Peregrine

ServiceCenter System Tailoring, Volume 1

Release 5.1

PART NO: SCT-5.1-ENG-01013-00233

Peregrine SYSTEMS®

Peregrine Systems, Inc.
Worldwide Corporate Headquarters
3611 Valley Centre Drive San Diego, CA 92130
Tel 800.638.5231 or 858.481.5000
Fax 858.481.1751
www.peregrine.com

Contents

Getting Started 11

Tailoring ServiceCenter 11

Using the System Tailoring Guides 12

Knowledge Requirements 13

Examples 13

Contacting Customer Support 13

  Peregrine's CenterPoint Web Site 14

  Corporate Headquarters 14

  North America and South America 14

  Europe, Asia/Pacific, Africa 14

Contacting Education Services 15

Chapter 1 **Forms Designer** 17

The Drawing Canvas 18

  Structure of the canvas 19

  Sizing your forms on the canvas 20

  Positioning multiple objects 20

The Tools Palette 21

The Properties Window 22

The Java Client Forms Designer 23

  Tools Palette 23

  Java Client Properties Window 24

  Design Tools Unique to the Java Client 25

  Working with Design Objects 26

  Designing Forms 26

Chapter 2 **The Forms Design Process** 29
Accessing Forms Designer 29
Preview of the Forms Design Process 30

Chapter 3 **The Properties Window** 33
Using the Properties Window 34
Sizing the Properties Window. 34
Selecting a property 35
Entering a value manually 35
Entering a Value from the Edit Area list 35
Understanding Properties 36
Common properties 37
Font Properties 38
Box Properties 39

Chapter 4 **The Tools Palette** 41
Pointer 42
Label 44
More About Labels 46
Text. 47
Button. 50
Bevel 54
Frame 56
MultiText Box 60
Fill Box 63
Graphic 67
Check Box 70
Radio Button 73
Combo Box 78
Marquee. 83
Subformat 86
Chart 91
More about the Input property 93
More about column colors 94
Table 94
More about tables. 101

Notebooks 102

More about notebooks. 104

Wrapping Label. 104

ComFill 108

Gfile 114

Timer 118

Attachment container 122

More about Attachment containers 125

Chapter 5 **Dynamic View Dependencies** 127

Specifying property conditions 129

Chapter 6 **Designer Tips** 133

Form Style Guidelines 133

Understanding ServiceCenter screen layouts 133

ServiceCenter default fonts 134

Using the Courier font. 134

Style guidelines 135

Rules of good screen design. 136

ServiceCenter and Bitmaps 137

Adding a bitmap to an object 138

Graphics and Animation 138

Optimum performance and image appearance 139

Text and graphics 139

Chapter 7 **The Form Wizard** 141

Accessing the Form Wizard 141

Creating a Form Using the Form Wizard 143

Chapter 8 **What is Format Control?** 145

Format Control Processes 148

Main Information 148

Views 148

Queries 148

Calculations 148

Validations 148

Subroutines 149

Additional Options 149
Privileges 149
Selecting a Process View 149
Creating Expressions 150
Definitions. 150
System Functions 151
Boolean (Logical) Fields 152
File Variable 153

Chapter 9 **Getting Started With Format Control 155**
Accessing Format Control 155
Access from Forms Designer 155
Direct Access. 156
Adding a Record 156

Chapter 10 **Array Maintenance 157**
Arrayed Structures 157
as.options 158
as.copy 163
as.delete 164
as.insert 165
as.move 166
Example. 167
as.get.name 169
as.sort. 169
Sorting Simple Arrays 173
Example. 173

Chapter 11 **Special Processing Considerations 179**
Incident Management 179
Change Management 180
Master Format Control Record 181
Detail Format Control Record 181
Approval Format Control Record 181
Format Control and Eventout Records 182
Page Messages 184

Incident Management Alert Records 185

Chapter 12 **Sequential Numbering for Format Control 187**

Data Types 188

Parameters 188

Simple Sequential Numbers. 189

Create the Form 190

Create a Database File 190

Create a Format Control Record 192

Add a Record to the Database. 194

Number File 194

Fields 195

Prefixes and Suffixes 196

Create a Format Control Record 198

Add a Data Record 199

Chapter 13 **Format Control Processes 203**

Form Name 205

Format Control Record 205

Main Information 206

Fields 207

Modifying a Format Control Record 211

Validations. 213

Fields 214

Examples 215

Calculations 221

Fields 222

Examples 223

Additional File Queries. 229

Fields 230

Privileges 233

Standard Tab Fields 234

Advanced Tab Fields 237

Subroutines 240

Fields 241

Parameters 242

Examples . . . 243
Fingerprinting . . . 243
Additional Options . . . 250
Fields . . . 253
Example . . . 256
Views . . . 258
Fields . . . 260

Chapter 14 **Posting** . . . **261**
Link Records . . . 262
Fields . . . 263
Link Line Definition File . . . 265
Posting Variables . . . 268
Setting up Posting via Format Control Subroutines or Additional Options . . 271
Fields . . . 271
Examples . . . 272
Automatic Updates via Format Control Subroutines . . . 272
Manual Posting via Format Control Additional Options . . . 283
Confirming the Posting Routine . . . 288

Chapter 15 **Error Messages** . . . **293**

Chapter 16 **Common Applications Called from Format Control** . . . **295**
Applications . . . 295
fill.fc . . . 295
post.fc. . . . 296
getnumb.fc . . . 297
message.fc . . . 298
validate.fields. . . . 299
database . . . 300
query.stored . . . 301
axces.page . . . 301
axces.write . . . 302
axces.fax. . . . 303
fingerprint . . . 304
sort.array . . . 305

Determining Parameters 305

Chapter 17 **Publish and Subscribe 307**
Automatic Updates 307
No User Interaction Necessary 307
Reduced System Load 308
Workflow 308

Chapter 18 **Static Messages 311**
Simple Messages 311
Publishing the Message 313
Simple Marquees 314

Chapter 19 **Management View of a System 317**
Dynamic Display Options 318
Enhanced forms 318
Option forms 320
Startup screen 322
ServiceInfo. 324
Publishing 326
Stored queries 326
Agent records 328
Scheduling background processes 331
Schedule file 335
Subscribing 337
Designing the form 338
Creating menus. 341
Displaying the data 343
Publishing in the RAD Environment 345

Chapter 20 **ServiceInfo 347**
Designing the Form 347
Subscribing to the Data 349

Index 351

Getting Started

The ServiceCenter System Tailoring Guides have supplemental information for system administrators who install and configure ServiceCenter. Tailoring is any change to standard functionality without changing actual code. For example, you can:

- Change the look and operation of forms.
- Change default values for objects on forms that ServiceCenter uses for field validation.
- Create macros, scripts, and stored queries.
- Changes to record definitions.

Use these guides to make further changes to support site-specific requirements, including special field validation, new or modified forms design, expanded or varied workflow, and automatic notifications.

Tailoring ServiceCenter

Most tailoring can be done using high-level ServiceCenter tools, without directly changing the RAD code that is the actual ServiceCenter development medium. There are several tailoring tools, including the Database Manager, Format Control Editor, Link Editor, and Revision Control that enable you to manipulate the common RAD code sets and algorithms in the document engine. Because these tools enable extensive changes to ServiceCenter, Peregrine recommends that you analyze your requirements carefully before you begin tailoring implementation. Balance the gains of tailoring against simplifying future upgrades to new releases.

Using the System Tailoring Guides

System tailoring information appears in three separate guides. The following table shows the focus of each guide and where you should look for more information.

System Tailoring Guide, Volume 1	System Tailoring Guide, Volume 2	System Tailoring Guide, Volume 3
■ Forms Designer	■ Document engine overview	■ Incident management
■ Format Control	■ Validity table processing	■ Stored queries
■ Array maintenance	■ The notification engine	■ Sequential numbers
■ Special processing considerations	■ Global lists and Global initer	■ Scripting
■ Sequential numbering for Format Control	■ Using and creating online help	■ Plug-in support
■ Format Control processes	■ The Cascade Update utility	■ Creating wizards
■ Format Control posting	■ The display application	■ Creating and editing macros
■ Format Control error messages	■ Advanced operations	■ Development audit utility
■ Format Control common applications	■ Adding and modifying fields	■ Revision control
■ Publishing ServiceCenter information	■ Calendar management	■ DDE support
■ Static messages		■ Data Policy
■ System management		■ Clocks
■ ServiceInfo forms		■ System language: data types, variables, operators, expressions
		■ ServiceCenter default variables
		■ Link management
		■ Virtual joins

You can use the Acrobat Search feature to locate more specific topics on the ServiceCenter 5.1 documentation CD-ROM.

Knowledge Requirements

The instructions in this guide assume a working knowledge of Peregrine Systems ServiceCenter and the installation platform. You can find more information in the following guides.

- For information about a particular platform, see the appropriate platform documentation.
- For information about customizing your environment using parameters, see the *ServiceCenter Technical Reference* guide.
- Before you run the ServiceCenter server, see the *ServiceCenter User's Guide*.
- For administration and configuration information, see the *ServiceCenter System Administrator's Guide* or the *ServiceCenter Application Administration Guide.*
- For database configuration information, see the *ServiceCenter Database Management and Administration Guide.*
- For copies of the guides, download PDF versions from the CenterPoint web site using the Adobe Acrobat Reader, which is also available on the CenterPoint Web Site. For more information, see *Peregrine's CenterPoint Web Site* on page 14. You can also order printed copies of the documentation through your Peregrine Systems sales representative.

Examples

The sample windows and the examples included in this guide are for illustration only, and may differ from those at your site.

Contacting Customer Support

Contact Peregrine Systems' Customer Support through the Centerpoint Web Site.

Peregrine's CenterPoint Web Site

You can also find information about version compatibility, hardware and software requirements, and other configuration issues at Peregrine's Centerpoint web site: *http://support.peregrine.com*

1 Log in with your login ID and password.

2 Select **Go** for **CenterPoint**.

3 Select **ServiceCenter** from **My Products** at the top of the page for configuration and compatibility information.

Note: For information about local support offices, select **Whom Do I Call?** from **Contents** on the left side of the page to display the **Peregrine Worldwide Contact Information**.

Corporate Headquarters

Address:	Peregrine Systems, Inc. Attn: Customer Support 3611 Valley Centre Drive San Diego, CA 92130
Telephone:	+(1) (858) 794-7428
Fax:	+(1) (858) 480-3928

North America and South America

Telephone:	+(1) (858) 794-7428 (Mexico, Central and South America)
Fax:	+(1) (858) 480-3928
E-mail:	support@peregrine.com

Europe, Asia/Pacific, Africa

For information about local offices, see *Peregrine's CenterPoint Web Site*. You can also contact *Corporate Headquarters*.

Contacting Education Services

Training services are available for the full spectrum of Peregrine Products including ServiceCenter.

Current details of our training services are available through the following main contacts or at:

http://www.peregrine.com/education

Address:	Peregrine Systems, Inc. Attn: Education Services 3611 Valley Centre Drive San Diego, CA 92130
Telephone:	+1 (858) 794-5009
Fax:	+1 (858) 480-3928

1 CHAPTER Forms Designer

Forms Designer is the utility you use to design, create, and update ServiceCenter forms. This utility gives access to a Drawing Canvas upon which you construct your forms, a Tools Palette from which you access design objects, and a Properties Window that you use to set attributes for each object.

In addition, Forms Designer provides a *Form Wizard* that you can use to automatically create forms based on a particular Database Dictionary.

The Drawing Canvas

When you access Forms Designer to create a new form, you are provided with a Drawing Canvas on which you build the form.

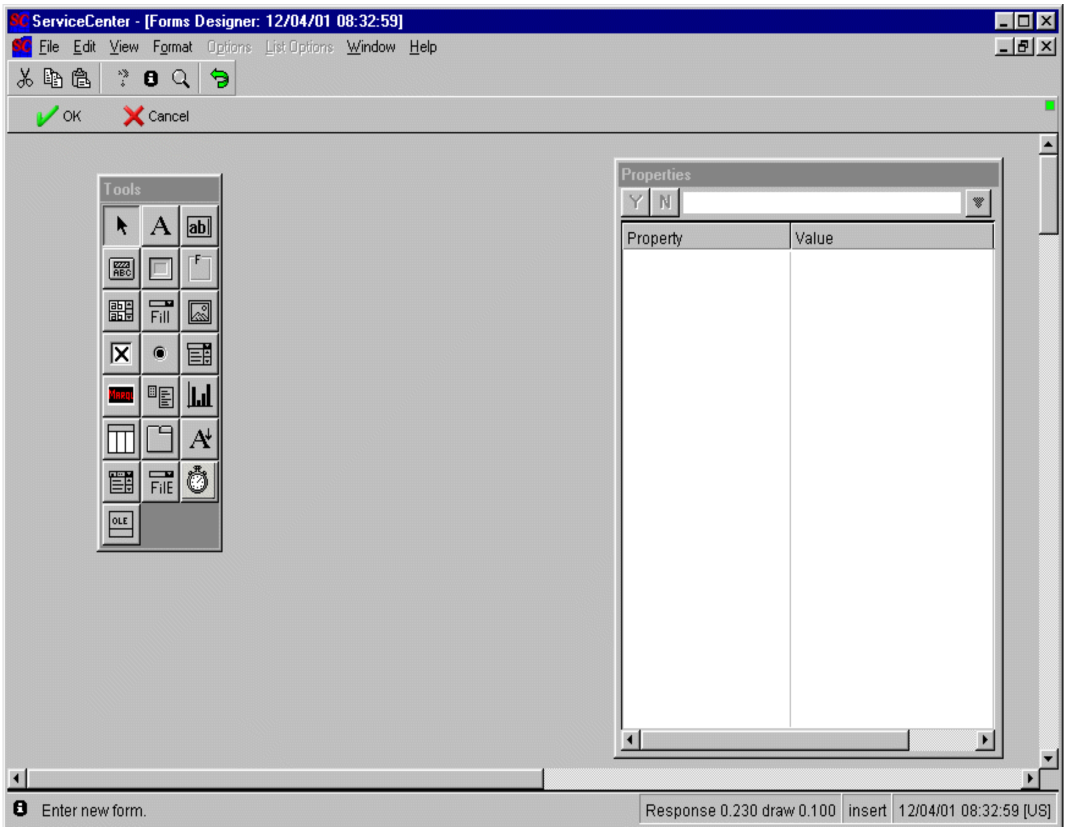

ServiceCenter - [Forms Designer: 12/04/01 08:32:59]
File Edit View Format Options List Options Window Help
OK
Cancel
Tools
Properties
Property
Value
Enter new form.
Response 0.230 draw 0.100
insert
12/04/01 08:32:59 [US]

Figure 1-1: The Drawing Canvas

To work effectively with the canvas, you should know the following:

- The structure of the canvas.
- How to size your forms on the canvas.
- How to position multiple objects.

Structure of the canvas

The canvas contains an invisible grid to which all design objects are referenced.

