code up to the ENDFORM statement for the About Box is not shown. It contains STATIC and EDIT control statements, which have examples elsewhere in this section.

```
ENDFORM;
```

Understanding the Item Filter Page Code

Here is the Item Filter page. Following it is the code that creates it, from the PartFilter file.

To display this page, perform the following steps:

1. Start the Planning Client. The Home Base page appears.
2. Click **Capable to Promise**. The Capable to Promise page appears.
3. Click **Select...** The Item Filter page appears. It will be blank.

Find: Item

Find Now Unit Filter: Planner Code Filter: Sort

Select Item Filter: Item Category 1 Filter: Item Category 2 Filter: MPS Filter: ☐

Description Filter: Item Class: Product

	Unit*	Item*	Description*	Item Class	Planner Code*	Item Category 1
1	Region1	Blue_Berry_Bicycle	Blue Berry Bicycle	Product	davis	Blue
2	Region1	Double_Blue_Tandem_Bike	Double Blue Tandem Bike	Product	jones	Blue
3	Region1	Double_Green_Tandem_Bike	Double Green Tandem Bike	Product	jones	Green
4	Region1	Double_Red_Tandem_Bike	Double Red Tandem Bike	Product	jones	Red
5	Region1	Green_Cucumber_Bicycle	Green Cucumber Bicycle	Product	davis	Green
6	Region1	Red_Pepper_Bicycle	Red Pepper Bicycle	Product	davis	Red

OK Cancel Apply Help

Item Filter page

In order to select an Item for the Capable to Promise page, perform the following steps:

1. Click **Find Now** on the Item Filter page. The grid on the page is filled in.
2. Double-click one of the rows; they are labeled "1", "2", "3", and so on in the page. The appropriate item page appears. The item page that appears depends upon the type of item; for example, it could be an inventory item or a product.
3. Click **OK** on the item page. The page closes.
4. Click **OK** on the Item Filter page. The page closes, and the Capable to Promise page changes as follows:
 - The Customer Unit (Site) EDIT field—named SelectSite in the CapableToPromise file—is filled in with the site display_name of the selected product.
 - The Item (Part) EDIT field—named SelectPart in the CapableToPromise file—is filled in with the display_name of the selected product.

The BEGINFORM statement defines the beginning of the Item Filter page. The ENDFORM statement, later in the code, defines the end of the page. The LABEL property places the label "Find: item" at the top of the page. The ONOK uses the SENDPARENT command to send the Parts (Items) topic to the Capable To Promise page, which is the parent page for the Item Filter page. The Capable to Promise page has an EDIT control named SelectSite and an EDIT control named SelectPart. The SENDPARENT command sends the topic of Parts—a COLUMN control on this page—to these EDIT controls on the Capable to Promise page, populating those EDIT controls. These EDIT controls use the NOAUTOPOPULATE property to not be populated until the SENDPARENT command sends them their topic.

[FORM]

BEGINFORM

LABEL "Find: Item"

```
ONOK {SENDPARENT(%[Parts].SelectedRow %[SelectSite].TopicForceApply),
      SENDPARENT(%[Parts].SelectedRow %[SelectPart].TopicForceApply)};
```

The PUSHBUTTON statement's LABEL property places the label "Find Now" on this button. The ONPRESS event causes the Select PUSHBUTTON to be enabled, and this button to be disabled, and refreshes the GRID control named "Parts".

```
PUSHBUTTON

NAME "Find"

LABEL "Find Now"

LOCATION 5, 5, 50, 14

ONPRESS {METHOD %[Parts].Refresh,
         METHOD %[Select].Enable(),
         METHOD %[Find].Disable()};
```

The PUSHBUTTON statement's LABEL property places the label "Select" on this button. The ONPRESS uses the SENDPARENT command to send the Parts (Items) topic to the Capable To Promise page, which is the parent page for the Item Filter page. This SENDPARENT behaves the same as the one in the BEGINFORM statement. The Capable to Promise page has an EDIT control named SelectSite and an EDIT control named SelectPart. The SENDPARENT command sends the topic of Parts—a GRID control on this page—to these EDIT controls on the Capable to Promise page, populating those EDIT controls. The DISABLED property disables this button until the FIND button is pressed.

```
PUSHBUTTON

NAME "Select"

LABEL "Select"

LOCATION 5, 23, 50, 14

DISABLED

ONPRESS {SENDPARENT(%[Parts].SelectedRow %[SelectSite].TopicForceApply),
         SENDPARENT(%[Parts].SelectedRow %[SelectPart].TopicForceApply)};
```

```
STATIC

LABEL "Unit Filter:"

LOCATION 70, 5, 65, 11;

EDIT

NAME "Site Filter"
```



```
LOCATION 135 5, 45, 11

TYPE "RPS_STRING"

ALWAYSEEDIT

ONCHANGE {METHOD %[Find].Enable(), METHOD %[Select].Disable()}

NOAPPLY;
```

STATIC

```
LABEL "Item Filter:"

LOCATION 70, 20, 65, 11;
```

EDIT

```
NAME "Part Filter"

LOCATION 135, 20, 135, 11

TYPE "RPS_STRING"

ALWAYSEEDIT

ONCHANGE {METHOD %[Find].Enable(), METHOD %[Select].Disable()}

NOAPPLY;
```

STATIC

```
LABEL "Description Filter:"

LOCATION 70, 35, 65, 11;
```

EDIT

```
NAME "Description Filter"

LOCATION 135, 35, 135, 11

TYPE "RPS_STRING"

ALWAYSEEDIT

ONCHANGE {METHOD %[Find].Enable(), METHOD %[Select].Disable()}

NOAPPLY;
```

STATIC

```
    LABEL "Item Class:"

    LOCATION 70, 50, 65, 11;

COMBOBOX

    NAME "Part Class"

    LOCATION 135, 50, 115, 11

    TYPE "RPS_CLASS"

    ALWAYSEDIT

    MENUPOPULATE TRANSACTION "ui_get_part_classes()"

    VALUEPOPULATE TRANSACTION "get_oid \"Base_Part\";"

    ONCHANGE {METHOD %[Find].Enable(), METHOD %[Select].Disable()}

    NOAPPLY;

STATIC

    LABEL "Planner Code Filter:"

    LOCATION 280, 5, 90, 11;

EDIT

    NAME "Planner Code Filter"

    LOCATION 370, 5, 45, 11

    TYPE "RPS_STRING"

    ALWAYSEDIT

    ONCHANGE {METHOD %[Find].Enable(), METHOD %[Select].Disable()}

    NOAPPLY;

STATIC

    LABEL "Item Category 1 Filter:"

    LOCATION 280, 20, 90, 11;

EDIT

    NAME "Part Category1 Filter"

    LOCATION 370, 20, 70, 11
```

```
TYPE "RPS_STRING"

ALWAYSEDIT

ONCHANGE {METHOD %[Find].Enable(), METHOD %[Select].Disable()}

NOAPPLY;

STATIC

    LABEL "Item Category 2 Filter:"

    LOCATION 280, 35, 140, 11;

EDIT

    NAME "Part Category2 Filter"

    LOCATION 370, 35, 70, 11

    TYPE "RPS_STRING"

    ALWAYSEDIT

    ONCHANGE {METHOD %[Find].Enable(), METHOD %[Select].Disable()}

    NOAPPLY;

STATIC

    LABEL "MPS Filter:"

    LOCATION 280, 50, 90, 11;

CHECKBOX

    NAME "MPS Filter"

    LOCATION 370, 50, 25, 11

    TYPE "RPS_INT"

    ALWAYSEDIT

    ONCHANGE {METHOD %[Find].Enable(), METHOD %[Select].Disable()}

    NOAPPLY;

PUSHBUTTON

    LABEL "Sort"
```

```
LOCATION 410, 65, 50, 14
```

```
ALWAYSEEDIT
```

```
ONPRESS {METHOD %[Parts].Refresh};
```

The BEGINGRID control is named "Parts". The TOPICPOPULATE QUERY gets the Part (Item) class, and populates each row with the instances of that class. It gets slot values for each Part instance, such as site.display_name, display name, description, and part category filters. The ColumnOrder method allows the columns in this grid to be rearranged by the user, and sorts the columns in ascending order according to the order of the columns in the grid. NOAUTOPOPULATE causes this grid to not be populated until the user presses the "Find Now" button, which contains a Refresh method for this grid.

```
BEGINGRID
```

```
NAME "Parts"
```

```
LOCATION 5, 80, 450, 110
```

```
NOAUTOPOPULATE
```

```
TOPICPOPULATE QUERY "select oid(%[Part Class].Value) where
    site.display_name ~ %[Site Filter].Value and
    display_name ~ %[Part Filter].Value and
    description ~ %[Description Filter].Value and
    planner_code ~ %[Planner Code Filter].Value and
    (%[MPS Filter].Value = 0 or
    ui_part_mps_filter(:part %input
        :mps_type_filter %[MPS Filter].Value
        :found %output)) and
    (%[Part Category1 Filter].Value = \"\" or
    ui_part_category_1_filter(:part %input
        :part_category_1_filter %[Part Category1 Filter].Value
        :found %output)) and
    (%[Part Category2 Filter].Value = \"\" or
    ui_part_category_2_filter(:part %input
        :part_category_2_filter %[Part Category2 Filter].Value
        :found %output))
```

```
order by %[Parts].ColumnOrder;;
```

This column is populated by the site.display_name slot values in the Part instances.

```
COLUMN

NAME "Site"

LABEL "Unit*"

WIDTH "45"

TYPE "RPS_INSTANCE"

NOEDIT

SORT "site.display_name"

VALUEPOPULATE PATH "site";
```

This column is populated by the display_name slot values in the Part instances.

```
COLUMN

NAME "Part"

LABEL "Item*"

WIDTH "135"

TYPE "RPS_INSTANCE"

NOEDIT

SORT "display_name"

VALUEPOPULATE PATH ".";
```

This column is populated by the description slot values in the Part (Item) instances.

```
COLUMN

NAME "Description"

LABEL "Description*"

WIDTH "135"

TYPE "RPS_STRING"

SORT "description"

VALUEPOPULATE PATH "description"

ONAPPLY{TRANSACTION "ui_modify_part_description(:part %[.RowTopic

:data %[.Value)"};
```

This column is populated by the `form_get_class_of_instance` transaction. This transaction gets the class names that are displayed in this column.