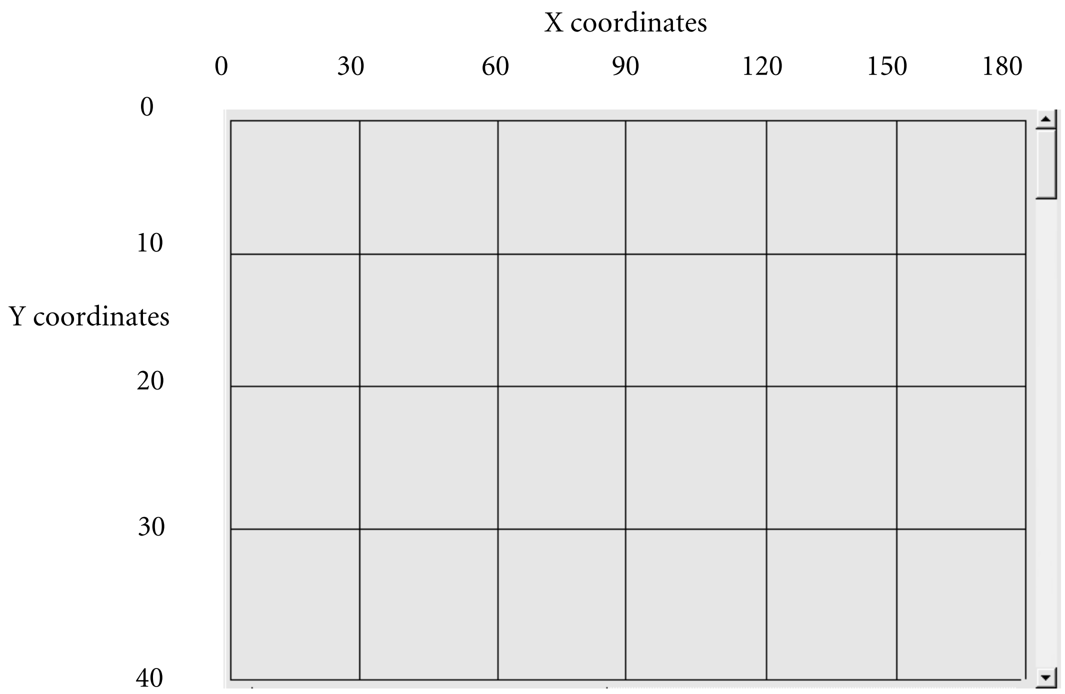


Figure 1-2: The Invisible Grid

Items to note about the invisible grid include:

- Grid coordinates start at (0,0) in the top left corner of the canvas. The X (horizontal) coordinates increase as you move to the right. The Y (vertical) coordinates increase as you move down.
- There is approximately a 3 to 1 ratio between X units and Y units. For example, 30 X units cover approximately the same amount of space as 10 Y units.
- For best results, try to use even X Y coordinates for objects you place on the canvas. The space required to display one character is 2x2 and, overall, the client will display better when you use even coordinates.

Sizing your forms on the canvas

Before creating a form, you should consider the form's ultimate size. Ideally, you will want to create a form that is completely visible on the user's screen so that the user does not have to scroll to view any area of the form.

Tip: Set your screen resolution to that of your users, if you know it, to ensure consistency of display.

The actual drawing canvas extends well beyond the boundaries shown in Figure 1-2 on page 19. However, to size a form so that it is completely visible to the user, stay within the boundaries shown. These boundaries represent the normal viewing area on most user's monitors.

If you need to size a form beyond the normal viewing area, consider extending the form downward. Users are more accustomed to scrolling vertically than they are horizontally.

Positioning multiple objects

- You can use the invisible grid to help you position multiple objects on the drawing canvas. For instance, if you want to position multiple objects with the same left border, be sure all the objects display the same X property. The Y property can be used in a similar way to position objects with the same top border. As already noted, try to use even X Y coordinates for objects you place on the canvas.

Tip: Using the mouse, grab and highlight all the items you want lined up and then enter the X or Y coordinate for all the items.

The Tools Palette

The Tools Palette allows you to create any of 22 design objects. These design objects are the building blocks of your form, each offering a unique look and functionality. Figure 1-3 on page 21 shows the Tools Palette and indicates the design objects that can be created with each tool.

Note: The Pointer allows you to select and modify design objects after they have been created.

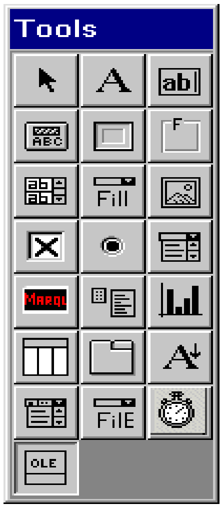


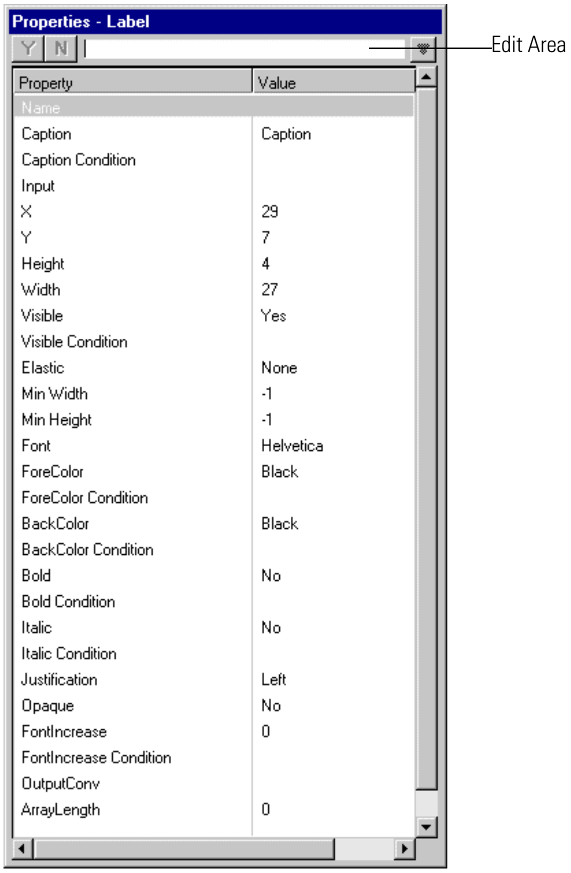
Figure 1-3: The Tools Palette

Refer to *The Tools Palette* on page 41 for a complete description of the tools and the design objects they create.

The Properties Window

When you create or select a design object, the Properties Window becomes populated with all the properties that apply to that object. Some of the properties that appear are common to all design objects, while others are specific only to that object. Use the edit area at the top of the Properties Window to set values for each property to give the object the attributes you desire.

Figure 1-4 on page 22 shows an example Properties Window for a Label object.

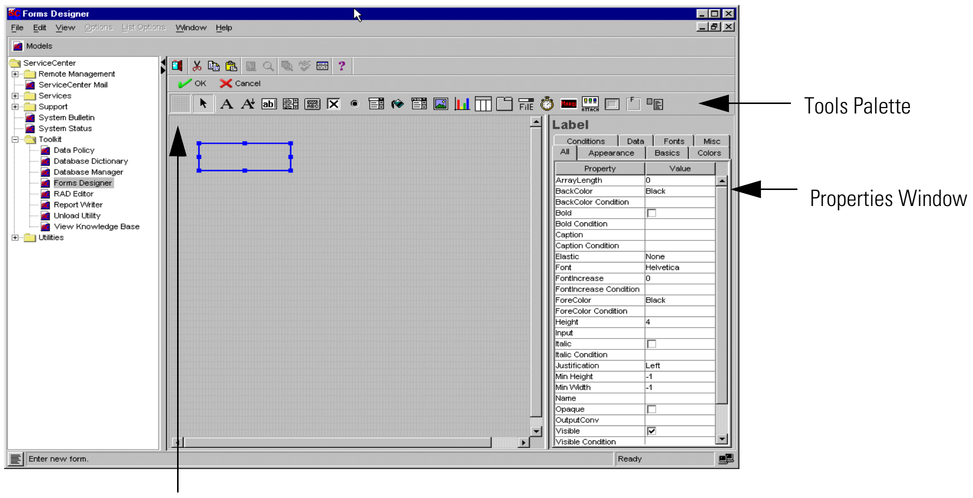


Property	Value
Name	
Caption	Caption
Caption Condition	
Input	
X	29
Y	7
Height	4
Width	27
Visible	Yes
Visible Condition	
Elastic	None
Min Width	-1
Min Height	-1
Font	Helvetica
ForeColor	Black
ForeColor Condition	
BackColor	Black
BackColor Condition	
Bold	No
Bold Condition	
Italic	No
Italic Condition	
Justification	Left
Opaque	No
FontIncrease	0
FontIncrease Condition	
OutputConv	
ArrayLength	0

Figure 1-4: Properties Window for a Label

The Java Client Forms Designer

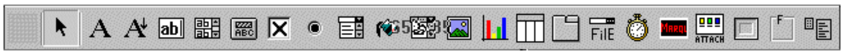
The Java client presents the Forms Designer module and tools in a different manner than the Windows client. The tools and features are the same, only the presentation differs. Users cannot open multiple frames using the Java client Forms Designer.



Property	Value
ArrayLength	0
BackColor	Black
BackColor Condition	
Bold	☐
Bold Condition	
Caption	
Caption Condition	
Elastic	None
Font	Helvetica
FontIncrease	0
FontIncrease Condition	
ForeColor	Black
ForeColor Condition	
Height	4
Input	
Italic	☐
Italic Condition	
Justification	Left
Min Height	-1
Min Width	-1
Name	
Opaque	☐
OutputConv	
Visible	☑
Visible Condition	

Click the **Grid** button to overlay a Grid on the forms design area. Click the **Grid** button again to remove the overlay.

Tools Palette



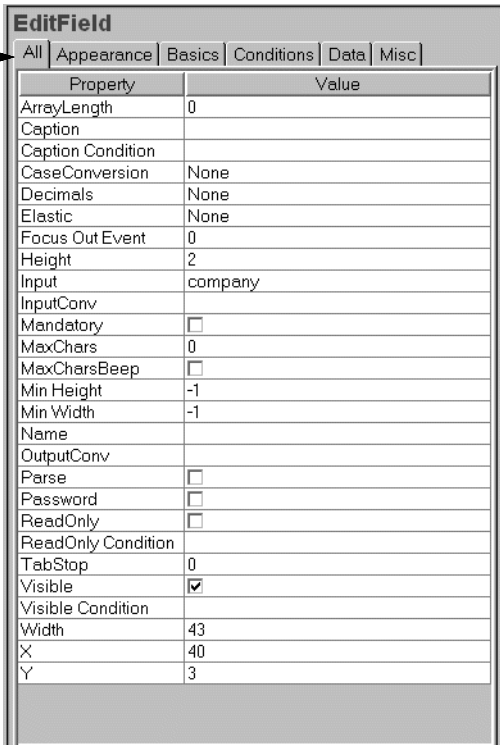
When you click the **Design** button from the Forms Design menu, the Java client opens the selected form and displays the Tools Palette along the top of the client in a menu bar, as opposed to opening the Tools Palette window. Placing an object on the design area will open the Properties Window for that tool in a pane to the right of the design space. The Java client Forms Designer includes a grid overlay tool. Click the **Grid** button to overlay the design area with a grid. Click the button again to remove the overlay.

Java Client Properties Window

The Properties Window for the Java client appears to the right of the screen and is presented in a tabbed format. The tabs group properties by function, with some functions appearing on multiple tabs. Click on a tab to select it and display the options for that tab. The options for each property are the same as the Windows client. Properties that behave differently than the Windows client are highlighted in *Design Tools Unique to the Java Client* on page 25.

Click the All tab to see all properties.

To see only properties that relate to certain aspects of the design object, click on a tab. For instance, to only see controls that relate to design and appearance, click the Appearance tab. Some properties appear under multiple tabs.

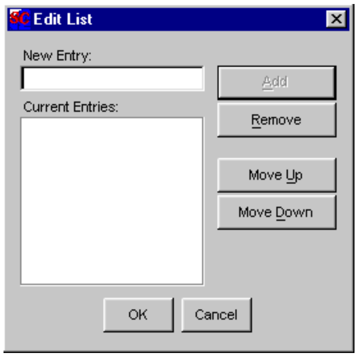


Property	Value
ArrayLength	0
Caption	
Caption Condition	
CaseConversion	None
Decimals	None
Elastic	None
Focus Out Event	0
Height	2
Input	company
InputConv	
Mandatory	☐
MaxChars	0
MaxCharsBeep	☐
Min Height	-1
Min Width	-1
Name	
OutputConv	
Parse	☐
Password	☐
ReadOnly	☐
ReadOnly Condition	
TabStop	0
Visible	☑
Visible Condition	
Width	43
X	40
Y	3

Design Tools Unique to the Java Client

Value List Editing

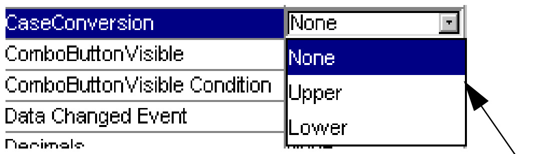
When you click an entry in the Properties Window that takes a list input, Display List for example, an Edit List dialog will open. The cursor focus for the dialog box is on the **New Entry** field by default, so you can begin typing as soon as the dialog appears. Click the **Add** button to add each entry to the list, or simply press the Enter key. You can also change an entry's position in the list by highlighting the entry and then clicking either **Move Up** or **Move Down**.



The Java Client Edit List dialog

Drop-down Selection

Options that have multiple values available to select from will appear in a drop-down list form.



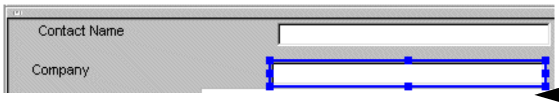
Drop -down list for case conversion

Justification

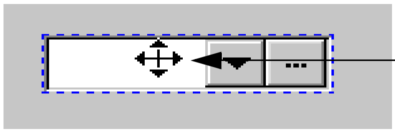
Text and label fields in the Java client Forms Designer can be assigned a justification value using the Justification drop-down menu in the Java client Forms Designer properties box.

Working with Design Objects

In the Java client Forms Designer, placing your mouse on a design object and highlighting it puts a blue box around the object, which you can then grab and stretch. The Java client offers more of these grab points than the Windows client.



Selected objects are highlighted with a blue background. Grab and stretch an object using the mouse.



Click and hold the object a second time to cause the movement cursor to appear.

To move an object:

1. Click the object to select it
2. Click and hold the object a second time. The movement cursor appears.
3. While still holding down the mouse button, move the object to the desired location on the design space.

Designing Forms

The Java client Forms Design process is very similar to that of the Windows client, with some small differences.

To place on an object on the work space

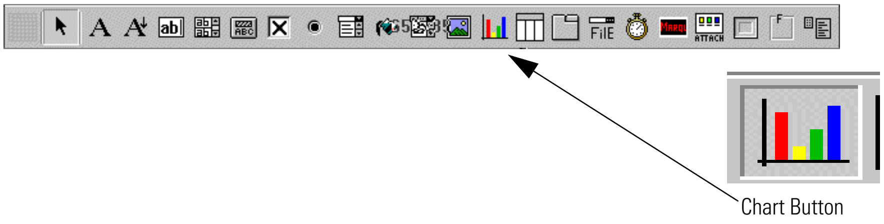
1. Select the object from the Tools palette.
2. Click where in the design area you want the object. The point you click will become the center point for the object.
3. If necessary, stretch the object to the desired size and placement.

   The Properties window for the object will not appear until you select the object.