For a description of the transaction, refer to `form_get_class_of_instance`.

COLUMN

NAME "Part Class"

LABEL "Item Class"

WIDTH "110"

TYPE "RPS_CLASS"

NOEDIT

VALUEPOPULATE TRANSACTION "form_get_class_of_instance(:topic_object
%[] .RowTopic)";

COLUMN

NAME "Planner Code"

LABEL "Planner Code*"

WIDTH "65"

TYPE "RPS_STRING"

SORT "planner_code"

VALUEPOPULATE PATH "planner_code"

ONAPPLY{TRANSACTION "ui_modify_part_planner_code(:part %[] .RowTopic
:data %[] .Value)"};

COLUMN

NAME "Part Category 1"

LABEL "Item Category 1"

WIDTH "65"

TYPE "RPS_INSTANCE"

NOEDIT

```

VALUEPOPULATE TRANSACTION "ui_get_part_category_1(:part %[] .RowTopic)";

COLUMN

NAME "Part Category 2"

LABEL "Item Category 2"

WIDTH "65"

TYPE "RPS_INSTANCE"

NOEDIT

VALUEPOPULATE TRANSACTION "ui_get_part_category_2(:part %[] .RowTopic)";

COLUMN

NAME "MPS"

LABEL "MPS"

WIDTH "45"

TYPE "RPS_INT"

WIDGET "CHECKBOX"

VALUEPOPULATE TRANSACTION "ui_get_part_mps_type(:part %[] .RowTopic)"

ONAPPLY{TRANSACTION "ui_modify_part_mps_type(:part %[] .RowTopic
      :data %[] .Value)"};

ENDGRID;

ENDFORM;

```

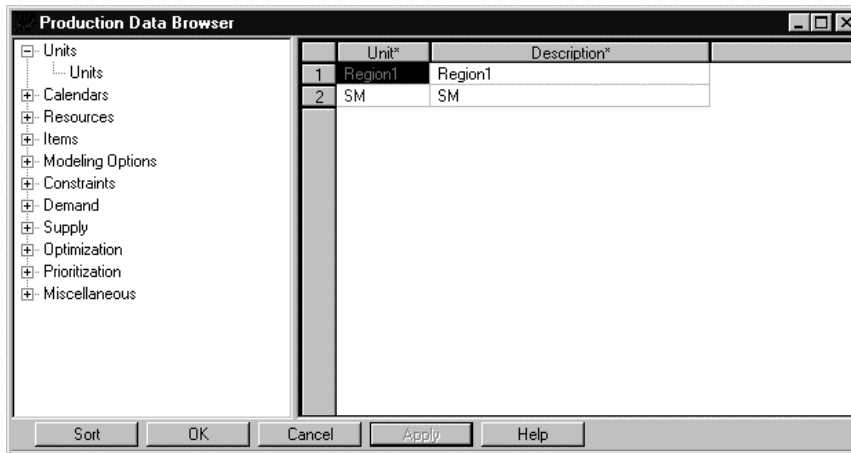
Understanding the Data Browser Page Code

Here is the Data Browser page. Following it is the code that creates it, from the PRA_ObjectBrowser file.

To display the data browser, do the following:

1. Start the Planning Client. The Home Base page appears.

2. Click **Browse Data**. The Data Browser page appears.



Data Browser

The BEGINBROWSER statement defines the beginning of the data browser. The ENDBROWSER statement, later in the code, defines the end of the data browser. The LABEL property places the label "Production Data Browser" at the top of the page.

```
[OBJECTBROWSER]

BEGINBROWSER

    LABEL "Production Data Browser";
```

The BEGINFOLDER statement defines the beginning of the folder that contains information for a set of classes: in this case, the set contains only the site (unit) class. The ENDFOLDER statement, later in the code, defines the end of the folder. The LABEL property places the label "Units" on the left side of the page, in the list of folder names.

```
BEGINFOLDER

    LABEL "Units";
```

The BEGINGRID statement defines the beginning of the grid to display the site (unit) class. The ENDGRID statement, later in the code, defines the end of the grid. The LABEL property places the label "Units" to the right and just below the folder name, where it appears when the user clicks the folder label. TOPICCLASS displays the instances of the Site class.

```
BEGINGRID

    NAME "Sites"

    LABEL "Units"

    TOPICCLASS "Site";
```

The following columns display the unit's (site's) display_name and description. VALUEPOPULATE PATH "." in the first COLUMN has the COLUMN display the

display_name for the site instance. VALUEPOPULATE PATH "description" in the second COLUMN has the COLUMN display the description slot for the site instance.

```
COLUMN
```

```
NAME "Site"
```

```
LABEL "Unit*"
```

```
WIDTH "45"
```

```
TYPE "RPS_INSTANCE"
```

```
NOEDIT
```

```
SORT "display_name"
```

```
VALUEPOPULATE PATH ".";
```

```
COLUMN
```

```
NAME "Description"
```

```
LABEL "Description*"
```

```
WIDTH "135"
```

```
TYPE "RPS_STRING"
```

```
SORT "description"
```

```
VALUEPOPULATE PATH "description"
```

```
ONAPPLY{TRANSACTION "form_set_slot_string(:object %[] .RowTopic
```

```
                                :slot_name \"description\"
```

```
                                :value %[] .Value)"};
```

```
ENDGRID;
```

```
ENDFOLDER;
```

The remaining code for the data browser up to the ENDBROWSDER statement is not shown. It contains BEGINFOLDER and ENDFOLDER control statements, which behave similarly to the previous BEGINFOLDER and ENDFOLDER.

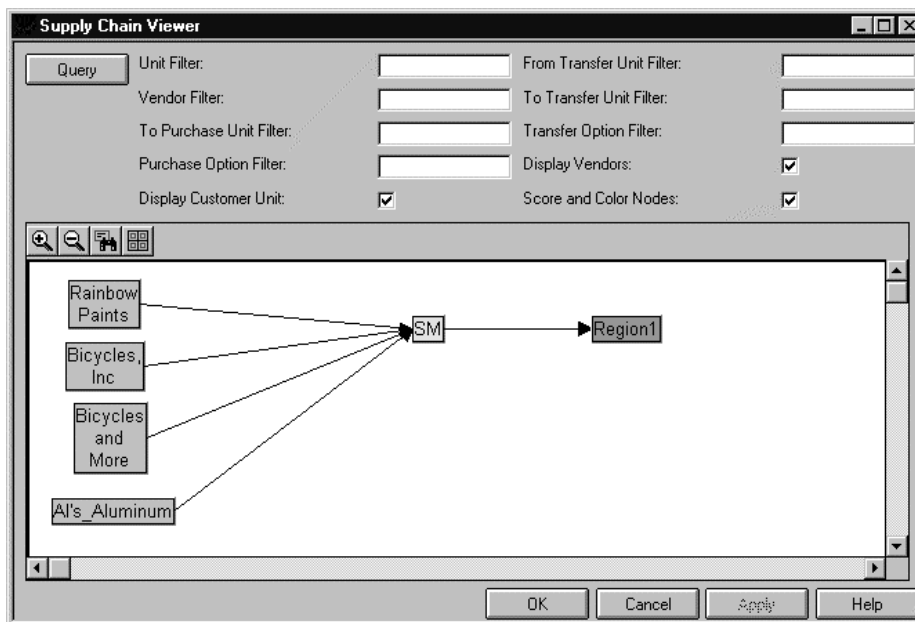
```
ENDBROWSER;
```

Understanding the Supply Chain Viewer Page Code

Here is the Supply Chain Viewer page. Following it is the code that creates it, from the SupplyChainViewer file.

To display the supply chain viewer, do the following:

1. Start the Planning Client. The Home Base page appears.
2. Select **File, Supply Chain Viewer**. The Supply Chain Viewer page appears.
3. To load data into the viewer, as shown below, click **Query**.



Supply Chain Viewer

The BEGINFORM statement defines the beginning of the Supply Chain Viewer page. The ENDFORM statement, later in the code, defines the end of the page. The LABEL property placed the label “Supply Chain Viewer” at the top of the page. The topic of this page is the SCV_Form_Window class. This class supplies the information that is needed to populate this page.

```
[FORM]
```

```
BEGINFORM
```

```
  LABEL "Supply Chain Viewer"
```

```
  TOPICPOPULATE TRANSACTION "transaction_get_form_topic(
    :form_topic_class_name \"SCV_Form_Window\")"
```

```
  ONCLOSE TRANSACTION "transaction_release_form_topic(
```

```
:form_topic %[].FormTopic)";
```

This STATIC control displays the label “Unit Filter” on the page.

```
STATIC
```

```
  LABEL "Unit Filter:"
```

```
  LOCATION 60, 5, 54, 11;
```

This EDIT control statement displays an edit field where the user enters a site (unit) filter. The TYPE of this EDIT is a character string. This EDIT field is populated by the site_filter slot on the temporary page backing object, which is from the SCV_Form_Window class. NOAPPLY means that if the user makes changes to this field, the Apply button on the Supply Chain Viewer page is not enabled.

```
EDIT
```

```
  NAME "Site Filter"
```

```
  LOCATION 175, 5, 65, 11
```

```
  TYPE "RPS_STRING"
```

```
  VALUEPOPULATE PATH "site_filter"
```

```
  NOAPPLY;
```

The rest of the controls used to display Vendor Filter, To Purchase Unit Filter, and so on, are similar to the STATIC and EDIT control statements shown above, so they are deleted from this example.