Example

To place a Chart object on a format:

1 Click the **Chart** button from the Java Client Forms Designer Tool Bar.



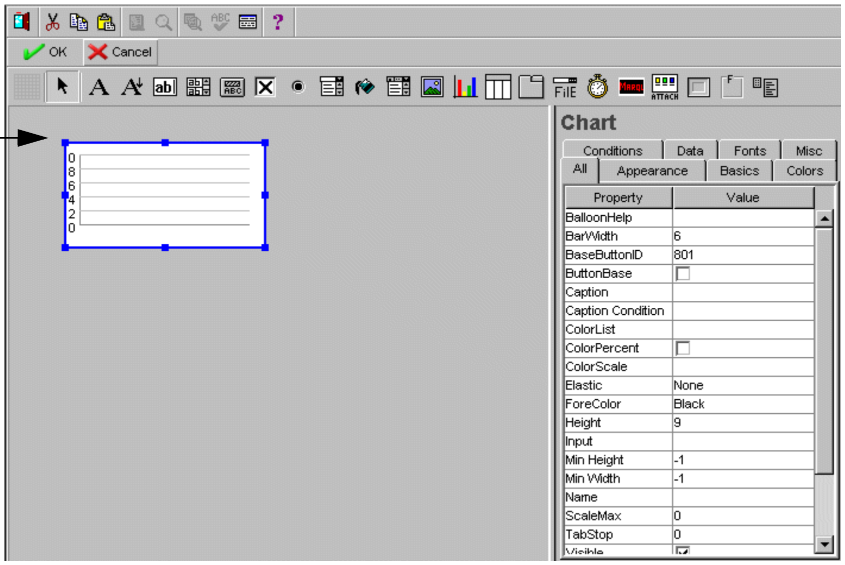
2 Place the Chart object on the form. You can:

a click in the design area. The point you click in the design area will be the center point of the object. After the object is on the design area, highlight the object and drag it to the desired dimensions.

b drag an outline to form the dimensions of the object.

3 Click and highlight the chart on the desktop to bring up the Properties Window and resize, if desired.

Chart object placed in the design space.



Property	Value
BalloonHelp	
BarWidth	6
BaseButtonID	801
ButtonBase	☐
Caption	
Caption Condition	
ColorList	
ColorPercent	☐
ColorScale	
Elastic	None
ForeColor	Black
Height	9
Input	
Min Height	-1
Min Width	-1
Name	
ScaleMax	0
TabStop	0
Visible	☑

4 Adjust the properties of the chart using the Properties Window.

5 Save and close the form.

2 CHAPTER The Forms Design Process

Accessing Forms Designer

To start Forms Designer:

1. From the main menu, click the **Toolkit** tab.

   The selections available in the Toolkit are displayed.

2. Click the **Forms Designer** button.

   The initial Forms Designer screen appears.

3. Choose to create a new form or search for an existing form.

   - To create a new form, enter a name for the form and then click **New**.

   Note: If you do not enter a name for the new form, you will be prompted for a name when you save the form.

   - To search for an existing form, enter the name of the form and press Enter. If you are not sure of the form's name, enter your best guess and click **Search**.

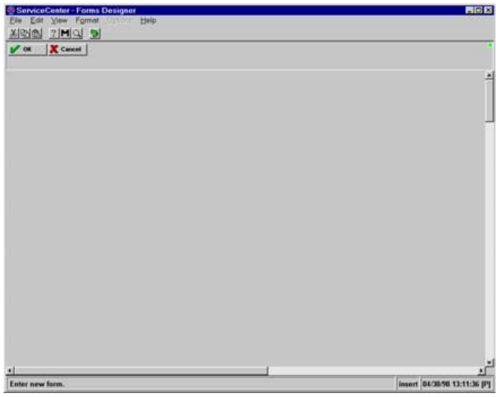
   As an example, click **New** to create a new form.

4. The system asks if you want to use the Form Wizard to create your new form.

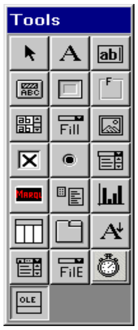
   - In this example, choose to not use the Form Wizard by clicking No.

5 A Drawing Canvas, Properties Window, and Tools Palette appear.

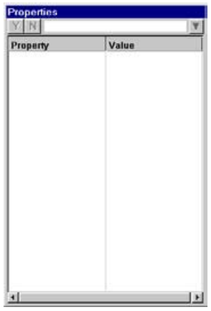
Drawing Canvas


ServiceCenter - Forms Designer
File Edit View Format Options Help
OK
Cancel
Enter new form.
Insert 04/30/98 13:11:36 [P]

Tools Palette



Properties Window



With the Drawing Canvas, Tools Palette, and Properties Window you can begin to design and create ServiceCenter forms.

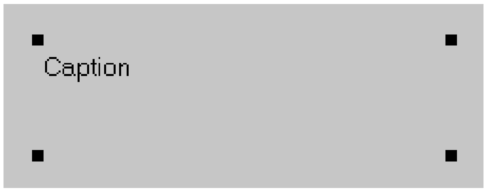
Preview of the Forms Design Process

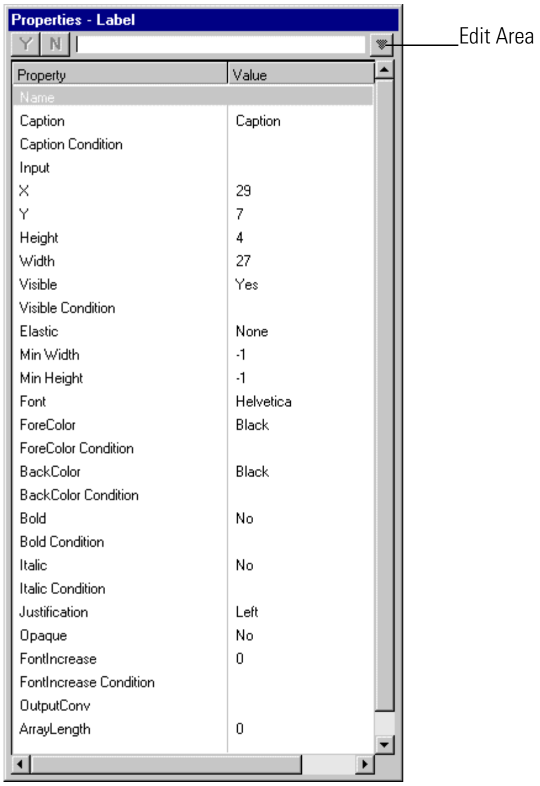
To give you an idea of the forms design process, we will use the construction of a *label* as an example.

1 Select the label tool by clicking on the label icon in the Tools Palette.

2 Draw a label by holding down the left mouse button and dragging the cursor across the Drawing Canvas. Release the mouse button to create the label.

The label and the corresponding Properties Window are displayed on the Drawing Canvas.





Property	Value
Name	
Caption	Caption
Caption Condition	
Input	
X	29
Y	7
Height	4
Width	27
Visible	Yes
Visible Condition	
Elastic	None
Min Width	-1
Min Height	-1
Font	Helvetica
ForeColor	Black
ForeColor Condition	
BackColor	Black
BackColor Condition	
Bold	No
Bold Condition	
Italic	No
Italic Condition	
Justification	Left
Opaque	No
FontIncrease	0
FontIncrease Condition	
OutputConv	
ArrayLength	0

3 The default caption for a new label is Caption. Enter the text you want displayed in the label by clicking on the word Caption in the body of the Properties Window (this enables the edit area for that property) and type your new text. For example, type New Label.

4 Click **Y** or press Enter.

The New Label caption is displayed on the Drawing Canvas and in the Properties Window. To modify your label's position on the canvas, font characteristics, and so forth, make changes to the Properties Window as described in *The Properties Window* on page 33.

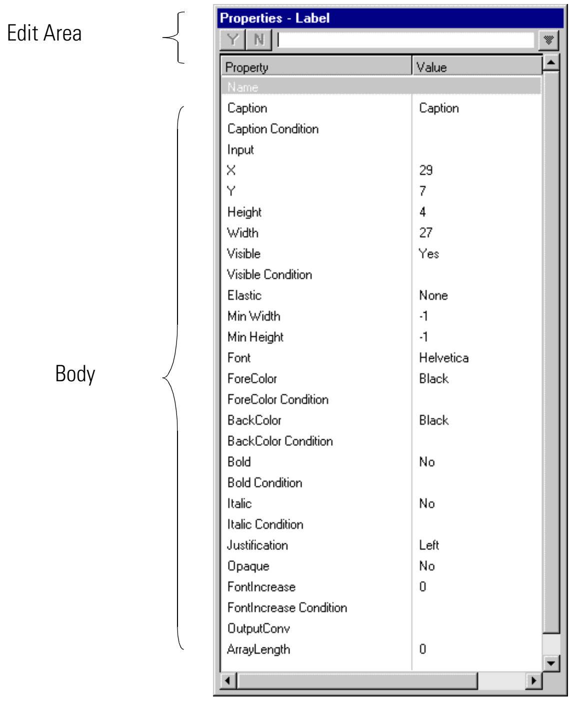
This is the basic process for creating design objects in Forms Designer. The next chapter describes the types of properties that appear in the Properties Window and how to set property values.

3 CHAPTER The Properties Window

The Properties Window is used to set property values for each design object on your format. The values you set determine how the object appears on the form and how the object interrelates with the user and with the ServiceCenter database. For an overview of properties and how the values you enter on this panel operate, please see *Dynamic View Dependencies* on page 127.

Using the Properties Window

The Properties Window is divided into two parts: the Body and the Edit Area. The Body is where you select a property. The Edit Area is where you enter a value for the property.



Property	Value
Name	
Caption	Caption
Caption Condition	
Input	
X	29
Y	7
Height	4
Width	27
Visible	Yes
Visible Condition	
Elastic	None
Min Width	-1
Min Height	-1
Font	Helvetica
ForeColor	Black
ForeColor Condition	
BackColor	Black
BackColor Condition	
Bold	No
Bold Condition	
Italic	No
Italic Condition	
Justification	Left
Opaque	No
FontIncrease	0
FontIncrease Condition	
OutputConv	
ArrayLength	0

Sizing the Properties Window

To size the Properties Window, place the mouse cursor near the outside border of the Properties Window and the cursor will change to a sizing tool. Modify the window size by holding down the left mouse button, dragging the edge or corner of the window until the proper size is achieved, and then releasing the mouse button.

The Property and Value columns can also be modified in this way.

Selecting a property

The Body of the Properties Window is the area from which you select a property. Note that this is a read-only area.

1 Select (highlight) a property by clicking on the property (or after you have clicked on a property, you can use the arrow keys to move up and down the Properties Window). You can also use the scroll button located on the right side of the Properties Window to view any properties that are not currently visible.

2 The Edit Area becomes enabled for that property. You are now ready to enter a value.

Entering a value manually

The Edit Area of the Properties Window is where you enter a value for the property. Once you have selected a property, the edit area becomes enabled for that property.

1 Type the new value. The new value is displayed in the Edit Area, but not in the property field.

2 Press Enter or click **Y** to set the new value for the design object. Clicking on another property will also set the new value. The new value is now displayed in the property field.

If you want to restore the original value of the property (and delete the new value), click **No**. Clicking anywhere outside the Properties Window will also restore the original value.

Entering a Value from the Edit Area list

For some properties, such as Font, the Edit Area allows you to select from a drop-down list. You will know when a property has a drop-down list because the List button will change from shaded to black.

1 Click the **List** button.

2 Make your selection from the drop-down list by clicking on the selection or by using the arrow keys.

3 Press Enter or click **Y** (twice if using the arrow keys).

For a shortcut, you can select from a drop-down list as follows:

1 Select a property that has a drop-down list.

2 Type the first letter of the selection. For instance, with a Font property you could type *T* for *Times*. Note that the letter you type is case sensitive. All fonts begin with a capital letter.

3 Press Enter or click **Y**.

Understanding Properties

When you look at the properties that are displayed for a design object, think of them in terms of the following categories:

- Common properties
- Font properties
- Box properties
- Object specific properties

Note: All character dimensions (coordinates for placement on the screen, height, width) should be even numbers, because the space required to display one character is 2x2. Placement coordinates are also impacted, in addition to height and width, and should also be kept even.

Also, when the ArrayLength of a field is set to be a number greater than 0 and the field is situated at an odd coordinate, it will automatically be shifted to an even coordinate.

Common properties

The first 10 properties in the Properties Window are common to all design objects. These properties are:

Property	Value
Name	Enter a unique identifier to associate a name with the component on the screen. This name is used by external applications such as RAD to change the properties of the component dynamically. This property is optional.
Caption	The Caption property allows you to enter text for design objects that are capable of displaying text. For instance, if the object is a label, enter the text for your label in this property field. The default value is *Caption*, which you replace with your own text.
Caption Condition	Enter a condition for evaluating the Caption property at runtime. The result of this evaluation is used to override the value assigned to the property.
Input	The Input property allows you to enter the name of the appropriate dbdict field or variable that corresponds to the design object.
X	The X property refers to the position of the object on the X (horizontal) axis of the alignment grid. The value corresponds to the left edge of the object.
Y	The Y property refers to the position of the object on the Y (vertical) axis of the alignment grid. The value corresponds to the top edge of the object.
Height	The Height property refers to the height of the object in alignment grid units.
Width	The Width property refers to the width of the object in alignment grid units.
Visible	Indicates whether the property is visible or not. A property can be created and used as a reference point rather than displaying it on a form.
Visible Condition	Can be used to establish a condition under which a field may or may not be visible.

Font Properties

Font Properties allow you to modify the appearance of text that is used in the design object. These properties are only displayed for Labels and Frames.

Property	Value
Font	You can select one of seven fonts available from the drop-down list. The default font is Helvetica.
ForeColor	Text color. You can select one of 17 colors available from the drop-down list. The default color is Blue.
BackColor	Text background color. You can select one of 17 colors available from the drop-down list. Note that the Opaque property must be set to *Yes* in order for the BackColor to have effect.
Bold	Applies **bold** to the text.
Italic	Applies *italics* to the text.
Justification	Justifies the text to the Left, Right, or Center. Select from the drop-down list.
Opaque	Enables the BackColor property.
FontIncrease	Font size is initially set to a standard size. To increase or decrease this standard size, change the numeric value of FontIncrease. The default for FontIncrease is 0 (which means no change to the standard font size). Increase font size by making the value larger than 0. Decrease font size by making the value less than 0.

Box Properties

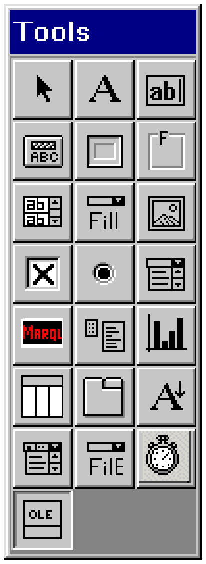
Box Properties apply to any box design object: text box, multi-line text box, fill box, or combo box.

Property	Value
TabStop	When a user tabs through a form, objects are selected in a certain order. If all objects have the default TabStop setting of 0, then each object is visited based on its top-to-bottom, left-to-right position on the form. You can alter the tab order for an object using the following guidelines: - An object with TabStop = -1 is skipped during tabbing. - An object with TabStop = 1 is given focus when the screen is first displayed. - All objects numbered with a non-zero TabStop are tabbed to in increasing TabStop order. Objects with duplicate TabStops are tabbed in a top-to-bottom, left-to-right order. - After all objects with TabStops > 0 are visited, tabbing proceeds to all objects with TabStops of 0. These objects are visited in top-to-bottom, left-to-right order. - ReadOnly objects are skipped during tabbing.
ReadOnly	Only allows the user to view the field.
Password	Makes the user's password entry invisible on the screen.
MaxChars	Specifies the number of characters the user can enter in the field. The default is 0, which means an unlimited number of characters can be entered.
MaxCharsBeep	Notifies the user (by sounding a beep) that the maximum number of characters has been reached.
CaseConversion	Changes the case of any text entered in the field. You can select None (allows upper- and lowercase), Upper, or Lower from the drop-down list.
Decimals	Keeps numeric values from having several numbers to the right of the decimal point (for example, 5.00375). You can select None (allows the long number), 0.0, or 0.00.
Parse	If Parse is set to *Yes*, the text entered in a field is parsed to verify that it is of the correct syntax. The syntax for the field is defined by the type of field it is displaying (date, expression, number, and so forth).

4 CHAPTER The Tools Palette

The Tools Palette allows you to create any of 22 design objects. These objects are the building blocks of your forms, each offering a unique look and functionality.

Figure 4-1 on page 42 shows the Tools Palette and indicates the design objects that can be created with each tool. The Pointer allows you to select and modify design objects after they have been created.



Pointer	Label	Text
Button	Bevel Frame	Frame
MultiText box	Fill	Graph
CheckBox	Radio Button	ComboBox
Marquee	SubFormat	Chart
Table	Notebook	Wlabel
ComFill	Gfile	Timer
Attachment container		

Figure 4-1: The Tools Palette

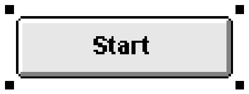
Pointer

The Pointer allows you to select, deselect, and modify design objects.

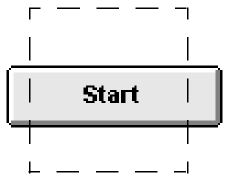
Usage:

To select one object:

Click on the object. Grab points are displayed, indicating that the object is selected.



Or,hold down the left mouse button and drag a lasso across any part of the object. After lassoing the object, handles are displayed.



To select multiple objects:

Hold down the **Shift** key and click on multiple objects one at a time. Grab points are displayed on each object. The properties box is enabled for the last object selected.

Or, drag and release a lasso across any part of the desired objects. After lassoing the objects, handles are displayed on each object.

To deselect one object:

Hold down the **Shift** key and click on the object. Grab points disappear, indicating that the object is no longer selected.

To deselect all objects:

Click on a blank area of the drawing canvas.

To position one object:

Click on the object and continue holding down the left mouse button. Drag the object to the desired position.

To resize an object:

Select the object. Drag the object handles to resize the object.

To delete one or more objects:

Select the object(s). Press the **Delete** key.

Label

Description:A Label is a single line of text you can use to give titles to forms, give labels to objects within the form, or otherwise place text on the form.

For information on using the label tool to create data fields, see *More About Labels* on page 46.

Property	Value
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	Enter the text for your label in this property.
Caption Condition	Enter a formula that will override the **Caption** property, if the conditions of the formula are met.
Input	No Input is required for a label, unless you are creating a field that will display data.
X	Horizontal position based on the left edge of the object.
Y	Vertical position based on the top edge of the object.
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.
Visible Condition	Enter a formula that will override the **Visible** property, if the conditions of the formula are met.

Property	Value
Elastic	Determines the width and height of an object in relation to its parent window. This property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.
Font	Select a font for your label from the drop-down list.
ForeColor	Text color. Select from 17 available colors.
ForeColor Condition	Enter a formula for evaluating the ForeColor property at run time. The result of this evaluation is used to override the value assigned to the property.
BackColor	Background color. The Opaque property must be set to *Yes* in order for the BackColor to have effect.
BackColor Condition	Enter a formula that will override the **BackColor Condition** property, if the conditions of the formula are met.
Bold	When set to *Yes*, applies **bold** to your label.
Bold Condition	Enter a formula that will override the **Bold Condition** property, if the conditions of the formula are met.
Italic	When set to *Yes*, applies *italics* to your label.

Property	Value
Italic Condition	Enter a formula that will override the **Italic** property, if the conditions of the formula are met.
Justification	You can justify your label to the Left, Right, or Center.
Opaque	Enables the BackColor property.
FontIncrease	Allows you to increase or decrease the standard font size.
FontIncrease Condition	Enter a formula that will override the **FontIncrease** property, if the conditions of the formula are met.
OutputConv	This property does not apply to labels.
ArrayLength	When a screen object is associated with an array data structure, this property specifies the size of the scrolling region that is used to view array entries. For example, if a field is assigned an Array Length of 5, the form will stack five fields vertically to allow viewing of five array entries. A scroll bar appears beside the fields to allow viewing of the array entries. If Array Length is set to -1, the form dynamically creates as many fields as there are entries in the array. If the Input data type is scalar, only a single Text Box is displayed. The default is 0, which means one vertical line of information is displayed.

More About Labels

The Label tool can also be used to create data fields in a form that contains a Query By Example (QBE) list.

A QBE list contains fields that display data from the ServiceCenter database.

To create data fields, use the Label tool as follows:

1 Select the Label tool and place a label object on the form.

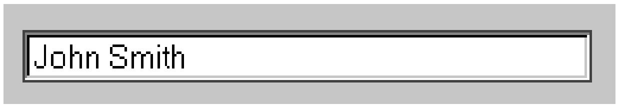
2 Enter an Input property. For example, for the data field under **Category** you would enter *header,category*.

3 Leave the Caption property blank.

Text

Description: The Text object displays the contents of a database field or variable, and conditionally allows the user to enter or modify its contents.

Example:



Property	Value
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	This property does not apply to text boxes.
Caption Condition	Enter a formula that will override the **Caption** property, if the conditions of the formula are met. Does not apply to text boxes.
Input	Enter the name of the appropriate database field or variable.
X	Horizontal position based on the left edge of the object.
Y	Vertical position based on the top edge of the object.
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.
Visible Condition	Enter a formula that will override the **Visible** property, if the conditions of the formula are met.

Property	Value
Elastic	Determines the width and height of an object in relation to its parent window. This property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.
TabStop	You will almost always use the default of *0* for TabStop. *Box Properties* on page 39 for more information on tabstops.
ReadOnly	Prohibits the user from changing the contents of the field.
ReadOnly Condition	Enter a formula that will override the **ReadOnly** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Password	Setting this to **Yes** makes the text entered in the field a series of asterisks on the screen. The data being displayed is not encrypted, just transposed
MaxChars	Specifies the number of characters the user can enter in the field. The default is 0 (unlimited).
MaxCharsBeep	If set to *Yes*, the system notifies the user (by sounding a beep) that the maximum number of characters has been reached (not available on UNIX systems).
CaseConversion	Changes the *case* of text entered in the field.

Property	Value
Decimals	Modifies numbers entered in the field to a number of decimal points you specify.
Parse	If Parse is set to *Yes*, the text entered in a field is parsed to verify that it is of the correct syntax. The syntax for the field is defined by the type of field it is displaying (date, expression, number, and so forth).
Focus Out Event	When this property is set to *true*, the event value is sent to the display RAD application when the field loses focus (when you leave the field). Refer to the RAD documentation for further information.
Mandatory	Setting Mandatory to true indicates that the field is required, and flags the top left of the field with a small red indicator. **This is a visible change only**: In order to make the field mandatory for any form it appears on, use Data Policy. To make the field mandatory for a small number of forms only, use Format Control.
AccessibleName	Allows names to be assigned to the component for use with accessibility software. This function is only supported by the Java client. This property is not required. If no name is present, the Java client defaults to looking for a nearby label to use for the name of the component. For buttons, check boxes or radio buttons, it will use the component's text. Accessibility software can query the Java Client to get this information, and use it to present it to the user in a variety of ways. For example, speech simulation software will say the name and the type of component when the component gets focus.
AccessibleDescription	Allows descriptions to be assigned to the component for use with accessibility software. This function is only supported by the Java client. This property is not required. If no description is present, the Java client defaults to looking for a nearby label to use for the name of the component. For buttons, check boxes or radio buttons, the Java client will use the component's text. Accessibility software can query the Java Client to get this information, and use it to present it to the user in a variety of ways. For example, speech simulation software will say the name and the type of component when the component gets focus, based on the settings in the AccessibleName and AccessibleDescription fields.

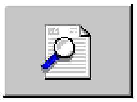
Property	Value
ArrayLength	When a screen object is associated with an array data structure, this property specifies the size of the scrolling region that is used to view array entries. For example, if a field is assigned an Array Length of 5, the form will stack five fields vertically to allow viewing of five array entries. A scroll bar appears beside the fields to allow viewing of the array entries. If Array Length is set to -1, the form dynamically creates as many fields as there are entries in the array. If the Input data type is scalar, only a single Text object is displayed. The default is 0, which means one vertical line of information is displayed.
InputConv	Does not apply to most text boxes you will create. This property is used for RAD subroutines that are designed to mask the display of data on an input field, or check and validate the entry of data into an input field. For instance, if the Text object displays date/time information, entering *input.time* in the InputConv property will append a time entered in the field to the current date.
OutputConv	Same as InputConv, but applies to output values. If you use Output conversion, you must use input conversion.

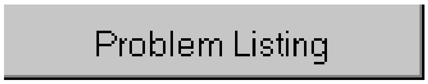
Button

Description:A Button activates a Control ID when clicked. You can use text or graphics, or both, to customize buttons.

Note: Do not use 888 as a button ID. 888 is a reserved number that calls XVT_EVENT_FINISH and will cause the client to exit.

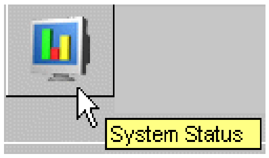
Examples:





Property	Value
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	If you want a caption on the button, enter it in this property.
Caption Condition	Enter a formula that will override the **Caption** property, if the conditions of the formula are met.
Input	Enter the name of the appropriate dbdict field or variable.
X	Horizontal position based on the left edge of the object.
Y	Vertical position based on the top edge of the object
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.
Visible Condition	Enter a formula that will override the **Visible** property, if the conditions of the formula are met.
Elastic	Determines the width and height of an object in relation to its parent window. This property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.

Property	Value
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.
TabStop	You will almost always use the default of *0* for TabStop. *Box Properties* on page 39 for more information on tabstops.
ButtonID	Numeric. Specifies an Option ID to transmit to a RAD *rio* or *fdisp* panel when clicked or corresponding display option, if the format has an associated display option. For example, 3 corresponds to **F3**. 0 corresponds to **Enter**. Refer to the *ServiceCenter RAD Guide* for more details.
ButtonID Condition	Enter a formula that will override the **ButtonID** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Justification	You can justify your label to the Left, Right, or Center.
Animated	Enables the looping display of two or more images specified by the Bitmaps property.
Bitmaps	The name(s) of one or more Microsoft Windows bit mapped image files (.BMP file format) to display on the button. ■ Bitmaps must reside in the path specified by `bitmap_path:` in the `sc.ini` file. You do not need to include the .BMP file suffix for this property. ■ Use a semicolon (;) to delimit multiple file names for use in animation sequences: `sc0;sc1;sc2;sc3;sc4` ■ This example sets up an animation loop that consists of five bitmaps. For tips on using images in ServiceCenter, refer to the *Graphics and Animation* on page 138.

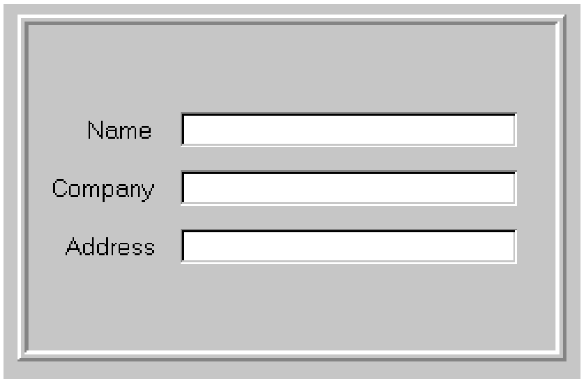
Property	Value
Push Bitmap	The name of a Microsoft Windows bit-mapped image (.BMP file format) to display when the button is pressed. Bitmaps must reside in the path specified by bitmap_path: in the *sc.ini* file. You do not need to include the .BMP file suffix for this property. For tips on using images in ServiceCenter, refer to the *Graphics and Animation* on page 138.
Speed	Integer. Delay in milliseconds between animation frames. Small values speed up animations. The Microsoft Windows minimum cycle time is 55.
BalloonHelp	A popup help description that ServiceCenter displays while the cursor is over a button.  Balloon Help is limited to 80 characters.
Bitmap and Caption	If this property is set to *true*, the button displays its bitmap and caption simultaneously. The placement of the caption depends on the value of the Justification property.

Property	Value
AccessibleName	Allows names to be assigned to the component for use with accessibility software. This function is only supported by the Java client. This property is not required. If no name is present, the Java client defaults to looking for a nearby label to use for the name of the component. For buttons, check boxes or radio buttons, it will use the component's text. Accessibility software can query the Java Client to get this information, and use it to present it to the user in a variety of ways. For example, speech simulation software will say the name and the type of component when the component gets focus.
AccessibleDescription	Allows descriptions to be assigned to the component for use with accessibility software. This function is only supported by the Java client. This property is not required. If no description is present, the Java client defaults to looking for a nearby label to use for the name of the component. For buttons, check boxes or radio buttons, the Java client will use the component's text. Accessibility software can query the Java Client to get this information, and use it to present it to the user in a variety of ways. For example, speech simulation software will say the name and the type of component when the component gets focus, based on the settings in the AccessibleName and AccessibleDescription fields.

Bevel

Description:A BevelFrame is a cosmetic rectangular border. Use bevel frames to make your forms more attractive or easier to understand.

Example:



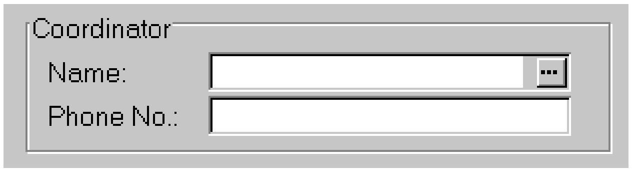
Property	Value
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	This property does not apply to the bevel frame object.
Caption Condition	Enter a formula that will override the **Caption** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Input	Enter the name of the appropriate database field or variable.
X	Horizontal position based on the left edge of the object.
Y	Vertical position based on the top edge of the object
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.

Property	Value
Elastic	Determines the width and height of an object in relation to its parent window. This property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.
Opaque	*Yes* or *No*. *Yes* forms an opaque rectangle within the bevel boundaries—if the bevel is placed over objects that were created previously, those objects are obscured. *No* forms only the bevel—objects behind the bevel are visible.
OuterBevelWidth	Integer. Width of the outer bevel. Default is *2*.
InnerBevelWidth	Integer. Width of the inner bevel. Default is *2*.
BackColor	Background color. The Opaque property must be set to *Yes* in order for the BackColor to have effect.

Frame

Description:A Frame is a rectangular border that is used to group logically associated items.

Example:



This example shows a frame with the caption *Coordinator*. The frame has been added to group fields which give information about the coordinator.

Property	Value
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	If you want a caption on the frame, enter it in this property.
Caption Condition	Enter a formula that will override the **Caption** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Input	Enter the name of the appropriate database field or variable.
X	Horizontal position based on the left edge of the object.
Y	Vertical position based on the top edge of the object
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.
Visible Condition	Enter a formula that will override the **Visible** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.

Property	Value
Elastic	Determines the width and height of an object in relation to its parent window. This property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.
Font	Select a font for the frame caption from the drop-down list.
ForeColor	Text color. Choose one of 17 colors available from the drop-down list.
ForeColor Condition	Enter a formula that will override the **ForeColor** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
BackColor	Background color. The Opaque property must be set to Yes in order for the BackColor to have effect.
BackColor Condition	Enter a formula that will override the **BackColor** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Bold	When set to *Yes*, applies **bold** to your frame caption.