The Query button refreshes the information shown in the diagramming control in the Supply Chain Filter page, using the information that was entered in the EDITs and CHECKBOXes.

```
PUSHBUTTON
```

```
  LABEL "Query"
```

```
  LOCATION 5, 5, 50, 14
```

```
  ONPRESS {METHOD %[SCV Diagram].Refresh};
```

The following BEGINDIAGRAM statement defines the beginning of the diagramming control for the Supply Chain Viewer page. The VALUEPOPULATE TRANSACTION uses the ui_scv_diagram_control_display transaction to get the information for this page. This transaction uses the inputs entered on this page, such as site_filter and vendor_filter, to display nodes and arcs in the diagramming control.

```
BEGINDIAGRAM
```

```
  NAME "SCV Diagram"
```

```

LOCATION 5, 80, 430, 160

NOAUTOPOPULATE

VALUEPOPULATE TRANSACTION "ui_scv_diagram_control_display(
    :scv_form_window %[] .FormTopic
    :site_filter  %[Site Filter].Value
    :vendor_filter %[Vendor Filter].Value
    :to_purchase_site_filter %[To Purchase Site Filter].Value
    :purchase_option_filter %[Purchase Option Filter].Value
    :display_customer_site %[Display Customer Site].Value
    :from_transfer_site_filter %[From Transfer Site Filter].Value
    :to_transfer_site_filter %[To Transfer Site Filter].Value
    :transfer_option_filter %[Transfer Option Filter].Value
    :score_nodes %[Score Nodes].Value
    :display_vendors %[Display Vendors].Value)"

```

The drawing area is defined here.

```

DRAWINGAREA

GRIDHEIGHT "100"

GRIDWIDTH "100"

```

The node is defined here. It uses mostly default settings.

```

NODE

FRAME "HEAVYINDENT";

```

The arc is not defined, so it uses default settings.

```

ENDDIAGRAM;

ENDFORM;

```

The instance menus for arcs and nodes are contained in the property sheet pages for the node or arc. In the case of the Supply Chain Viewer, the node class has two subclasses: vendor node and unit node, and the arc class has two subclasses: transfer order node and purchase order node. The property sheet pages for these subclasses appear when you double-click a node or arc, and the

instance menu for these property sheet pages appear when you right-click a node or arc. This is the behavior for a property sheet page; when an instance of a class is double-clicked, the property sheet page for that class appears, containing that instance.

The property sheet pages for the arcs are `Purchase_Arc` and `Transfer_Arc`; the property sheet pages for the nodes are `Site_Node` (for unit nodes) and `Vendor_Node`. You can make this property sheet page appear by double-clicking a Unit node in the Supply Chain Viewer.

`Vendor_Node` is a class, so the `Vendor_Node` property sheet page appears when you double-click a vendor node in the Supply Chain Viewer. The following page appears when you double-click the Rainbow Paints vendor node shown in the Supply Chain Viewer page.



Vendor Node page

Following is the code for `Vendor_Node`.

`Vendor_Node` is listed as a class and as a property sheet page in the catalog file. It also has its class listed in the code below. Most of the code is similar to previous examples.

```
[CLASS]

BEGINCLASS

    NAME "Vendor_Node"

    LABEL "Vendor Node";

ENDCLASS;

[PROPERTY SHEET]

BEGINPROPSHEET

    LABEL "Vendor Node";

    STATIC

        LABEL "Vendor:"

        LOCATION 5, 5, 41, 11;

    EDIT
```

```

NAME "Vendor"

LOCATION 65, 5, 110, 11

TYPE "RPS_INSTANCE"

NOEDIT

VALUEPOPULATE PATH "vendor";

STATIC

    LABEL "Description:"

    LOCATION 5, 20, 54, 11;

EDIT

NAME "Description"

LOCATION 65, 20, 135, 11

TYPE "RPS_STRING"

VALUEPOPULATE PATH "vendor.description"

NOAPPLY;

```

```
ENDPROPSHEET;
```

The instance menu for the vendor node will show Properties; clicking **Properties** will execute LOADPROPSHEET, which will load and display this page, the Vendor Node page. If you were to define a BEGINNODEMENU in the Supply Chain Viewer code, the additions to the page would appear in this instance menu.

```

[INSTANCEMENU]

BEGINMENU

    LABEL "Vendor_Node";

MENUITEM

    LABEL "Properties"

    ONPICK{LOADPROPSHEET ("Vendor_Node", :ParentsTopic %[] .PopupMenuTopic)};

ENDMENU;

```

Understanding Pages Formatting Standards

This section explains the formatting standards that you should follow when you are creating pages. These standards help your pages to be consistent with the Planning pages.

Font

Standard font. This is the system font.

Display Target

800 x 600 display

Regional Settings Time Style

HH:mm:ss

Text Labels

Put a label on all Edit Fields and Grid Columns.

To calculate the units for an approximate label width, multiply the number of characters you wish to display by 4.5.

The label height should be 11 units.

Left justify labels of Edit Fields (default). Center justify labels of Grid Columns.

Initial capitalize all important words in the label

Put a colon (:) at the end of all prompts.

Place all column headings on top of lists. Do not put a colon after column headings.

If the Grid column can be sorted, then place a star (*) at the end of the label.

Edit Fields

To calculate the units for an approximate edit field width, multiply the number of characters you wish to display by 4.5. Add 5 units if a combo box is displayed next to the control. Add 5 units or more to offset the edit field from the text label.

The edit field height should be 11 units.

Suggested Width for Edit Fields:

Suggested Width for Edit Fields

<i>Data Type</i>	<i>Suggested Width</i>
Class	25 characters—110 units—without a combo box; 115 units with a combo box
Unit	10 characters; 45 units
Resource	20 characters—90 units—without a combo box; 95 units with a combo box
Item	30 characters—135 units—without a combo box; 140 units with a combo box
Option Name	25 characters—110 units—without a combo box; 115 units with a combo box
Task	30 characters; 135 units
Description	30 characters; 135 units
Date	17 characters; 70 units
Time	10 characters; 50 units
Quantity	10 characters—50 units—for a large number or a number with a lot of precision; 8 characters—40 units—for a small number
Enums	The number of characters in the longest enum value. Add 5 units for combo control.

Justification for Edit Fields

<i>Data Type</i>	<i>Suggested Justification</i>
String:	LEFT
Date:	LEFT
Enum:	LEFT
Integer:	RIGHT
Time:	RIGHT
Float:	RIGHT
Borders:	Defaults to 3-D Lowered
Grids:	Height: At least 5 rows in the grid; 60 units

Data Type	Suggested Justification
	Recommended 10 rows in the grid for the default size; 110 units
Width:	Sum of the column widths of each column plus 30 units
Command Buttons:	Use predefined buttons where appropriate. OK Cancel Apply Help Query Add Delete
Button Size:	14 units height, at least 50 units wide. Increase the width as appropriate to fit text in the button.

Spacing between objects on a page

Objects on page	Spacing
Top, Left location of the first item:	5 units, 5 units
Horizontal space between buttons:	5 units
Vertical space between Label fields and/or Edit Fields:	4 units
Horizontal space between a Label field and an Edit field:	5 units. Left location of the edit field is the number of the label characters multiplied by 4.5.

To line up a column of Label/Edit fields:

Left location of all edit fields to be lined up should be the maximum number of label characters multiplied by 4.5. Be sure to include for the colon ":" if one is present.

Suggested Formatting Rules:

Start every control keyword on a new line.

Indent control keywords by 2 spaces. Avoid using tabs.

Place the control keywords in the following order:

NAME
LABEL
TOPICCLASS
LOCATION

WIDTH
 TYPE
 WIDGET
 NOEDIT
 REQUIRED
 RIGHT/LEFT/CENTER
 float format
 TOPICPOPULATE
 MENUPOPULATE
 VALUEPOPULATE
 ONAPPLY
 ONOK
 ONPICK
 ONPRESS

Here is a section of code from the Site (Unit) page that shows some of these formatting rules.

```

[PROPERTY SHEET]

BEGIN PROPSHEET

  LABEL "Unit";

  STATIC

    LABEL "Unit:"

    LOCATION 5, 5, 45, 11;

  EDIT

    NAME "Site"

    LOCATION 50, 5, 45, 11

    TYPE "RPS_INSTANCE"

    NOEDIT

    VALUEPOPULATE PATH ".";
  
```

Using Standard Pages Transactions

This section documents the standard transactions that you can use when you create pages. These transactions perform one of three functions:

- Getting data from the server. For example, a transaction can get the value of a slot on a topic object, and your page can then show this data on a page.

- Changing data on the server, such as the value of a slot on the topic object. For example, a user might enter or edit data on a page, and a transaction can send this data to the server, setting the value of a slot on a topic object.
- Other miscellaneous functions, such as sorting rows in a grid.