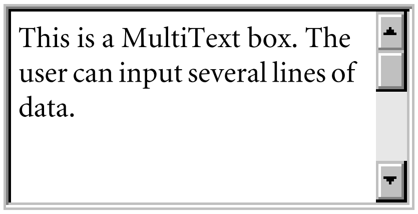
Property	Value
Bold Condition	Enter a formula for evaluating the Bold property at run time. The result of this evaluation is used to override the value assigned to the property.
Italic	When set to *Yes*, applies *italics* to your frame caption.
Italic Condition	Enter a formula that will override the **Italic** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Justification	You can justify your frame caption to the Left, Right, or Center.
Opaque	Enables the BackColor property.
FontIncrease	Allows you to increase or decrease the standard font size.
FontIncrease Condition	Enter a formula that will override the **FontIncrease** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
OutputConv	This property does not apply to frames.
ArrayLength	When a screen object is associated with an array data structure, this property specifies the size of the scrolling region that is used to view array entries. For example, if a field is assigned an Array Length of 5, the form will stack five fields vertically to allow viewing of five array entries. A scroll bar appears beside the fields to allow viewing of the array entries. If Array Length is set to -1, the form dynamically creates as many fields as there are entries in the array. If the Input data type is scalar, only a single Text object is displayed. The default is 0, which means one vertical line of information is displayed.
ShadowStyle	Changes the appearance of the border. You can select Inset or Raised—two variations of a shaded border. The example shown on the previous page has a ShadowStyle of Inset.

MultiText Box

Description:A MultiText box allows the user to input several lines of data. This object contains scroll bars and allows text wrapping.

Note: A MultiText Box field must be defined as an array of characters in the Database Dictionary.

Example:



This is a MultiText box. The user can input several lines of data.

Property	Value
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	If you want a caption on the MultiText box, enter it in this property.
Caption Condition	Enter a formula that will override the **Caption** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Input	Enter the name of the appropriate database field or variable.
X	Horizontal position based on the left edge of the object.
Y	Vertical position based on the top edge of the object
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.

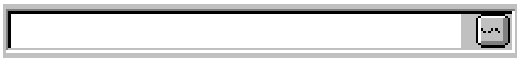
Property	Value
Visible Condition	Enter a formula that will override the **Visible** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Elastic	Determines the width and height of an object in relation to its parent window. This property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.
TabStop	You will almost always use the default of *0* for TabStop.
ReadOnly	Prohibits the user from changing the contents of the field.
ReadOnly Condition	Enter a formula that will override the **ReadOnly** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Password	This property does not apply to MultiText boxes.
MaxChars	This property does not apply to MultiText boxes.
MaxCharsBeep	This property does not apply to MultiText boxes.
CaseConversion	Changes the *case* of text entered in the field.

Property	Value
Decimals	This property does not apply to MultiText boxes.
Parse	If Parse is set to *Yes*, the text entered in a field is parsed to verify that it is of the correct syntax. The syntax for the field is defined by the type of field it is displaying (date, expression, number, and so forth).
Focus Out Event	When this property is set to *true*, the event value is sent to the RAD application when the field loses focus (when you leave the field). Refer to the RAD documentation for further information.
Mandatory	Setting Mandatory to true indicates that the field is required, and flags the top left of the field with a small red indicator. **This is a visible change only**: In order to make the field mandatory for any form it appears on, use Data Policy. To make the field mandatory for a small number of forms only, use Format Control.
AccessibleName	Allows names to be assigned to the component for use with accessibility software. This function is only supported by the Java client. This property is not required. If no name is present, the Java client defaults to looking for a nearby label to use for the name of the component. For buttons, check boxes or radio buttons, it will use the component's text. Accessibility software can query the Java Client to get this information, and use it to present it to the user in a variety of ways. For example, speech simulation software will say the name and the type of component when the component gets focus.
AccessibleDescription	Allows descriptions to be assigned to the component for use with accessibility software. This function is only supported by the Java client. This property is not required. If no description is present, the Java client defaults to looking for a nearby label to use for the name of the component. For buttons, check boxes or radio buttons, the Java client will use the component's text. Accessibility software can query the Java Client to get this information, and use it to present it to the user in a variety of ways. For example, speech simulation software will say the name and the type of component when the component gets focus, based on the settings in the AccessibleName and AccessibleDescription fields.

Fill Box

Description:A Fill Box allows the user to input data by clicking on an ellipsis button and then selecting data from a QBE list. This type of object allows the user to look up and browse information in another file, validity table, and so forth. It can also be used to copy (fill) information from another table to the user's current screen.

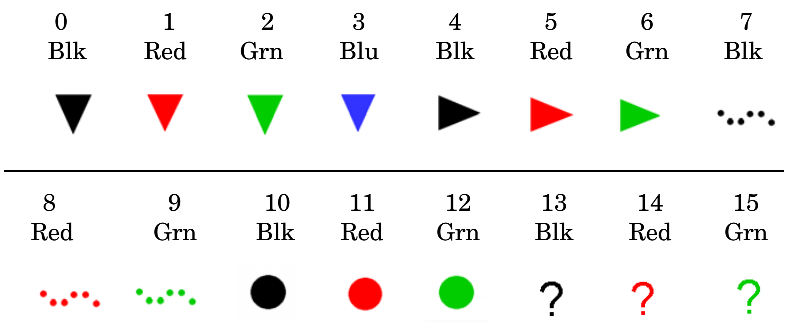
Example:


Fill box shown in design mode.

Property	Value
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	If you want a caption on the fill box, enter it in this property.
Caption Condition	Enter a formula that will override the **Caption** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Input	Enter the name of the appropriate database field or variable.
X	Horizontal position based on the left edge of the object.
Y	Vertical position based on the top edge of the object
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.
Visible Condition	Enter a formula that will override the **Visible** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.

Property	Value
Elastic	Determines the width and height of an object in relation to its parent window. This property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.
TabStop	You will almost always use the default of *0* for TabStop. *Box Properties* on page 39 for more information on tabstops.
ReadOnly	Prohibits the user from changing the contents of the field.
ReadOnly Condition	Enter a formula that will override the **ReadOnly** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Password	This property does not apply to fill boxes.
MaxChars	Specifies the number of characters the user can enter in the field. The default is 0 (unlimited).
MaxCharsBeep	If set to *Yes*, the system notifies the user (by sounding a beep) that the maximum number of characters has been reached.
CaseConversion	Changes the *case* of text entered in the field.
Decimals	Modifies numbers entered in the field to a number of decimal points you specify.

Property	Value
Parse	If Parse is set to *Yes*, the text entered in a field is parsed to verify that it is of the correct syntax. The syntax for the field is defined by the type of field it is displaying (date, expression, number, and so forth).
Focus Out Event	When this property is set to *true*, the event value is sent to the RAD application when the field loses focus (when you leave the field). Refer to the RAD documentation for further information.
Mandatory	Setting Mandatory to true indicates that the field is required, and flags the top left of the field with a small red indicator. **This is a visible change only**: In order to make the field mandatory for any form it appears on, use Data Policy. To make the field mandatory for a small number of forms only, use Format Control.
AccessibleName	Allows names to be assigned to the component for use with accessibility software. This function is only supported by the Java client. This property is not required. If no name is present, the Java client defaults to looking for a nearby label to use for the name of the component. For buttons, check boxes or radio buttons, it will use the component's text. Accessibility software can query the Java Client to get this information, and use it to present it to the user in a variety of ways. For example, speech simulation software will say the name and the type of component when the component gets focus.
AccessibleDescription	Allows descriptions to be assigned to the component for use with accessibility software. This function is only supported by the Java client. This property is not required. If no description is present, the Java client defaults to looking for a nearby label to use for the name of the component. For buttons, check boxes or radio buttons, the Java client will use the component's text. Accessibility software can query the Java Client to get this information, and use it to present it to the user in a variety of ways. For example, speech simulation software will say the name and the type of component when the component gets focus, based on the settings in the AccessibleName and AccessibleDescription fields.
ButtonID	Numeric. Specifies a Option ID to transmit a RAD `rio` or `fdisp` panel when clicked. For example, 3 corresponds to **F3**. 0 corresponds to **Enter**. The default is 9 - the default Option for Fill. Refer to the *ServiceCenter RAD Guide* for more details.

Property	Value
ButtonBitmap	Enter one of 16 possible button and color combinations:  0 Blk 1 Red 2 Grn 3 Blu 4 Blk 5 Red 6 Grn 7 Blk 8 Red 9 Grn 10 Blk 11 Red 12 Grn 13 Blk 14 Red 15 Grn
ArrayLength	When a screen object is associated with an array data structure, this property specifies the size of the scrolling region that is used to view array entries. For example, if a field is assigned an Array Length of 5, the form will stack five fields vertically to allow viewing of five array entries. A scroll bar appears beside the fields to allow viewing of the array entries. If Array Length is set to -1, the form dynamically creates as many fields as there are entries in the array. If the Input datatype is scalar, only a single Text Box is displayed. The default is 0, which means one vertical line of information is displayed.
HideEdit	If the value of the HideEdit property for a Fill object is set to *Yes*, the text in the Fill object cannot be displayed or edited.
InputConv	Does not apply to most fill boxes you will create. This property is used for RAD subroutines that are designed to mask the display of data on an input field, or check and validate the entry of data into an input field. For instance, if the Text object displays date/time information, entering *input.time* in the InputConv property will append a time entered in the field to the current date.
OutputConv	Same as InputConv, but applies to output values. If you use Output conversion, you must use input conversion.

Graphic

Description:A Graphic displays a cosmetic image from a Microsoft Windows BMP file.

Example:

Property	Value
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	If you want a caption on the graphic, enter it in this property.
Caption Condition	Enter a formula that will override the **Caption** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Input	Enter the name of the appropriate database field or variable.
X	Horizontal position based on the left edge of the object.
Y	Vertical position based on the top edge of the object
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.
Visible Condition	Enter a formula that will override the **Visible** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.

Property	Value
Elastic	Determines the width and height of an object in relation to its parent window. This property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.
Bitmap	The name of a Microsoft Windows bit-mapped image file (.BMP file format). The bitmap must reside in the path specified by **bitmap_path**: in the *sc.ini* file. You do not need to include the .BMP file suffix for this property.
Bitmap Condition	Enter a formula that will override the **Bitmap** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
AutoResize	*Yes* or *No*. *Yes* displays the full image regardless of the dimensions of the Picture object. *No* resizes the image at run time to fit the dimensions of the Picture object.
Frame	*Yes* or *No*. *Yes* displays a beveled frame around the image.

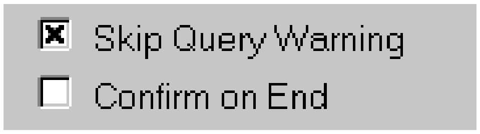
Property	Value
Constrain	*Yes* or *No*. *No* is the default. Used only if AutoResize is turned on. AutoResize allows a bitmap to be stretched or shrunk to fit the rectangular border defined in Forms Designer to hold the bitmap. Setting Constrain to *Yes* will allow the bitmap to be reduced to fit inside the rectangle, but will not expand the bitmap if it is smaller than the rectangle.
ReadOnly	Prohibits the user from changing the contents of the field.
ReadOnly Condition	Enter a formula that will override the **ReadOnly** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.

Check Box

Description:A Check Box is a Boolean (logical) field that can evaluate to true, false, unknown, or null.

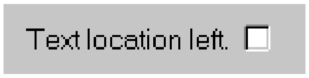
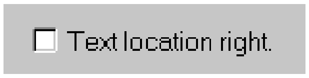
Note: Check boxes are not supported on QBE lists.

Example:



Property	Value
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	If you want a caption on the checkbox, enter it in this property.
Caption Condition	Enter a formula that will override the **Caption** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Input	Enter the name of the appropriate database field or variable.
X	Horizontal position based on the left edge of the object.
Y	Vertical position based on the top edge of the object
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.

Property	Value
Visible Condition	Enter a formula that will override the **Visible** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Elastic	Determines the width and height of an object in relation to its parent window. This property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.
TabStop	You will almost always use the default of *0* for TabStop. *Box Properties* on page 39 for more information on tabstops.
ReadOnly	If set to Yes, then this field prohibits the user from changing the contents of the field.
Read Only Condition	Enter a formula that will override the **Read Only** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Focus Out Event	When this property is set to *true*, the event value is sent to the RAD application when the field loses focus (when you leave the field). Refer to the RAD documentation for further information.

Property	Value
AccessibleName	Allows names to be assigned to the component for use with accessibility software. This function is only supported by the Java client. This property is not required. If no name is present, the Java client defaults to looking for a nearby label to use for the name of the component. For buttons, check boxes or radio buttons, it will use the component's text. Accessibility software can query the Java Client to get this information, and use it to present it to the user in a variety of ways. For example, speech simulation software will say the name and the type of component when the component gets focus.
AccessibleDescription	Allows descriptions to be assigned to the component for use with accessibility software. This function is only supported by the Java client. This property is not required. If no description is present, the Java client defaults to looking for a nearby label to use for the name of the component. For buttons, check boxes or radio buttons, the Java client will use the component's text. Accessibility software can query the Java Client to get this information, and use it to present it to the user in a variety of ways. For example, speech simulation software will say the name and the type of component when the component gets focus, based on the settings in the AccessibleName and AccessibleDescription fields.
TextLocation	*Left* or *Right*. Specifies the position of the caption with respect to the check box.  
State	Integer. Reserved for future use.
State Condition	Enter a formula that will override the **State** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.

Property	Value
ArrayLength	When a screen object is associated with an array data structure, this property specifies the size of the scrolling region that is used to view array entries. For example, if a field is assigned an Array Length of 5, the form will stack five fields vertically to allow viewing of five array entries. A scroll bar appears beside the fields to allow viewing of the array entries. If Array Length is set to -1, the form dynamically creates as many fields as there are entries in the array. If the Input datatype is scalar, only a single Text Box is displayed. The default is 0, which means one vertical line of information is displayed.
ForeColor	Text color. Choose one of 17 colors available from the drop-down list.
Data Changed Event	Specify the option number (such as button ID) to call if the data contained in the object has changed. This sends an event to the display RAD application.
Mandatory	Setting Mandatory to true indicates that the field is required, and flags the top left of the field with a small red indicator. **This is a visible change only:** In order to make the field mandatory for any form it appears on, use Data Policy. To make the field mandatory for a small number of forms only, use Format Control.

Radio Button

Description:A Radio Button allows the user to select one value from a set of mutually-exclusive values. The items in the set are associated with a database field or variable.

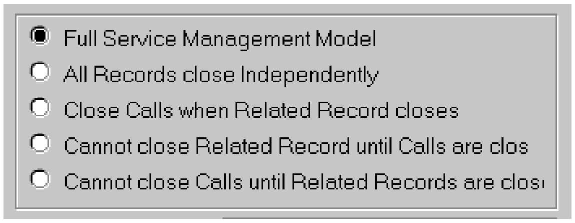
Use radio buttons to give the user a quick way to choose one option from a set of fixed and mutually exclusive string values.

Stylistically, it is good practice to group related radio buttons within a frame or bevel. Also:

- If the choice is logical (true or false), use a check box instead.
- If choices are extensive or if the user may need to enter a choice manually, use a combo box instead.

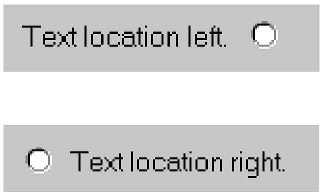
Note: Radio buttons are not supported on QBE lists.

Example:


● Full Service Management Model
○ All Records close Independently
○ Close Calls when Related Record closes
○ Cannot close Related Record until Calls are clos
○ Cannot close Calls until Related Records are clos

Property	Value
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	If you want a caption on the Radio button, enter it in this property.
Caption Condition	Enter a formula that will override the **Caption** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Input	Enter the name of the appropriate database field or variable.
X	Horizontal position based on the left edge of the object.
Y	Vertical position based on the top edge of the object
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.