Getting Data with Page Transactions

The following transactions get data from the server. Use these transactions to get data to populate your pages and controls.

form_get_slot_instance

Description: Gets the value of a specified slot on a given topic object. The slot must be of type instance. This action will also return the value of a user specified slot for display purposes, or if one is not specified, the value of the display_name slot. Use this with VALUEPOPULATE only when you cannot use PATH by itself to populate the control; PATH will subscribe to the slot, refreshing the displayed value if its value is changed on another page or control.



For a description of subscribe, refer to VALUEPOPULATE in Writing Statement Properties.

Inputs:

topic_object INSTANCE

The topic object.

slot_name STRING

The slot whose value is to be returned.

display_slotname STRING

The name of the slot whose value is to be returned in the value_name output variable.

display_slotype STRING

The type of the slot whose value is to be returned in the value_name output variable.

Required Inputs: topic_object, slot_name

Input Defaults:

display_slotype STRING

Outputs:

value INSTANCE

The value of the specified slot.

value_name STRING

The string value of the user specified display slot.

error_msg STRING

An error message to be returned. (Not currently used.)

Instances updated: None

form_get_topic_instance

Description: Gets the value of a given topic object. It returns the instance and the display name of a specified slot of that instance. This action is generally used when the instance whose value is desired, does not have a display_name slot. Use PATH "." when the instance has a display_name slot; PATH will subscribe to the slot, refreshing the displayed value if its value is changed on another page or control.



For a description of PATH ".", refer to VALUEPOPULATE in Writing Statement Properties.

Inputs:

topic_object INSTANCE

The topic object.

slot_name STRING

The slot whose value is to be returned. (Not used.)

display_slotname STRING

The slot whose value is to be returned as a string.

display_slottype STRING

The type of the slot whose display name is to be returned.

Required Inputs: topic_object

Input Defaults: display_slottype STRING

Outputs:

value INSTANCE

The value of the topic object.

value_name STRING

The string value of the specified slot to be returned.

error_msg STRING

An error message to be returned. (Not currently used.)

Instances updated: None

form_get_topic_class

Description: Gets the topic class. It returns the class and the name of the class for display purposes.

Inputs:

topic_object INSTANCE
The topic object. (Not used.)

slot_name STRING
The slot whose value is to be returned. (Not used.)

topic_class CLASS
The topic class.

Required Inputs: topic_class

Input Defaults: None

Outputs:

value CLASS
The topic class.

value_name STRING
The string value of the topic class.

error_msg STRING
An error message to be returned. (Not currently used.)

Instances updated: None

form_get_slot_class

Description: Gets the value of a specified slot on the topic object. The slot must be of type class. It returns the class and the name of the class.

Inputs:

topic_object INSTANCE
The topic object.

slot_name STRING
The slot whose value is to be returned.

Required Inputs: topic_object, slot_name

Input Defaults: None

Outputs:

value CLASS

The value of the specified slot.

value_name STRING

The string value of the class of the specified slot.

error_msg STRING

An error message to be returned. (Not currently used.)

Instances updated: None

form_get_slot_string_class

Description: Gets the value of a specified slot on the topic, which is a string containing the name of the class. It returns the class and the name of the class.

Inputs:

topic_object INSTANCE

The topic object.

slot_name STRING

The slot whose value is to be returned.

Required Inputs: topic_object, slot_name

Input Defaults: None

Outputs:

value CLASS

The class as specified by the string value of the slot.

value_name STRING

The string value of the class specified by the slot.

error_msg STRING

An error message to be returned. (Not currently used.)

Instances updated: None

form_get_class_of_instance

Description: Gets the class of a specified topic object. It returns the class and the name of the class.

Inputs:

topic_object INSTANCE

The topic object.

slot_name STRING

The slot whose value is to be returned. (Not used.)

Required Inputs: topic_object

Input Defaults: None

Outputs:

value CLASS

The class of the specified object.

value_name STRING

The string value of the class of the specified object.

error_msg STRING

An error message to be returned. (Not currently used.)

Instances updated: None

form_get_slot_boolean

Description: Gets the value of a specified slot on a given topic object. The slot must be of type Boolean and the action will return the integer equivalent: FALSE = 0, TRUE = 1.

Inputs:

topic_object INSTANCE

The topic object.

slot_name STRING

The slot whose value is to be returned.

Required Inputs: topic_object, slot_name

Input Defaults: None

Outputs:

value INTEGER

The integer value of the slot.

error_msg STRING

An error message to be returned. (Not currently used.)

Instances updated: None

form_get_subclasses_of_class

Description: Gets all the subclasses of a class. It returns an osset of all the subclasses.

Inputs:

topic_object INSTANCE

The topic object. (Not used.)

slot_name STRING

The slot whose value is to be returned. (Not used.)

class_name STRING

The name of the class whose subclasses are to be returned.

Required Inputs: class_name

Input Defaults: None

Outputs:

value oset of CLASS

The list of subclasses.

hr_strings oset of STRING

The list of the names of the subclasses returned.

error_msg STRING

An error message to be returned. (Not currently used.)

Instances updated: None

form_get_instances_of_class

Description: Gets all the instances of a class and any subclasses of the class. It returns an oset of all the instances.

Inputs:

topic_object INSTANCE

The topic object. (Not used.)

slot_name STRING

The slot whose value is to be returned. (Not used.)

class_name STRING

The name of the class whose instances are to be returned.

Required Inputs: class_name

Input Defaults: None

Outputs:

value oset of INSTANCE

The list of instances.

hr_strings oset of STRING

The list of the names of the instances returned.

error_msg STRING

An error message to be returned. (Not currently used.)

Instances updated: None

form_get_enums_of_class

Description: Gets all the values of an enum class. It returns an oset of all the values.

Inputs:

topic_object INSTANCE

The topic object. (Not used.)

slot_name STRING

The slot whose value is to be returned. (Not used.)

class_name STRING

The name of the enum class whose values are to be returned.

Required Inputs: class_name

Input Defaults: None

Outputs:

value oset of INSTANCE

The list of values.

hr_strings oset of STRING

The string value of the enum values returned.

error_msg STRING

An error message to be returned. (Not currently used.)

Instances updated: None

get_project_times

Description: Gets the values of the start of time, end of time, early fence, late fence, leveling fence, and current time for an environment.

Inputs:

env_name STRING

The environment name.

Required Inputs: env_name

Input Defaults: None

Outputs:

start_of_time DATE
The environment's start of time.

end_of_time DATE
The environment's end of time.

early_fence DATE
The environment's early fence.

late_fence DATE
The environment's late fence.

leveling_fence DATE
The environment's leveling fence.

current_time DATE
The environment's current time.

exit_msg STRING
An exit message to be read by the user.

Instances updated: None

Setting Data with Page Transactions

The following are generic transactions that set data on the server. In general, specific transactions are created for modifying data on the server, as there may be error checking or other specific requirements done. However, these generic transactions can be used to take values that the user enters on the pages and controls to set data on the server.

form_set_slot_int

Description: Sets the value of a specified slot on a given object. The slot must be of type integer.

Inputs:

object INSTANCE
The input object.

slot_name STRING
The slot whose value is to be modified.

value INTEGER
The new value to set the slot to.

Required Inputs: object, slot_name, value

Input Defaults: None

Outputs: None

Instances updated: object.slot_name

form_set_slot_float

Description: Sets the value of a specified slot on a given object. The slot must be of type float.

Inputs:

object INSTANCE
The input object.

slot_name STRING
The slot whose value is to be modified.

value FLOAT
The new value to set the slot to.

Required Inputs: object, slot_name, value

Input Defaults: None

Outputs: None

Instances updated: object.slot_name

form_set_slot_string

Description: Sets the value of a specified slot on a given object. The slot must be of type string.

Inputs:

object INSTANCE
The input object.

slot_name STRING
The slot whose value is to be modified.

value STRING
The new value to set the slot to.

Required Inputs: object, slot_name, value

Input Defaults: None

Outputs: None

Instances updated: object.slot_name

form_set_slot_date

Description: Sets the value of a specified slot on a given object. The slot must be of type date.

Inputs:

object INSTANCE

The input object.

slot_name STRING

The slot whose value is to be modified.

value DATE

The new value to set the slot to.

Required Inputs: object, slot_name, value

Input Defaults: None

Outputs: None

Instances updated: object.slot_name

form_set_slot_time

Description: Sets the value of a specified slot on a given object. The slot must be of type time.

Inputs:

object INSTANCE

The input object.

slot_name STRING

The slot whose value is to be modified.

value TIME

The new value to set the slot to.

Required Inputs: object, slot_name, value

Input Defaults: None

Outputs: None

Instances updated: object.slot_name

form_set_slot_instance

Description: Sets the value of a specified slot on a given object. The slot must be of type instance.

Inputs:

object INSTANCE

The input object.

slot_name STRING

The slot whose value is to be modified.

value INSTANCE

The new value to set the slot to.

Required Inputs: object, slot_name, value

Input Defaults: None

Outputs: None

Instances updated: object.slot_name

form_set_slot_boolean

Description: Sets the value of a specified slot on a given object. The slot must be of type Boolean enum, and the value must be of type integer, with a value of 0 or 1.

Inputs:

object INSTANCE

The input object.

slot_name STRING

The slot whose value is to be modified.

value INTEGER

The new value to set the slot to.

Required Inputs: object, slot_name, value

Input Defaults: None

Outputs: None

Instances updated: object.slot_name

Page Transactions that Perform Other Functions

form_check_topic_instance

Checks if the passed topic object is a valid object. If so, it returns the object. If not, it fails with an error.