Property	Value
Visible Condition	Enter a formula that will override the **Visible** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Elastic	Determines the width and height of an object in relation to its parent window. This property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.
TabStop	You will almost always use the default of *0* for TabStop. See *Box Properties* on page 39 for more information on tabstops.
ReadOnly	Prohibits the user from changing the contents of the field.
Read Only Condition	Enter a formula that will override the **Read Only** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Focus Out Event	When this property is set to *true*, the event value is sent to the RAD application when the field loses focus (when you leave the field). Refer to the RAD documentation for further information.

Property	Value
AccessibleName	Allows names to be assigned to the component for use with accessibility software. This function is only supported by the Java client. This property is not required. If no name is present, the Java client defaults to looking for a nearby label to use for the name of the component. For buttons, check boxes or radio buttons, it will use the component's text. Accessibility software can query the Java Client to get this information, and use it to present it to the user in a variety of ways. For example, speech simulation software will say the name and the type of component when the component gets focus.
AccessibleDescription	Allows descriptions to be assigned to the component for use with accessibility software. This function is only supported by the Java client. This property is not required. If no description is present, the Java client defaults to looking for a nearby label to use for the name of the component. For buttons, check boxes or radio buttons, the Java client will use the component's text. Accessibility software can query the Java Client to get this information, and use it to present it to the user in a variety of ways. For example, speech simulation software will say the name and the type of component when the component gets focus, based on the settings in the AccessibleName and AccessibleDescription fields.
TextLocation	Left or Right. Specifies the position of the caption with respect to the Radio Button. 
State	Integer. Reserved for future use.
Value	Value to assign to the field or variable specified in the Input property when this Radio Button is selected.
ForeColor	Text color. Choose one of 17 colors available from the drop-down list.

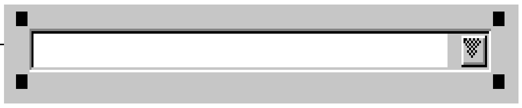
Property	Value
Data Changed Event	Specify the option number (such as button ID) to call if the data contained in the object has changed.
Mandatory	Setting Mandatory to true indicates that the field is required, and flags the top left of the field with a small red indicator. **This is a visible change only**: In order to make the field mandatory for any form it appears on, use Data Policy. To make the field mandatory for a small number of forms only, use Format Control.

Combo Box

Description:A Combo Box allows the user to click on a dedicated button and select from a drop-down list. The items in the list are associated with a database field or variable. A combo box also allows the user to type in a value if the **selectonly** property is set to false(no).

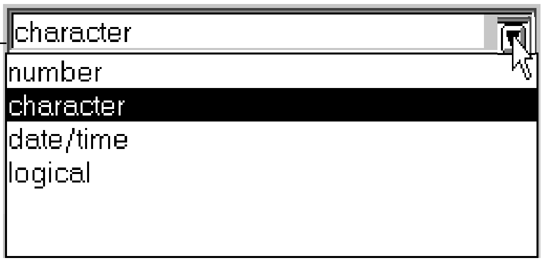
Example:

The combo box as displayed in Forms Designer



The first example shows how the combo box is displayed on the Drawing Canvas when you create it in Forms Designer. Note that the drop-down list is not visible.

The combo box as seen by the user



The second example shows how the user will see the combo box once the dedicated button is clicked.

Property	Value
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	If you want a caption on the combo box, enter it in this property.
Caption Condition	Enter a formula that will override the **Caption** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Input	Enter the name of the appropriate database field or variable.
X	Horizontal position based on the left edge of the object.
Y	Vertical position based on the top edge of the object
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.
Visible Condition	Enter a formula that will override the **Visible** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Elastic	Determines the width and height of an object in relation to its parent window. This property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.

Property	Value
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.
TabStop	You will almost always use the default of *0* for TabStop. See *Box Properties* on page 39 for more information on tabstops.
ReadOnly	Prohibits the user from changing the contents of the field.
Read Only Condition	Enter a formula that will override the **Read Only** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Password	This property does not apply to combo boxes.
MaxChars	Specifies the number of characters the user can enter in the field. The default is 0 (unlimited).
MaxCharsBeep	If set to *Yes*, notifies the user (by sounding a beep) that the maximum number of characters has been reached.
CaseConversion	Changes the case of text entered in the field.
Decimals	Modifies numbers entered in the field to a number of decimal points you specify.
Parse	If Parse is set to *Yes*, the text entered in a field is parsed to verify that it is of the correct syntax. The syntax for the field is defined by the type of field it is displaying (date, expression, number, and so forth).
Focus Out Event	When this property is set to *true*, the event value is sent to the RAD application when the field loses focus (when you leave the field). Refer to the RAD documentation for further information.

Property	Value
Mandatory	Setting Mandatory to true indicates that the field is required, and flags the top left of the field with a small red indicator. **This is a visible change only**: In order to make the field mandatory for any form it appears on, use Data Policy. To make the field mandatory for a small number of forms only, use Format Control.
AccessibleName	Allows names to be assigned to the component for use with accessibility software. This function is only supported by the Java client. This property is not required. If no name is present, the Java client defaults to looking for a nearby label to use for the name of the component. For buttons, check boxes or radio buttons, it will use the component's text. Accessibility software can query the Java Client to get this information, and use it to present it to the user in a variety of ways. For example, speech simulation software will say the name and the type of component when the component gets focus.
AccessibleDescription	Allows descriptions to be assigned to the component for use with accessibility software. This function is only supported by the Java client. This property is not required. If no description is present, the Java client defaults to looking for a nearby label to use for the name of the component. For buttons, check boxes or radio buttons, the Java client will use the component's text. Accessibility software can query the Java Client to get this information, and use it to present it to the user in a variety of ways. For example, speech simulation software will say the name and the type of component when the component gets focus, based on the settings in the AccessibleName and AccessibleDescription fields.

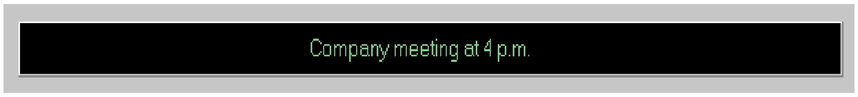
Property	Value
ValueList	This property (in conjunction with the Input property) defines how the DisplayList values are identified in the database. For example, on the previous page we have a drop-down list that displays the following values: ■ number ■ character ■ date/time ■ logical The DisplayList for this drop-down list is a semicolon-delimited list that reads as follows: ■ number;character;date/time;logical For the ValueList, you define how the DisplayList values are identified in the database (i.e., you give them a code that the database uses). For instance, you may enter the following semicolon delimited values in ValueList: ■ *n*;*c*;*d*;*l* (for *number*, *character*, *date/time*, *logical*) If, for instance, the user selects *number* from the drop-down list, the corresponding value in the ValueList (*n*) is assigned to the field specified by the Input property. If the ValueList is the only populated property, the values become the list of display elements for the user. Value and Display Lists are entered using the List Editing Dialog. You can enter hard coded entries for each list, or you can supply a variable as the first and only entry. The run time values of the variable are used to populate these lists. ■ You can also use the `select` function to generate a value list. The `select` function was designed to work against many records and grab a scalar value from each one to make a list. See *Dynamic functions* on page 131.
Value List Condition	Enter a formula that will override the **Value List** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
DisplayList	This property defines the values that are displayed in the drop-down list at run time. You can specify a literal list of semicolon-delimited values, or you can specify the name of an array containing the list of values. There must be a one-to-one correspondence between the values for ValueList and for DisplayList. If the Display list is the only populated property, the display values are what is written to the database.

Property	Value
DisplayList Condition	Enter a formula that will override the **DisplayList** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
BoxLines	Integer. Window size of the drop-down list.
SelectOnly	*Yes* or *No*. *Yes* requires the user to select from the combo box drop-down list. If the user makes a manual entry, the value must already exist in the drop-down list (defined in the ValueList and DisplayList properties). A setting of *No* allows the user to select from the drop-down list or to manually enter a value. *No* will work with or without a drop-down list as defined in the ValueList and DisplayList properties.
ShowUnmatchedValue	Setting this field to **Yes** allows users to see values set in the Value list even if they do not correspond with the Display list values. The **SelectOnly** parameter must be set to **Yes.**
ArrayLength	When a screen object is associated with an array data structure, this property specifies the size of the scrolling region that is used to view array entries. For example, if a field is assigned an Array Length of 5, the form will stack five fields vertically to allow viewing of five array entries. A scroll bar appears beside the fields to allow viewing of the array entries. If Array Length is set to -1, the form dynamically creates as many fields as there are entries in the array. If the Input data type is scalar, only a single Text Box is displayed. The default is 0, which means one vertical line of information is displayed.
Data Changed Event	Specify the option number to call if the data contained in the object has changed. This sends an event to the Display RAD application.

Marquee

Description:A Marquee is an attention-getting text message that scrolls continuously across the screen from right to left.

Example:



Property	Value
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	Text entered here will display on the marquee.
Caption Condition	Enter a formula that will override the **Caption** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Input	Enter the name of the appropriate database field or variable.
X	Horizontal position based on the left edge of the object.
Y	Vertical position based on the top edge of the object
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.
Visible Condition	Enter a formula that will override the **Visible** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.

Property	Value
Elastic	Determines the width and height of an object in relation to its parent window. This property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.
ForeColor	Text color. One of 17 colors available from the drop-down list.
BackColor	Background color. The only Background color available for a Windows client marquee is Black. The Java client supports multiple background colors.
FontIncrease	Allows you to increase or decrease the standard font size.
Font	Select a font for your marquee caption from the drop-down list.
Bold	Applies **bold** to your marquee caption.
Italic	Applies *italics* to your marquee caption.

Subformat

Description:A Subformat is a grouping of information (structure) that is imported into a larger form. Subformats generally contain information that is used in multiple forms—a header, for instance.

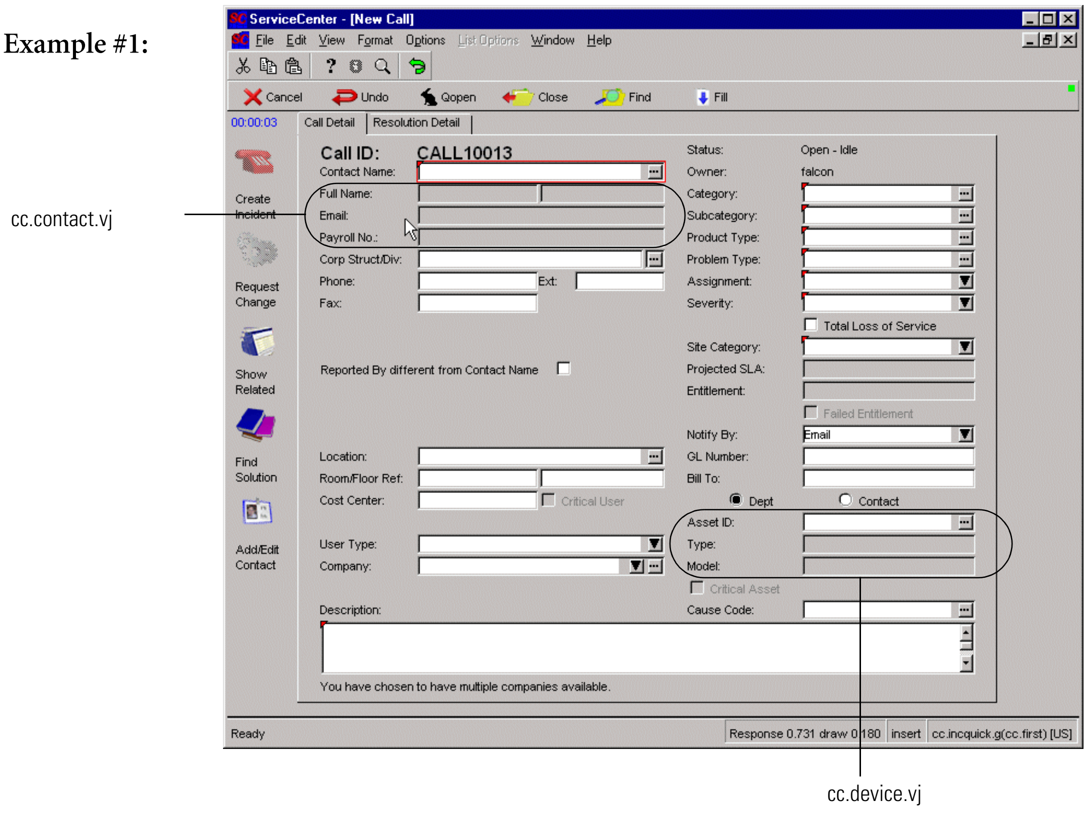
Subformats are constructed in the same way as standard formats, with the exception that they are created to appear within another, larger, format. Subformats can be micro versions of larger forms, containing specific key information that is valuable for display on other formats (e.g., inventory data subformats which appear on incident ticket formats).

The advantage of subformats is that they allow you to create a structure once and then use the structure over again by importing it into other forms. In this way, subformats make it easier for you to create and maintain forms.

Example #1 shows the form *cc.incquick.g* as it is displayed to a user at run time. This form was created using two subformats—one for contact information and the other for device information.

Example #2 shows how the form is displayed to you in Forms Designer.

Example #1:



00:00:03

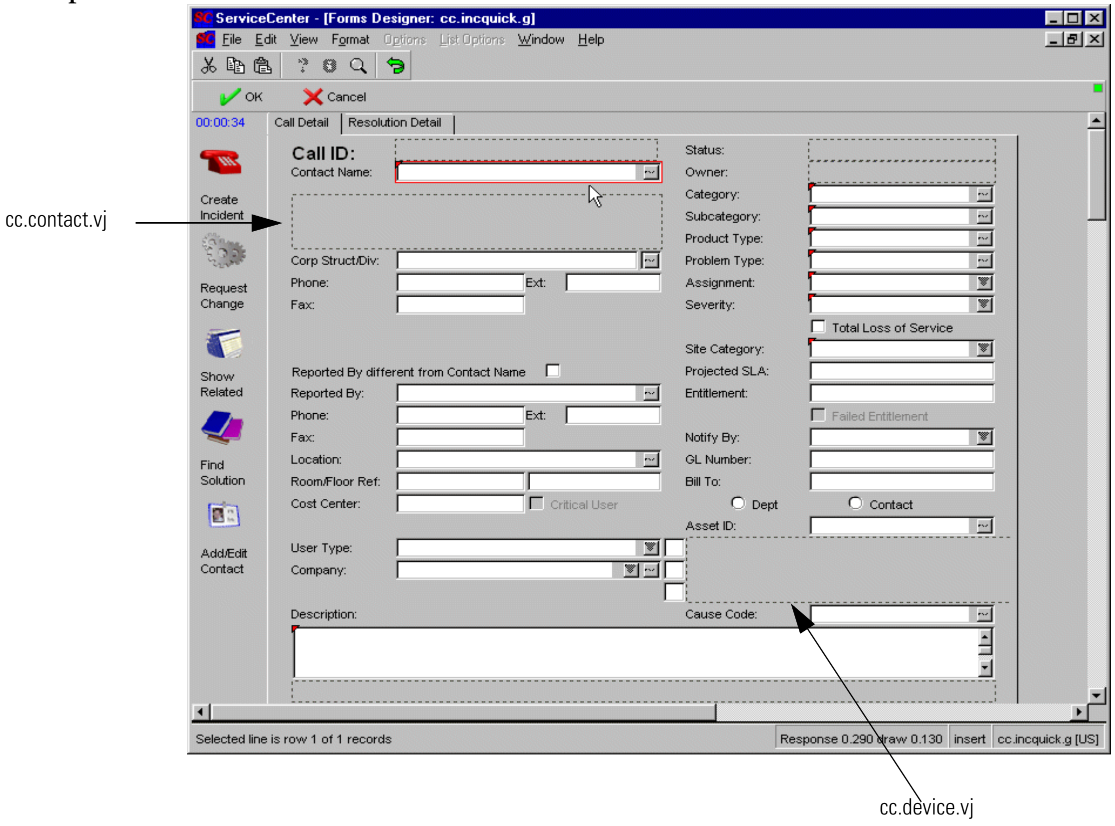
Call ID: CALL10013
Contact Name:
Full Name:
Email:
Payroll No.:
Corp Struct/Div:
Phone: Ext:
Fax:
Reported By different from Contact Name
Location:
Room/Floor Ref:
Cost Center: Critical User
User Type:
Company:
Description:

Status: Open - Idle
Owner: falcon
Category:
Subcategory:
Product Type:
Problem Type:
Assignment:
Severity:
Total Loss of Service
Site Category:
Projected SLA:
Entitlement:
Failed Entitlement
Notify By: Email
GL Number:
Bill To:
[✓ Dept] Contact
Asset ID:
Type:
Model:
Critical Asset
Cause Code:

You have chosen to have multiple companies available.

Ready | Response 0.731 draw 0.180 | insert | cc.incquick.g(cc.first) [US]

Example #2



Selected line is row 1 of 1 records | Response 0.290 draw 0.130 | insert | cc.incquick.g [US]

In design mode, subformat objects are displayed as dashed boxes, as are Labels. The two dashed boxes shown on the drawing canvas correlate with the subformats shown in Example #1. Note that the width of each box is not important—a subformat will assume the size you specify in its individual Properties Window.

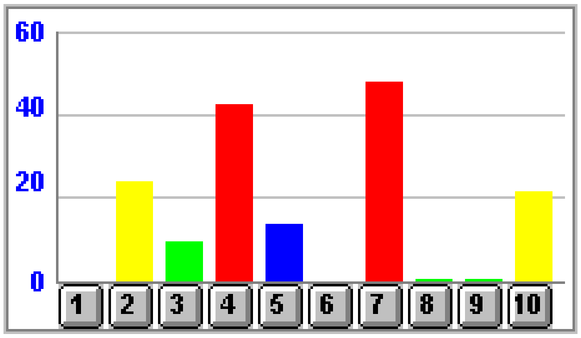
Property	Values
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	This property does not apply to subformats
Caption Condition	Enter a formula that will override the **Caption** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Input	Enter the name of the appropriate database field or variable. You have two choices for databases: ■ to display information from a dbdict, enter the structure name defined in the dbdict. ■ to display linked information, enter the field name defined in the link file for the database you are working with.
X	Horizontal position based on the left edge of the object.
Y	Vertical position based on the top edge of the object
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.
Visible Condition	Enter a formula that will override the **Visible** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Elastic	Determines the width and height of an object in relation to its parent window. This property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.

Property	Values
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.
Format	Format used to display the secondary form.
VirtualJoin	*Yes* or *No*. *Yes* associates virtual join run time processing with the subformat object. *No* indicates you are using a same-file join.
DisplayBlank	*Yes* or *No*. Only for VirtualJoins. *Yes* displays the subformat even if no records were found for the virtual join. *No* skips display of the format if there are no matching records.
DisplayUsing Table	Indicates that the virtual join data will be displayed in a table on the subformat
ArrayLength	When a screen object is associated with an array data structure, this property specifies the size of the scrolling region that is used to view array entries. For example, if a field is assigned an Array Length of 5, the form will stack five fields vertically to allow viewing of five array entries. A scroll bar appears beside the fields to allow viewing of the array entries. If Array Length is set to -1, the form dynamically creates as many fields as there are entries in the array.

Chart

Description:A Chart displays the contents of a numeric array as a two-dimensional, color-coded bar chart with (optional) definable buttons.

Example:



Property	Value
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	This property does not apply to charts.
Caption Condition	Enter a formula that will override the **Caption** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Input	Enter the name of the appropriate database field or variable.
X	Horizontal position based on the left edge of the object.
Y	Vertical position based on the top edge of the object
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.

Property	Value
Visible Condition	Enter a formula that will override the **Visible** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Elastic	Determines the width and height of an object in relation to its parent window. This property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.
TabStop	Defaults to -1. With the default setting, the chart will not receive focus. If you want a chart to receive focus, set the TabStop property to a number greater than or equal to 0. With this setting, you can tab into the chart.
BarWidth	Integer. Width of bars in alignment grid units.
ScaleMax	Integer. Specifies the maximum vertical bar height. Columns that exceed the ScaleMax setting will appear clipped. A setting of *0* causes the chart to automatically scale vertically to the tallest column.
ButtonBase	*Yes* or *No*. *Yes* enables a row of buttons across the base of the chart, one button per column.

Property	Value
BaseButtonID	Numeric. Specifies the Control ID for the left-most chart button. The Base ButtonID is incremented by one from left to right. Using the default value of 801 as an example, the left-most button would transmit 801, the second button from the left would transmit 802, the third 803, and so forth. The BaseButton ID relates to an option number in the menu record. Assign Button IDs to the chart and create an associated option number in the menu record, so that each button calls a stored query.
ColorList	Specifies a list of semicolon delimited colors. The ColorScale and ColorPercent properties determine how the list is applied to the columns. The available colors are: Black, Red, Green, Blue, Gray, Light Gray, Dark Gray, Yellow, Cyan, Magenta, White, Forest, Navy, Purple, Teal, Brick, and Manila
ColorScale	Specifies a list of semicolon delimited numeric values that determine the color of a column.
ColorPercent	*Yes* or *No*. *Yes* indicates that ColorScale as a list of percentages from 1 to 100.
ForeColor	Text color. Select from seventeen colors available from the drop-down list.
BalloonHelp	A one-line help description that ServiceCenter displays while the user's cursor is over the button. Balloon Help is limited to 80 characters.

More about the Input property

The input property most commonly specifies a numeric array. For example, a global variable such as *$category.first.response* might store the average number of minutes between the first report of trouble and the first response by a technician for each of eight categories:

```
18,29,62,19,37,45,16,30
```

The elements in this array would be charted from left to right. If you needed to create a static chart (for example, for testing), you could specify a literal list of numbers delimited by semicolons in the Input property:

```
18;29;62;19;37;45;16;30
```

More about column colors

There are four methods for assigning column colors:

- **Default (all red)**—to keep all columns red, leave the ColorList and ColorScale properties blank.
- **Alternate colors—**to alternate the column colors, specify a ColorList and leave ColorScale blank. The colors in ColorList are applied to the columns repeatedly from left to right. For example, a ColorList property of *Blue;Yellow* specifies blue odd numbered columns and yellow even numbered columns.
- **Color based on column value—**to assign a color based on the value of a column, specify a ColorList and ColorScale, and set ColorPercent to *No*.

For example, you could have a ColorScale property of:

30;45

and a ColorList property of:

Green;Yellow;Red

This sets up a three-tiered ColorScale that determines which ColorList color to apply to a column.

column value <= 30use 1st ColorList (Green)

30 < column value <= 45use 2nd ColorList (Yellow)

column value > 45use 3rd ColorList (Red)

- **Color based on relative column value—**to assign a color based on the relative value of a column, specify a ColorList and set ColorPercent to *Yes*. For ColorScale, specify percentages from 1 to 100. Relative value for a column is computed as follows:

100 * (column value)/ScaleMax

Table

Description:A Table allows you to display one or more columns of data in a scrollable pane. For more information about other features that are supported by this tool, refer to *More about tables* on page 101.

Example:

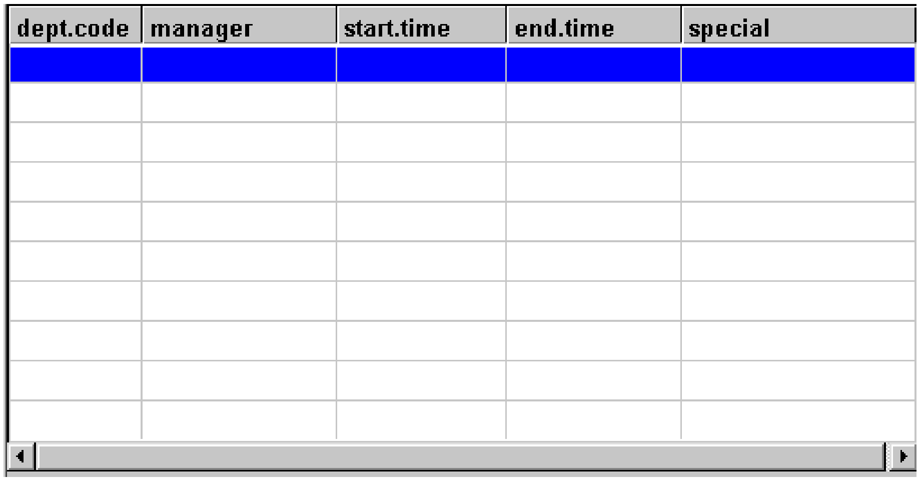


Table Properties (fields editable by clicking inside the table):

Property	Value
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	This property does not apply to tables. Use the Columns property to apply headings to columns within the table, or click on a column and enter the text in the **Column's** Caption field.
Caption Condition	Enter a formula that will override the **Caption** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Input	Enter the name of the appropriate database field or variable.
X	Horizontal position based on the left edge of the object.
Y	Vertical position based on the top edge of the object
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.

Property	Value
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.
Visible Condition	Enter a formula that will override the **Visible** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Elastic	Determines the width and height of an object in relation to its parent window. This property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.
TabStop	You will almost always use the default of *0* for TabStop. See *Box Properties* on page 39 for more information on tabstops.
MultipleSelections	This option is not currently implemented.
Columns	List of columns in the table.
ColumnHeadings	If *Yes*, headings are displayed above the columns.
ReadOnly	If *Yes*, the cells in the table cannot be edited.
AllowRowDelete	If *Yes*, the user can delete rows by pressing the **Delete** key. If *Yes*/Prompted, the user is first prompted before deletion is done.

Property	Value
BackColor	The color of the table background.
BackColor Condition	Enter a formula that will override the **BackColor** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
ForeColor	Color used for the table contents (text).
ForeColor Condition	Enter a formula that will override the **ForeColor** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
SelectedBackColor	Background color of a selected row.
SelectedForeColor	Foreground color of selected text.
HeadingBackColor	Background color of the top row used to display column headings.
SeparatorColor	Color used to draw the column and row separators.
InvertReadOnly Colors	If *Yes*, the table is drawn in inverted colors (background and separator colors are swapped, normally resulting in a grayed-out table).
OldStyleFdisp	If *Yes*, the tables used in FDISP panels will revert row selections to ROW 0 as soon as the user makes a function key selection in the tool.
DefaultButtonID	If an individual column ButtonID is set to 0, the DefaultButtonID is used. See the definition for the ButtonID property in the Column Properties definitions.
DoubleClick ButtonID	The ID generated when a selected row is double clicked. See the definition for the DoubleClick ButtonID property in the Column Properties definitions.
Refresh Rate	Specify the refresh rate (how often the client queries the server for an update) for the table, in seconds. This setting can effect system performance if too fast a rate is used.
DoubleClickField	The ID generated when a selected field is double clicked.
Font	Select a font for your table from the drop-down list.
FontIncrease	Allows you to increase or decrease the standard font size.
Bold	When set to *Yes*, applies **bold** to the default font.

Property	Value
Bold Condition	Enter a formula that will override the **Bold** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Italic	When set to *Yes*, applies *italics* to the default font.
Italic Condition	Enter a formula that will override the **Italic** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.

Column Properties (editable by clicking on the column heading after you have created it):

Note: Column Properties cannot be accessed if 'ColumnHeadings' property is set to **No**.

Property	Value
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	Enter the text for your column heading in this property.
Caption Condition	Enter a formula that will override the **Caption** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Input	Enter the array variable used as Input for the column.
X	Horizontal position based on the object's left edge.
Y	Vertical position based on the object's top edge.
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.

Property	Value
Visible Condition	Enter a formula that will override the **Visible** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Elastic	Determines the width and height of an object in relation to its parent window. This property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.
Field	If the array variable used as Input is an array of structures, you may enter the field name within the structure that should be displayed in the column.
Column Width%	Size of the column as a percentage of the table's visible width.
MinWidth	If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Column Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the column.
Sizable	If *Yes*, the user can resize the column using the mouse.
Clickable Title	If *Yes*, the column's heading acts as a clickable button.

Property	Value
ButtonID	The ID generated when a clickable title button is pressed. Behaves just like other objects with a ButtonID property.
ShowTitle	If *Yes*, the heading for the column is visible.
ValueList	List of display choices for editing cells in the column. If this list is empty, cells in the column are edited using an edit control. If the list is not empty, cells in the column are edited using a combo box with the choices in the ValueList available in a drop-down menu.
Bitmaps	List of display bitmaps. This option works in conjuction with ValueList. If the input to a cell matches a value in the ValueList, the Bitmap with the corresponding index is displayed instead.
DisplayList	This property defines the values that are displayed in the drop-down list at run time. You can specify a literal list of semicolon-delimited values, or you can specify the name of an array containing the list of values. There must be a one-to-one correspondence between the values for ValueList and for DisplayList.
SelectOnly	This property is relevant only if a ValueList is provided. If *Yes*, the column allows selections from a combo box populated with values from the ValueList but will not allow manual editing of the cell text.
ReadOnly	If *Yes*, cells in the column cannot be edited.
MaxChars	Specifies the number of characters the user can enter in the field. The default is 0 (unlimited).
MaxCharsBeep	If set to *Yes*, the system notifies the user (by sounding a beep) that the maximum number of characters has been reached.
CaseConversion	Changes the *case* of text entered in the column cells.
Decimals	Modifies numbers entered in the column cells to a number of decimal points you specify.
Parse	If Parse is set to *Yes*, the text entered in a field is parsed to verify that it is of the correct syntax. The syntax for the field is defined by the type of field it is displaying (date, expression, number, and so forth).
Justification	Allows you to align column contents to the Left, Center, or Right.

Property	Value
InputConv	This attribute allows you to specify the name of an input conversion routine. If specified, the routine is called at runtime to convert the contents of the field before it is displayed on the screen. For instance, you could have a function that converts dollars to French francs.
OutputConv	Same as InputConv, but applies to output values.
Password	Setting this to **Yes** makes the text entered in the field a series of asterisks on the screen. The data being displayed is not encrypted, just transposed.

More about tables

The tables you create in Forms Designer appear as a rectangular region subdivided by rows and columns. The look and feel mimics tables in Microsoft Windows applications like Excel.