Inputs:

topic_object INSTANCE

The topic object.

Required Inputs: topic_object

Input Defaults: None

Outputs:

value INSTANCE

The object, if it is valid.

exit_msg STRING

An exit message to be read by the user.

Instances updated: None

form_sort_slot_instance_list

Description: Sorts the rows of a grid.



For an example, refer to Understanding the Purchase Order Page Code.

Inputs:

topic_object INSTANCE

The topic object.

slot_name STRING

The slot name.

order_by STRING

The string value of the order by clause.

Required Inputs: topic_object, slot_name, order_by

Input Defaults: None

Outputs:

value oset of INSTANCES

The sorted list of instances.

exit_msg STRING

An exit message to be read by the user.

Instances updated: None

Arrays and Vectors

Although the PepperCode programming language supports them, the Resource Language does not support arrays or vectors.

CHAPTER 11

Using the Visual Browser

The visual browser, BowserNG, is a browser for the following entities in the server: PepperCode (or SPL) files, actions, action schemas, classes, enums, and C++ functions. It allows you to see such information as the source code for these entities, which SPL files are referenced by `#include` statements in an SPL file, and what entities contain references to a specified action, action schema, enum, or C++ function.



Note: Since the Visual Browser refers to PepperCode files as SPL files, this section will do this also.

Installing Tcl/Tk (Windows NT only)

Before you can run the visual browser on Windows NT, you must install Tcl/Tk. To do so, perform the following steps:

1. Where the product is installed, navigate to the following directory:
 - `RPS_SDK\bin` if you have Pepper Tools. `$RPS_SDK` is the directory where you installed Pepper Tools.
 - `RPS\product\DETACH` if you have the regular product without Pepper Tools. `RPS` is the directory where you installed the product.
2. Run the `win76.exe` program. Either run it from the command line or double-click its icon.
3. Follow the steps in the Install Shield program.

Running the Visual Browser

To run the visual browser, perform the following steps where the product is installed:

1. Start the visual browser.
 - If the visual browser was installed with Pepper Tools on UNIX, enter the following commands, where `$RPS_SDK` is the directory where you installed Pepper Tools:

```
cd $RPS_SDK/bin
```

```
browser_ng
```

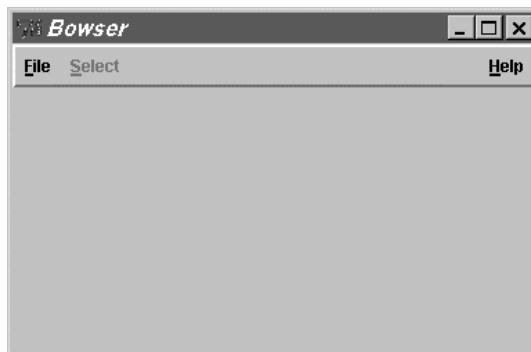
- If the visual browser was installed with Pepper Tools on Windows NT, navigate to the RPS_SDK\bin directory and then double-click browser_ng_tcl, where RPS_SDK is the directory where you installed Pepper Tools.
- If the visual browser was installed with the product on UNIX, enter the following commands, where \$RPS is the directory where you installed the product:

```
cd $RPS/product/DETACH
```

```
browser_ng
```

- If the visual browser was installed with the product on Windows NT, navigate to the RPS\product\DETACH directory and then double-click browser_ng_tcl, where RPS is the directory where you installed the product.

The visual browser main window appears, as shown below. The Select menu will not be active, since you have not yet selected source files or a database to browse.

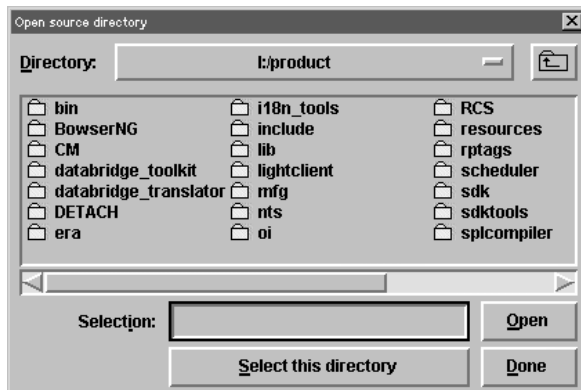


Visual Browser main window

You will either parse the source files that you want to browse, or you will open a database that you want to browse. To see how to parse source files—and create a database from those source files that you can open when you run the browser at a later time—go to step 2. To see how to open a database, go to step 3.

2. Select **File, Parse Source Files**, then navigate to and select the directories that contain the source files that you want to browse, then go to Step 4.

The Parse Source Files page appears, as shown below.



Parse Source Files Page

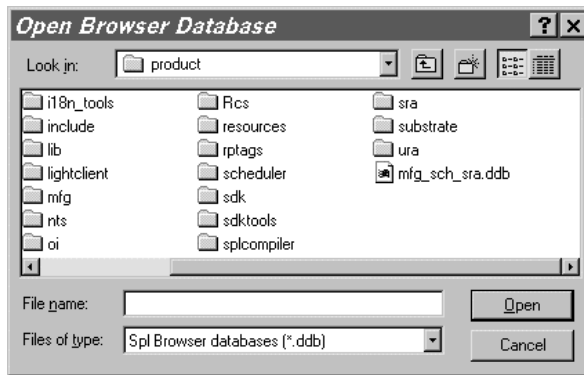
Source files end with “.spl”. Navigate to and select the directory or directories containing the source files that you wish to browse by doing the following:

- a. Navigate to the desired directory. Click **Open** or double-click on directory names to enter a directory. Use the upper directory button in the top right page corner to enter the next higher level of directories.
- b. Highlight the directory name and click **Select This Directory**. If a directory is a subdirectory of a directory that you have already selected, you do not have to select that directory, since it will automatically be included with the directory you already selected.

Repeat steps a) and b) for every directory that you want to select.

- c. After you have selected all the directories that you wish to browse, click **Done** or the close page button—the X button in the top right page corner. The Parse Source page disappears.
 - d. If you want to create a database file, select **File, Save** and save to a file ending with “.ddb”. This will create a database file that you can browse the next time you run the visual browser. Browsing this database file will give you the same information as browsing the source files that you just selected. This can be useful if you often browse the same source files.
 - e. Go to Step 4.
3. Select File, Open Database.

The Open Database page appears, as shown below. This page allows you to navigate to the database file that you want to browse.



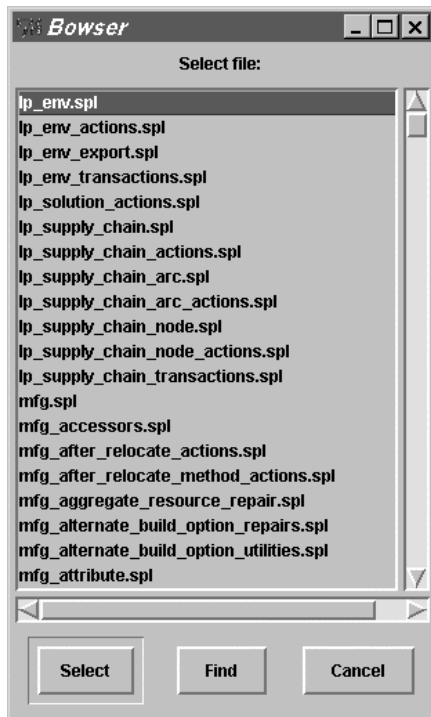
Open Database Page

Database files end with “.ddb”. Either double-click the file name, or highlight it and click **Open**. The open Database Page disappears. If you want to close this page without selecting a database, click **Cancel**.

The directory \$RPS/product/ura or RPS\product\ura—\$RPS or RPS being the directory where you installed the product—contains the database file for the server: ura.ddb.

4. From the now active Select menu on the main page, choose one of the entity types from the menu: SPL File, Action, Action Schema, Class, Enum, C++ Function, Read-only Action, or Read-only Schema.

This brings up a page that lists all of the entities of the type—for example, SPL File or Action—that are contained in the directories or database that you selected. For example, if you choose SPL File, you will get a list page that lists SPL files. This will look similar to the following:



List Page: SPL File

The list pages all have three buttons:

Select: select the highlighted entity on the list.

Find: open a Find page that allows you to type the name of an entity you wish to select.

Cancel: Close the list window without selecting an entity.

5. By clicking **Select** or **Find**, select an entity to browse: in this case, the entities are SPL. A entity list page appears that lists the entities within the selected entity. The page below shows the entities within the SPL file mfg_transactions.spl.



Entity List Page: SPL File

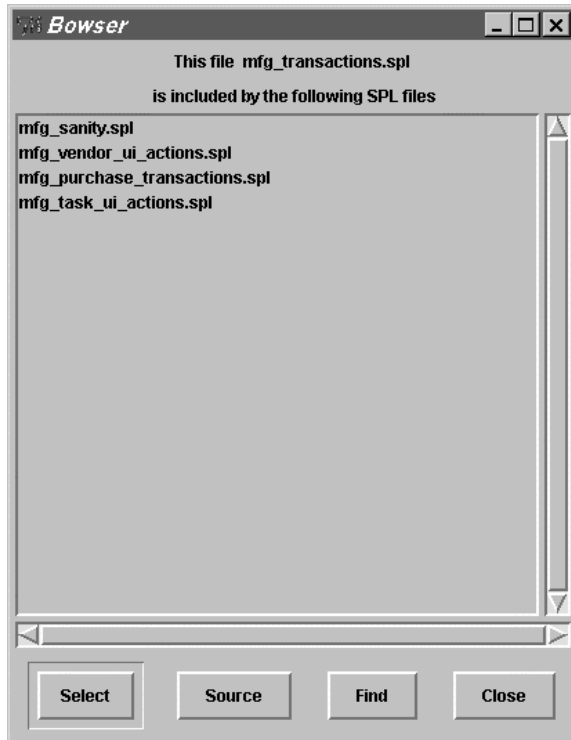
You can select an entity to browse from this list page as well by double-clicking it. For example, double-clicking action <transaction_material_only_planning> will bring up a list page showing the entities within the action transaction_material_only_planning.

- Click one of the buttons on the bottom of the page to show information.

The table below shows what happens when you press one of the buttons in the SPL File Entity list page. To see what happens when you press the buttons on all the Entity list pages, refer to “Using the Visual Browser Menus,” later in this section.

Button	<i>What happens when you press the button</i>
Includes	Lists the SPL files whose names are contained in the #include statements in this file.
Included_by	Lists the SPL files that contain #include statements that name this file.
Source	Displays the source code for this SPL file.
Find	Finds the occurrences of the text string that you enter within this page. The Find operation is case sensitive.
Close	Closes the page.

Here is an example of what you could see after pressing the Included_by button for the SPL file mfg_transactions.spl. It shows that the SPL files listed in the following page all have an #include statement that references mfg_transaction.spl.



Include List Page: mfg_transactions.spl

You can select an entity to browse from this list page as well by double-clicking it. For example, double-clicking mfg_sanity.spl will bring up a list page showing the entities within the SPL file mfg_sanity.spl.

7. When you have finished browsing, go to the main browser window and select **File, Quit**.

Using the Visual Browser Menus

The visual browser main menu offers the following items. The top menu items are in the left column.

<i>Main Menu</i>	<i>Submenu</i>	<i>Description</i>
File		
	Open Database	Open a database file for browsing. You will be supplied a database file that corresponds to the 7.5 release SPL source files,

Main Menu	Submenu	Description
		and when you browse that database file, you will get the same information that you would from the corresponding source files except for seeing the source code itself. You can also create your own database files.
	Parse Source Files	Open SPL source files for browsing.
	Save	Once you have opened source files for browsing, you can use Save to create a database file.
	Quit	Quit the visual browser.
Select		
	SPL File	Shows you a list of all the entities defined in this SPL file and lets you select one or more to display. To select more than one item, hold down the shift key as you click.
	Action	Shows you all of the input/output/local parameters defined in this action and inherited from action schema, if any. Inherited parameters are prefixed by the schema name followed by double colons ("::"). Parameter default values, if any, will be shown after the parameter names. To select more than one item, hold down the shift key as you click.
	Action Schema	Shows you all of the input/output/local parameters defined in this action schema. Parameter

Main Menu	Submenu	Description
		default values, if any, will be shown after the parameter names. To select more than one item, hold down the shift key as you click.
	Class	Shows you all of the slots defined in this class or inherited from its parents. Inherited slots are prefixed by the ancestral class's name and double colons. Slot properties, such as default value or class slot, are shown after the slot name. To select more than one item, hold down the shift key as you click.
	Enum	Shows you the values of the enum. To select more than one item, hold down the shift key as you click.
	C++ Function	Shows you the return type and function parameters of this C++ function. To select more than one item, hold down the shift key as you click.
	Sort by...	Allows you to sort the entities in the entity list pages that appear when you click the includes/references or included_by/referenced_by buttons. You can sort by name of the entity, or by the type of the entity. Default is by name.
	Read-only Action	Shows you all of the input/output/local parameters defined in this read-only action and inherited from action schema, if any. Inherited

Main Menu	Submenu	Description
		parameters are prefixed by the schema name followed by double colons ("::"). Parameter default values, if any, will be shown after the parameter names. To select more than one item, hold down the shift key as you click.
	Read-only Schema	Shows you all of the input/output/local parameters defined in this read-only action schema. Parameter default values, if any, will be shown after the parameter names. To select more than one item, hold down the shift key as you click.
Help		
	About Browser	Displays a copyright notice.
	Help	Displays a help text file containing information about the visual browser.

When you use the Select menu and one of its submenus, the browser displays a page containing a list of the entities buttons that you can use. The buttons vary slightly with each submenu: "Spl File", "Action", "Action Schema", "Class", "Enum", "C++ Function".



Note: You will be supplied with ura.ddb, which is the database file for the product. It is located at \$RPS/product/ura or RPS\product\ura, \$RPS or RPS being the directory where you installed the product. This database allows you to browse the Red Pepper product. However, some of the source code will not be shown when you use this database. This means that the Source buttons will not always show the source code when you browse this database.

Using the SPL File Entity List Page Buttons

Button	<i>What happens when you press the button</i>
Includes	Lists the SPL files whose names are contained in the #include statements in this file.
Included_by	Lists the SPL files that contain #include statements that name this file.
Source	Displays the source code for this SPL file.
Find	Finds the occurrences of the text string that you enter within this page. The Find operation is case sensitive.
Close	Closes the page.

Using the Action and Read-only Action Entity List Buttons

Button	<i>What happens when you press the button</i>
Schema	Displays the schema of this action, if any.
References	Lists the references to entities that are contained in this action.
Referenced_by	Lists the entities that contain references to this action.
Source	Displays the source code for this action.
Find	Finds the occurrences of the text string that you enter within this page. The Find operation is case sensitive.
Close	Closes the page.

Using the Action Schema and Read-only Action Schema Entity List Buttons

Button	<i>What happens when you press the button</i>
Actions_of_schema	Displays the actions that inherited from this action schema.
References	Lists the references to entities that are contained in this action schema.
Referenced_by	Lists the entities that contain references to this action schema.
Source	Displays the source code for this action schema.
Find	Finds the occurrences of the text string that you enter within this page. The Find operation is case sensitive.
Close	Closes the page.

Using the Class Entity List Buttons

Button	<i>What happens when you press the button</i>
Parents	Lists the direct parents of this class.
Children	Lists the direct or indirect descendants of this class.
References	Lists the references to entities that are contained in this class.
Referenced_by	Lists the entities that contain references to this class.
Source	Displays the source code for this class.
Find	Finds the occurrences of the text string that you enter within this page. The Find operation is case sensitive.
Close	Closes the page.

Using the Enum Entity List Buttons

<i>Button</i>	<i>What happens when you press the button</i>
Referenced_by	Lists the entities that contain references to this enum.
Source	Displays the source code for this enum.
Find	Finds the occurrences of the text string that you enter within this page. The Find operation is case sensitive.
Close	Closes the page.

Using the C++ Function Entity List Buttons

<i>Button</i>	<i>What happens when you press the button</i>
References	Lists the references to entities that are contained in this C++ function.
Referenced_by	Lists the entities that contain references to this C++ function.
Source	Displays the source code for this C++ function.
Find	Finds the occurrences of the text string that you enter within this page. The Find operation is case sensitive.
Close	Closes the page.

CHAPTER 12

Writing Queries

All client queries are active. The execution of a "select" statement creates an instance of type "Query" and subscribes to it on behalf of the client; the server then sends the client a notification for that object just as it would for any other subscribed object.

Thereafter, every time the server processes the net changes following an action-interpreter command, it reevaluates the query. If the reevaluation changes the instance, then the subscription mechanism sends a notification to the client.

To end a query, either unsubscribe or delete the query object itself.

The Query class is created automatically if the PepperCode program does not create it explicitly. If the PepperCode program does create it explicitly, its declaration must have the following slots, or the server will emit an error message and all queries will fail:

```
class Query {  
  
    int first_difference          // origin zero  
  
    oset[instance<Base_Class>] instances  
  
};
```

The client can use the action interpreter "get_list" command described later to obtain a subset of the list of instances on the slot "instance"; this is the "cursor on osets" feature.

The "first_difference" slot is initially zero. Whenever the server reevaluates a query and the "instances" list changes, it sets the "first_difference" slot to indicate the index, origin zero, of the earliest change in the list.

Understanding Query Syntax

A query is an operation where the server creates and updates a list of objects which pass a test. The list is stored on a slot of type oset[instance<Base_Class>] on an ordinary PepperCode instance of a special class called "Query", so that one may obtain the updated list at any time by reading that slot.

A query is first performed when the server is presented with a "select" statement, which has the form:

```
select <class list>  
  
    where <selection criteria>
```

```
order by <ordering criteria>;
```

For example, you might say:

```
select Resource_Constraint where weight = 1.5 order by quantity;
```

The first logical step in evaluating a query is to select all instances of all classes in the class list, eliminating duplicates.

The second logical step is to evaluate the "selection criteria" expression for each instance selected in the first step, and to eliminate any instance for which the expression is false, or for which a slot is missing.

The third logical step is to sort the remaining instances according to the list of keys specified by the "order by" clause.

The result of the query is this sorted set of instances.

The "where" clause is optional, and defaults to "true". The "order by" clause is optional, but without it the list of objects is unsorted.

The syntax of the individual clauses is explained in the following sections.

Writing a Query class list

The comma-separated class list contains one or more PepperCode class names. Each name may be either an identifier or a quoted string—quoted strings allow strange names like "SO-00-005" or "Nine \" Nails"—or an oid such as "oid(12345)".

Alternatively, the entire list may be "*", indicating all classes.

Writing a Query selection criteria

This is a Boolean expression which compares slots against literal values, or invokes a transaction which takes an instance as an argument and yields a Boolean result.

Using Query Expressions

An expression may be any of:

```
<expression> and <expression>
```

```
<expression> or <expression>
```

```
not <expression>
```

```
( <expression> )
```

```
<predicate>
```


Using Query Predicates

A predicate may be:

```
<slot_or_literal> <relational_operator> <slot_or_literal>

<transaction_name> ( <argument_list> )
```

An example of the first kind of predicate is:

```
constraint.selected_object_class.weight = 1.5 and name != "alpha"
```

In a predicate, a slot name can be either a single identifier or a list of identifiers separated by dots. The first identifier in the dot-separated list is the name of a slot on the instance which is being tested for inclusion or exclusion in the oset of results.

Thus, for example:

```
constraint.selected_object.class.weight
```

means "take the object which is a candidate for inclusion in the query result set and look up its 'constraint' slot; then take the object on that slot and look up its 'selected_object' slot; then take the object on that slot and look up its 'class' slot; then take the object on that slot and look up the value of its 'weight' slot".

Using Relational Operators

A relational operator may be:

=	
!=	
>	
<	
>=	
~	(does string on left satisfy regular expression on right?)
~=	(does string on left not satisfy regular expression on right?)
?.	(does the slot specified on the right side of the operator exist on the object specified on the left side?)

You can use literals and slot names on either side of a relational operator, as in the following examples:

```
select * where 1 = 0; // Example 1
```

```
select * where one_slot = another_slot; //Example 2
```

For floating-point operands, the comparison operators ignore roundoff errors using the feature described in conjunction with the PepperCode intrinsic SET_EPSILON.

A programmer would never write example 1 explicitly, of course, but it might be generated by a client resource form when testing to see whether an entry field holds a particular reserved value.

Binary operators are left-associative and take precedence in the order just presented; use parentheses when in doubt.

Notice that the regular expression operators are not commutative: the right operand is always treated as the regular expression, whether it is a literal or a slot name.

A literal may be:

```
<integer_literal>
```

```
<float_literal>
```

```
<string_literal>
```

```
<oid_literal>
```

Integer literals are decimal.

Float literals observe the usual C++ rules, but must have either a decimal point or an exponent. For example, "0" is an integer but "0.0" is a float. There is no automatic conversion of data type between an integer literal and a float slot, or vice versa.

String literals begin and end with double-quotes, and any double-quote or backslash character inside the string must be preceded by a backslash.

An Oid literal looks like this, where "name" is optional and ignored:

```
oid(123 "name")
```

If data types are mismatched within the "where" clause, or if you apply an operator to an impossible data type, the query will evaluate to "false" and the server will emit an error message.

If a slot of type instance (class) is nil, it is treated as if it were Null_Instance (Null_Class).

Boolean operators perform short-circuit evaluation. This means that if the left argument of "and" evaluates to "false", the right argument is not evaluated; if the left argument of "or" evaluates to "true", the right argument is not evaluated.

The "?. " operator tells whether a slot exists on an object; it is particularly useful along with short-circuit evaluation. For example, if a slot named "schedule" sometimes has the value "Null_Instance" but at other times has a real instance, the following predicate will quietly ignore objects for which "schedule" is nil, and will select objects for which the schedule is a class having a display name beginning with "weekly":

```
schedule ?. display_name and display_name ~ "^weekly_"
```

An example of the second kind of predicate is:

```
transaction_test_weight(:object %input
:slot_name "weight" :acceptable %output)
```

In this kind of predicate, the argument list includes one or more argument tuples. At least one of the tuples must be:

```
:<argument_name> %output
```

This argument must have integer type. PepperCode does not now have an intrinsic Boolean data type, and while the PRA defines the enumeration "Boolean_Flag", an arbitrary PepperCode program might not. If the value is nonzero, the transaction predicate is treated as "true"; otherwise, it's treated as "false".

In addition, one or more arguments may be:

```
<argument_name> %input
:<argument_name> <literal>
```

For each argument which uses "%input", the data type must be "instance".

If the transaction attempts to make changes to its context, the effect is undefined; in the current implementation, the changes are quietly discarded.

Using Instance Operator Syntax

Any identifier in a "select" statement implicitly gives the name of a slot, not the name of an instance; slots are assumed to belong to the candidate instance for which the "where" clause is currently being evaluated.

An operator "%instance" lets you use an identifier to specify the name of an instance, which need not be a member of the candidate set, instead. In the following example, "some_slot" refers to a slot on each of the candidate instances in the set of instances of "Some_Class"; "its_slot" refers to a slot on the particular instance whose name is "my_instance":

```
select Some_Class where some_slot = %instance(my_instance).its_slot;
```

The grammar does not actually require parentheses after %instance, so this has the same meaning as the preceding example:

```
select Some_Class where some_slot = %instance my_instance.its_slot;
```

Using Arithmetic Operators

You can use the following arithmetic operators on integer and floating point values, with the typical precedence rules:

.	addition, unary addition
-	subtraction, unary subtraction

*	multiplication
/	division

Using Parentheses To Enforce Operator Precedence

You can use parentheses to alter or enforce precedence:

```
select Some_Class where (slot1 * (slot2 + 5)) = 77;
```

Using Conversion Operators

You can use these conversion operators:

%integer	convert floating point value to integer
%float	convert integer value to floating point
%date	convert date string to date (integer)

The %date operator uses the same syntax for the string representation of the date as the PepperCode function STRING_TO_DATE uses.

Example:

```
select ClassWithDateSlot where date_slot = %date "12/31/96"
```

Using Oset Operators

There is an operator "%length" which takes an oset of objects as its operand and returns an integer giving the number of members in the oset:

```
select Some_Class where (%length slot_of_type_oset) > 0;
```

And there is a binary operator "%in" which asks whether a particular value belongs to an oset. If the left operand of "%in" has type X (where X can be integer, float, string, or instance), the right operand has type oset[X]. The "%in" operator returns true if the left operand is a member of the right operand set:

```
select Some_Class where slot_of_type_instance %in slot_of_type_oset;
```

Using Expression Comparisons

Regular expression comparisons use these rules:

.	Matches any single character.
---	-------------------------------

<code>^</code>	Matches the beginning of the string.
<code>\$</code>	Matches the end of the string.
<code>\x</code>	Matches the character x.
<code>[abcd]</code>	Matches any single character from the set abcd.
<code>[^abcd]</code>	Matches any single character not in the set abcd.
<code>[a-d]</code>	Matches any single character between a and d inclusively.
<code>[^a-d]</code>	Matches any single character not between a and d inclusively.
<code>(regexp)</code>	Matches anything that matches regexp.
<code>*</code>	Matches a sequence of 0 or more of the preceding atom.
<code>.</code>	Matches a sequence of 1 or more of the preceding atom.
<code>?</code>	Matches 0 or 1 occurrence of the preceding atom.
<code>e1 e2</code>	Matches either expression e1 or expression e2.

Note that these comparisons behave like Unix "grep", not like the Unix or MS-DOS shells. For example:

- To match "abc" but not "123abc456", you would use the pattern "`^abc$`", not the pattern "`abc`".
- The pattern "`*.c`" is illegal; to match any string that ends in a dot and the character "c", you would use "`\.c$`".
- To match any string containing exactly two characters, you would use the pattern "`^..$`", not the pattern "`??`".

Anchoring patterns with "`^`" at the beginning and "`$`" at the end not only ensures that you don't inadvertently match strings that you don't want, but also improves execution speed.

Nesting Select Statements

You may nest one "select" statement inside the "where" clause of another, as the following—admittedly nonsensical—example shows. A nested "select" statement must always be enclosed in parentheses. Unlike an outer "select" statement, it does not create a query object and send notifications to the client. Instead, it generates a temporary, unnamed osset of instances containing the results of the query. You can then use the "`%in`" or "`%length`" operator on that osset:

```
select Some_Class where

    %length (select Another_Class where some_slot = 5) > 0;
```

The reason this is nonsensical is that if the inner select generates an empty set, the outer select will generate an empty set; if the inner select generates a nonempty set, the outer select will generate a set of all instances of "Some_Class", regardless of the slot values on those instances! A better example might be:

```
select Some_Class where

    some_slot %in (select Another_Class where another_slot = 5);
```

This second example first selects some set of instances of Another_Class for which another_slot is 5, and creates an inner result set; then it examines the instances of Some_Class, asking for each one whether the slot "some_slot" points to one of the instances of Another_Class in the inner result set. Provided that "some_slot" always points to an instance of "Another_Class", then the following query is equivalent:

```
select Some_Class where some_slot.another_slot = 5;
```

In addition to the normal "select" statement, which starts with a list of class names and tests all instances of those classes against the "where" predicate, there is a variant called "select_aset" which starts with an aset of instances and tests only those instances (the PepperCode QUERY_OSET intrinsic, described elsewhere, actually uses the variant form). Here is an example which uses the normal form as the outer statement, and the variant as the nested statement:

```
select Some_Class where 0 != %length

    (select_aset slot_of_type_aset where some_slot = 5);
```

As an example of the use of nested "select" statements, suppose you have a class "Product" with a slot called "price" which contains a floating point number, and a slot called "customers" which contains an aset of instances of a second class called "Customer". Suppose also that "customer" has a slot "geography" which contains a string. To ask for all of the products costing more than \$50.00 for which there are customers in France, you might say:

```
select Product

    where (price > 50.0) and

        (%length (select customers where geography = "France")) > 0;
```

Writing a Query Ordering Clause

There are two forms of the "order" clause: order randomly and order by.

Using order randomly

If you say "order randomly", the instances are shuffled using a random number generator. Because the client is not permitted to issue any commands—other than PepperCode transactions, which appear in the log file—which alter the user-visible state of the server, the client may not use "order randomly" directly. The client may invoke an PepperCode transaction which calls the QUERY_OSET intrinsic, and that transaction can use "order randomly". This form of query can only be used in PepperCode code, not in client pages.

Using order by

If you say "order by", you use a list of keys to sort the instances; the OID of the instance is implicitly the last key in the list, so the sort order is completely deterministic.

When you use the "order by" form, you provide a comma-separated list of the primary, secondary, ..., xxx-ary keys for sorting the instances remaining after the second step. Each key is a path consisting of one or more slot name identifiers separated by ".". Sorting uses ascending order by default, but you may put "-" after each key to indicate descending order.

For example:

```
order by selected_object.display_name,
        selected_object.initial_amount-;
```

would obtain the primary key by looking up the slot "selected_object" on each instance in the result set, and then looking up the slot "display_name" on the "selected_object" instance.

The data type of a key must be integer, date, time, float, or string.

If a slot cannot be found, the query reports an error.

In the previous example, if the "selected_object" were an instance which does not have an "initial_amount" slot, an error would occur.

There are three exceptions to this rule.

- First, in the current implementation, access to keys is "lazy"; the query doesn't look at the secondary keys when comparing a particular pair of result objects unless the primary keys compare equal. You decide whether to depend on that implementation detail.
- Second, if the query encounters "Null_Instance" while following a path to a slot, it pretends that it found that slot, but that the slot had a default value of zero or the empty string. If it encounters "Null_Instance" while following the paths to the slots on both of the result objects, it pretends that the objects compared equal for that key.
- Third, if an object does not have a slot called "name", then the query will pretend that there is a slot called "name" containing the string name of the object; if an object does not have a slot called "class", then the query will pretend that there is a slot called "class" containing the name of the class to which that object belongs.

The entire "order by" clause is optional; or you can include the "order by" keywords and omit the list of keys. In either case, the instances are not sorted.

Arrays and Vectors

Although the PepperCode programming language supports them, the query language does not support arrays or vectors.

Index

A

- action entity list page buttons for visual browser 11-11
- action schema entity list page buttons for visual browser 11-12
- Add pages 2-13
- Form Painter 9-13
- grids 9-13
- spreadsheets 9-13
- Form Painter 9-9
- pages 9-9
- columns 9-13
- Form Painter 9-16
- controls 9-16
- Application Interface 1-1
- arithmetic operators for queries 12-5

B

- bookmarks 5-14
 - creating 5-16
 - deleting 5-18
 - editing 5-15
 - switching between 5-17

C

- C++ function entity list page buttons for visual browser 11-13
- CD-ROM
 - ordering xiv
- check boxes 1-13
- Class Editor 8-1
 - adding a subclass 8-7
 - viewing and editing a class 8-3
- class entity list page buttons for visual browser 11-12
- classes
 - viewing and editing 8-3
- command buttons
 - Data Browser 2-3
 - Supply Chain Viewer 3-7
- Combination Charts 6-8
- command buttons 1-13
- popup menus 1-7
- components 1-8
 - creating and editing with Form Painter 9-8

Control panel

- loading settings 4-6
- saving settings 4-5

controls

- adding to a page with Form Painter 9-9
- deleting from a page with Form Painter 9-10

conversion operators for queries 12-6

- Form Painter 9-8

- pages 9-2

D

data

- copying and moving 1-15
- moving and sorting data instance rows 2-3
- navigating names 2-2
- viewing instances 2-2

Data Browser 2-1

- accessing 2-1
- Add pages 2-13
- closing 2-12
- command buttons 2-3
- Find pages 2-14
- moving and sorting data instance rows 2-3
- navigating data names 2-2
- viewing data instances 2-2

data files

- working with 4-3

dataset

- loading new 4-4

- Form Painter 9-10

- pages 9-10

- drop-down lists 1-10

E

- Form Painter 9-13

- grids 9-13

- spreadsheets 9-13

- Form Painter 9-11

- columns 9-13

- controls 9-11

- pages 9-5

- Form Painter 9-15

- pages 9-5

- timeline (CTC) controls 9-15

- Enum Entity list page buttons for visual browser

- 11-13

- expression comparisons for queries 12-6

F

- File Browser 4-1
 - loading control panel settings 4-6
 - loading files 4-1
 - loading new dataset 4-4
 - saving control panel settings 4-5
 - saving files 4-3
 - snapshots 4-7
- file entity list page buttons for visual browser 11-11
- files
 - loading 4-1
 - saving 4-3
 - saving and loading 4-1
- Find pages 2-14
- Forms Painter 9-1
 - creating a page 9-2
 - editing a page 9-5
 - installing and starting 9-1
 - opening a page 9-4
 - saving a page 9-4

G

- Gantt Chart
 - closing 6-22
 - resource 6-18
- Gantt Charts
 - accessing 6-1
 - exploring 6-2
 - reading 6-4
 - task bar 6-5
 - tasks 6-11
 - timeline 6-3
 - toolbar 6-2
- Graphic Charts 6-1
- grids 1-11
 - changing column width 1-11
 - moving and sorting rows 1-12

H

- Histogram Chart
 - popup menu 6-20
- Histogram Charts
 - navigating 6-9
 - reading 6-6
 - viewing inventory 6-7
 - viewing resources 6-6
- Histograms
 - accessing 6-1
 - closing 6-22
 - navigating 6-9
- Home Base page 5-12

I

- instance operator syntax for queries 12-5
- Inventory Histogram Charts 6-7
- Inventory Report 6-8

K

- keyboard methods 1-2

L

- list boxes 1-10

M

- Main Menu 5-1
 - after connecting to server 5-3
 - after loading a dataset 5-4
 - before connecting to server 5-1
- main toolbar 1-14
- Form Painter 9-11
 - controls 9-11
 - pages 9-7
- Form Painter 9-7
- menus
 - canceling 1-7
 - choosing menu commands 1-8
 - features 1-4
 - popup 1-6
 - selecting menu item 1-6
- message boxes 1-16
- mouse methods 1-2

O

- pages 9-4
- order by for queries 12-9
- order randomly for queries 12-9
- oset operators for queries 12-6

P

- pages
 - entering data 1-10
 - working with page elements 1-9
- parenthesis for queries 12-6
- PeopleBooks
 - CD-ROM, ordering xiv
 - printed, ordering xiv
- PepperCode
 - browsing 11-1
- popup menus 1-6

- opening 1-7
- predicates for queries 12-3
- property sheet
 - setting instance and set menus with Form Painter 9-6
- property sheets 1-9
- push buttons 1-13

Q

- queries
 - arithmetic operators 12-5
 - conversion operators 12-6
 - evaluating 12-2
 - expression comparisons 12-6
 - expression syntax 12-2
 - instance operator syntax 12-5
 - nesting select statements 12-7
 - order by 12-9
 - order randomly 12-9
 - oset operators 12-6
 - parenthesis 12-6
 - predicate syntax 12-3
 - relational operators 12-3
 - syntax 12-1
 - writing 12-1
 - writing a query list class 12-2
 - writing selection criteria 12-2
- query
 - Supply Chain Diagram 3-6

R

- radio buttons 1-13
- relational operators for queries 12-3
- Resource Gantt Chart 6-18
 - cascading and collapsing tasks 6-19
 - dragging tasks 6-20
 - popup menu 6-19
- Resource Report 6-8
- Resources Histogram Charts 6-6

S

- pages 9-4
- scroll bars 1-12
- select statement nesting for queries 12-7
- server
 - configuring and establishing 5-1
- Server User information 5-14
- snapshots 4-7
 - backup 4-7
 - loading 4-8
 - saving archive 4-7
 - what-if 4-7

- spreadsheets 7-1
 - accessing 7-1
 - adding a row type 7-6
 - adding summary information 7-14
 - applying advanced filtering 7-13
 - applying basic filtering 7-12
 - changing sort order 7-10
 - choosing rows to display 7-5
 - deleting a row type 7-9
 - enabling or disabling sorting 7-11
 - exploring 7-1
 - modifying date columns 7-5
 - page features 7-4
 - pages 7-2
 - row types 7-6
 - setting filter criteria 7-12
 - setting number of rows 7-10
 - sorting 7-11
 - viewing page grids 7-3
- status bar 1-16
- subclasses
 - adding 8-7
- supply 6-16
- Supply Chain Diagram 3-2
 - command buttons 3-7
 - customizing 3-6
 - editing 3-4
 - toolbar 3-3
 - vendor information 3-2
- Supply Chain Viewer 3-1
 - accessing 3-1

T

- Task Detail 6-9
- tasks 6-11
 - cascading and collapsing on Resource Gantt Chart 6-19
 - changing status 6-17
 - dragging on Resource Gantt Chart 6-20
 - expanding and compressing 6-13
 - freezing and unfreezing 6-14
 - modifying with mouse 6-11
 - pop-up menu 6-12
 - viewing demand 6-15
 - viewing detail 6-17
 - viewing properties 6-17
 - viewing supply 6-16
 - viewing task intervals 6-14

V

- vendor information 3-2
- demand 6-15
- visual browser 11-1
 - action entity list buttons 11-11

action schema entity list page buttons 11-12
C++ function entity list page buttons 11-13
class entity list page buttons 11-12
Enum Entity list page buttons 11-13

file entity list page buttons 11-11
installing Tcl/Tk for Windows NT 11-1
menus 11-7
running 11-1