Tables support many features, including the following:

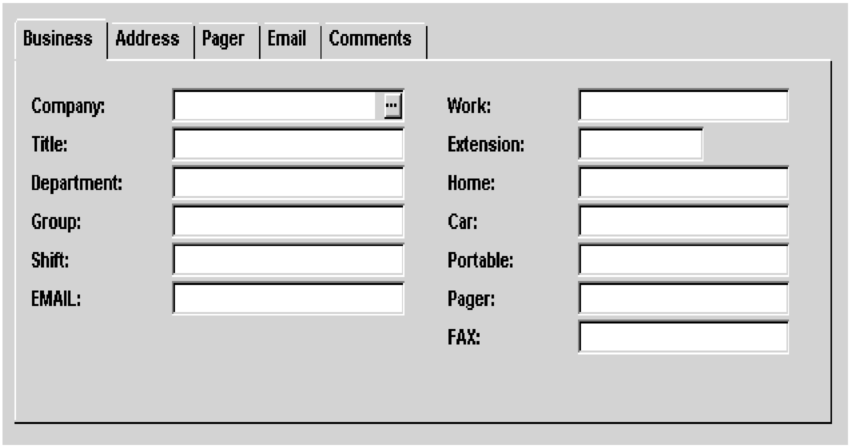
- Cells may display text or bitmaps.
- Columns may vary in width. The user can resize the columns using the mouse.
- Columns may have an optional heading button, which can be mapped to perform an action.
- All rows are the same height, defined by the table's font.
- The user can scroll through data by use of scrollbars or the keyboard.
- All expected keyboard navigation is supported (Up, Down, Next, Prev, Last, First, and so forth).
- Cells can be editable or read-only.
- Selecting a read-only cell results in a highlight of the entire row.
- Selecting an editable cell results in the in-place creation of an edit field or drop-down combo box that lets the user edit the cell.
- Users can edit a field by clicking on it or by pressing the **Enter** key while the table has focus. Tabbing events received while the table is in edit mode result in navigation from one cell to the next. Pressing **Enter** while a cell is being edited results in the changes being accepted and the cell's row being highlighted. Tabbing events received when a row is highlighted are handled by passing focus to the next object in the window.

Notebooks

Description: *Notebooks* (or Tabbed Notebooks) allow you to subdivide the contents of a screen into logical groups or categories. They provide an aesthetic way of organizing large amounts of data into small spaces.

This tool supports many features which are described in detail in *More about notebooks* on page 104.

Example:



Property	Value
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	Enter the text for your notebook heading in this property.
Caption Condition	Enter a formula that will override the **Caption** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Input	Enter the array variable used as Input for the column.

Property	Value
X	Horizontal position based on the object's left edge.
Y	Vertical position based on the object's top edge.
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.
Visible Condition	Enter a formula that will override the **Visible** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Elastic	Determines the width and height of an object in relation to its parent window. This property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.

Property	Value
TabStop	Allows you to override the default tabbing order within a form. By default, the TAB key navigates the focus on a form from top to bottom and left to right. However, if some objects have a tabstop of non-zero, they are visited first, in ascending order.
Pages	A semicolon-delimited list of the tab headings.

More about notebooks

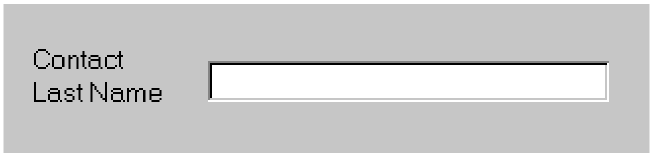
A tabbed notebook can have one or more pages, and defaults to two pages when first placed. Each page provides its own drawing canvas where you can insert other controls.

Navigate to each page by selecting its tab with the mouse or from the keyboard by using the right and left arrow keys. You can assign tabstops to the objects inside each page, but remember that the object with tabstop 1 is given focus first. This means that inserting an object with tabstop 1 inside a page will cause that page to be the first opened page when a screen is constructed. In Forms Designer, you create objects inside each page simply by flipping to the correct page and drawing inside. You can paste objects into a page by first selecting the notebook and clicking inside the target page before performing the paste command.

Wrapping Label

Description: The Wrapping Label tool allows you to place multiline text on a label. You can give titles to forms, give labels to objects within the form, or otherwise place text on the form.

Example:



The wrapping label tool can also be used to place a data field in a form that displays a QBE list.

Property	Value
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	Enter the text for your wrapping label in this property.
Caption Condition	Enter a formula that will override the **Caption** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Input	Enter the array variable used as Input for the column.
X	Horizontal position based on the object's left edge.
Y	Vertical position based on the object's top edge.
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.
Visible Condition	Enter a formula that will override the **Visible** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Elastic	Determines the width and height of an object in relation to its parent window. This property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.

Property	Value
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.
Font	Select a font for your label from the drop-down list.
ForeColor	Text color. Select from 17 available colors.
ForeColor Condition	Enter a formula for evaluating the ForeColor property at run time. The result of this evaluation is used to override the value assigned to the property.
BackColor	Background color. The Opaque property must be set to *Yes* in order for the BackColor to have effect.
BackColor Condition	Enter a formula that will override the **BackColor** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Bold	When set to *Yes*, applies **bold** to your label.
Bold Condition	Enter a formula that will override the **Bold** property, if the conditions of the formula are met.
Italic	When set to *Yes*, applies *italics* to your label.
Italic Condition	Enter a formula that will override the **Italic** property, if the conditions of the formula are met.
Opaque	Enables the BackColor property.
FontIncrease	Allows you to increase or decrease the standard font size.
FontIncrease Condition	Enter a formula that will override the **FontIncrease** property, if the conditions of the formula are met.

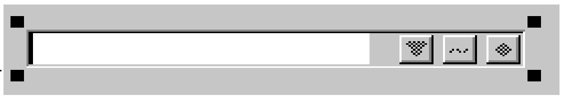
Property	Value
OutputConv	This property does not apply to labels.
ArrayLength	When a screen object is associated with an array data structure, this property specifies the size of the scrolling region that is used to view array entries. For example, if a field is assigned an ArrayLength of 5, the form will stack five fields vertically to allow viewing of five array entries. A scroll bar appears beside the fields to allow viewing of the array entries. If ArrayLength is set to -1, the form dynamically creates as many fields as there are entries in the array. If the Input data type is scalar, only a single Text Box is displayed. The default is 0, which means one vertical line of information is displayed.

ComFill

Description: ComFill is a combination combo box and fill button. ComFill has all the properties of the individual combo box, plus the capability for fill button.

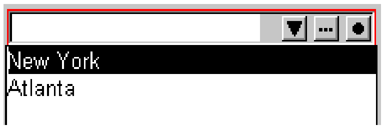
Example:

the comfill box as displayed in Forms Designer



The first example shows how the comfill box is displayed on the Drawing Canvas when you create it in Forms Designer. Note that the drop-down list is not visible.

the comfill box as seen by the user



The second example shows how the user will see the comfill box once the dedicated button is clicked.

Property	Value
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	Enter a caption for the ComFill object in this property.
Caption Condition	Enter a formula that will override the **Caption** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.

Property	Value
Input	Enter the name of the appropriate database field or variable.
X	Horizontal position based on the left edge of the object.
Y	Vertical position based on the top edge of the object
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.
Visible Condition	Enter a formula that will override the **Visible** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Elastic	Determines the width and height of an object in relation to its parent window. This property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.
TabStop	You will almost always use the default of *0* for TabStop. *Box Properties* on page 39 for more information on tabstops.
ReadOnly	Prohibits the user from changing the contents of the field.

Property	Value
ReadOnly Condition	Enter a formula that will override the **ReadOnly** property, if the conditions of the formula are met.
Password	This property does not apply to ComFill boxes.
MaxChars	Specifies the number of characters the user can enter in the field. The default is 0 (unlimited).
MaxCharsBeep	If set to *Yes*, notifies the user (by sounding a beep) that the maximum number of characters has been reached.
CaseConversion	Changes the case of text entered in the field.
Decimals	Modifies numbers entered in the field to a number of decimal points you specify.
Parse	If Parse is set to *Yes*, the text entered in a field is parsed to verify that it is of the correct syntax. The syntax for the field is defined by the type of field it is displaying (date, expression, number, and so forth).
Focus Out Event	When this property is set to *true*, the event value is sent to the RAD application when the field loses focus (when you leave the field). Refer to the RAD documentation for further information.
Mandatory	Setting Mandatory to true indicates that the field is required, and flags the top left of the field with a small red indicator. **This is a visible change only**: In order to make the field mandatory for any form it appears on, use Data Policy. To make the field mandatory for a small number of forms only, use Format Control.
AccessibleName	Allows names to be assigned to the component for use with accessibility software. This function is only supported by the Java client. This property is not required. If no name is present, the Java client defaults to looking for a nearby label to use for the name of the component. For buttons, check boxes or radio buttons, it will use the component's text. Accessibility software can query the Java Client to get this information, and use it to present it to the user in a variety of ways. For example, speech simulation software will say the name and the type of component when the component gets focus.

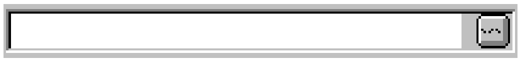
Property	Value
AccessibleDescription	Allows descriptions to be assigned to the component for use with accessibility software. This function is only supported by the Java client. This property is not required. If no description is present, the Java client defaults to looking for a nearby label to use for the name of the component. For buttons, check boxes or radio buttons, the Java client will use the component's text. Accessibility software can query the Java Client to get this information, and use it to present it to the user in a variety of ways. For example, speech simulation software will say the name and the type of component when the component gets focus, based on the settings in the AccessibleName and AccessibleDescription fields.
ValueList	This property (in conjunction with the Input property) defines how the DisplayList values are identified in the database. For example, suppose we have a drop-down list that displays the following values: ■ number ■ character ■ date/time ■ logical The DisplayList for this drop-down list is a semicolon-delimited list that reads as follows: ■ number;character;date/time;logical For the ValueList, you define how the DisplayList values are identified in the database (i.e., you give them a code that the database uses). For instance, you may enter the following semicolon delimited values in ValueList: ■ *n*;*c*;*d*;*l* (for *number*, *character*, *date/time*, *logical*) If, for instance, the user selects *number* from the drop-down list, the corresponding value in the ValueList (*n*) is assigned to the field specified by the Input property. If the ValueList is the only populated property, the values become the list of display elements for the user. Value and Display Lists are entered using the List Editing Dialog. You can enter hard coded entries for each list, or you can supply a variable as the first and only entry. The run time values of the variable are used to populate these lists. ■ You can also use the `select` function to generate a value list. The `select` function was designed to work against many records and grab a scalar value from each one to make a list. See *Dynamic functions* on page 131.

Property	Value
Value List Condition	Enter a formula that will override the **Value List** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
DisplayList	This property defines the values that are displayed in the drop-down list at run time. You can specify a literal list of semicolon-delimited values, or you can specify the name of an array containing the list of values. There must be a one-to-one correspondence between the values for ValueList and for DisplayList. If the Display list is the only populated property, the display values are what is written to the database.
DisplayList Condition	Enter a formula that will override the **DisplayList** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
BoxLines	Integer. Window size of the drop-down list.
SelectOnly	*Yes* or *No*. *Yes* requires the user to select from the combo box drop-down list. If the user makes a manual entry, the value must already exist in the drop-down list (defined in the ValueList and DisplayList properties). A setting of *No* allows the user to select from the drop-down list or to manually enter a value. *No* will work with or without a drop-down list as defined in the ValueList and DisplayList properties.
ArrayLength	When a screen object is associated with an array data structure, this property specifies the size of the scrolling region that is used to view array entries. For example, if a field is assigned an Array Length of 5, the form will stack five fields vertically to allow viewing of five array entries. A scroll bar appears beside the fields to allow viewing of the array entries. If Array Length is set to -1, the form dynamically creates as many fields as there are entries in the array. If the Input data type is scalar, only a single Text Box is displayed. The default is 0, which means one vertical line of information is displayed.
Data Changed Event	Specify the option number to call if the data contained in the object has changed. This sends an event to the Display RAD application.

Property	Value
ComboButton Visible	If ComboButton Visible is set to *No*, the combobox object of the ComFill button is hidden from view when the form is displayed.
ComboButton Visible Condition	Enter a formula that will override the **ComboButton Visible** property, if the conditions of the formula are met.
InputConv	This attribute allows you to specify the name of an input conversion routine. If specified, the routine is called at runtime to convert the contents of the field before it is displayed on the screen. For instance, you could have a function that converts dollars to French francs.
OutputConv	Same as InputConv, but applies to output values.
FillButtonID	Numeric. Specifies a Control ID to transmit when clicked. For example, 3 corresponds to **F3**. 0 corresponds to **Enter**. Refer to the *ServiceCenter RAD Guide* for more details.
FillButtonBitmap	Choose an icon to display on the field, if desired. You can select from 1 to 17. The default value is 7
FillButtonVisible	If FillButton Visible is set to *No*, the Fill button object of the ComFill button is hidden from view when the form is displayed.
FillButtonVisible Condition	Enter a formula that will override the **FillButton Visible** property, if the conditions of the formula are met.
ThirdButtonID	Numeric. Specifies a Control ID to transmit when clicked. For example, 3 corresponds to **F3**. 0 corresponds to **Enter**. Refer to the *ServiceCenter RAD Guide* for more details.
ThirdButtonBitmap	Choose an icon to display on the field, if desired. You can select from 1 to 17. The default value is 10.
ThirdButton Visible	If ThirdButton Visible is set to *No*, the Third Button object is hidden from view when the form is displayed.
ThirdButton Visible Condition	Enter a formula that will override the **ThirdButton Visible** property, if the conditions of the formula are met.

Gfile

Description: A Gfile Box launches an **Open File** dialog box.



Property	Value
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	This property does not apply to the GFile object.
Caption Condition	Enter a formula that will override the **Caption** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Input	Enter the name of the appropriate database field or variable.
X	Horizontal position based on the left edge of the object.
Y	Vertical position based on the top edge of the object.
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.
Visible Condition	Enter a formula that will override the **Visible** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.

Property	Value
Elastic	Determines the width and height of an object in relation to its parent window. This property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.
TabStop	You will almost always use the default of *0* for TabStop. See *Box Properties* on page 39 for more information on tabstops.
ReadOnly	Prohibits the user from changing the contents of the field.
ReadOnly Condition	Enter a formula that will override the **ReadOnly** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Password	This property does not apply to Gfill boxes.
MaxChars	Specifies the number of characters the user can enter in the field. The default is 0 (unlimited).
MaxCharsBeep	If set to *Yes*, the system notifies the user (by sounding a beep) that the maximum number of characters has been reached.
CaseConversion	Changes the *case* of text entered in the field.
Decimals	Modifies numbers entered in the field to a number of decimal points you specify.

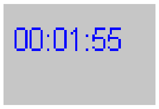
Property	Value
Parse	If Parse is set to *Yes*, the text entered in a field is parsed to verify that it is of the correct syntax. The syntax for the field is defined by the type of field it is displaying (date, expression, number, and so forth).
Focus Out Event	When this property is set to *true*, the event value is sent to the RAD application when the field loses focus (when you leave the field). Refer to the RAD documentation for further information.
Mandatory	Setting Mandatory to true indicates that the field is required, and flags the top left of the field with a small red indicator. **This is a visible change only**: In order to make the field mandatory for any form it appears on, use Data Policy. To make the field mandatory for a small number of forms only, use Format Control.
AccessibleName	Allows names to be assigned to the component for use with accessibility software. This function is only supported by the Java client. This property is not required. If no name is present, the Java client defaults to looking for a nearby label to use for the name of the component. For buttons, check boxes or radio buttons, it will use the component's text. Accessibility software can query the Java Client to get this information, and use it to present it to the user in a variety of ways. For example, speech simulation software will say the name and the type of component when the component gets focus.
AccessibleDescription	Allows descriptions to be assigned to the component for use with accessibility software. This function is only supported by the Java client. This property is not required. If no description is present, the Java client defaults to looking for a nearby label to use for the name of the component. For buttons, check boxes or radio buttons, the Java client will use the component's text. Accessibility software can query the Java Client to get this information, and use it to present it to the user in a variety of ways. For example, speech simulation software will say the name and the type of component when the component gets focus, based on the settings in the AccessibleName and AccessibleDescription fields.
File Directory	The absolute path to a directory. For example, c:\temp. If no directory is specified, the ServiceCenter RUN directory defaults.
File Type	The MIME- type of the file the File Open dialog will prompt for. For example,.txt for text files. If no MIME type is specified, the default is All Files (*.*)

Property	Value
Title	Enter a title for the File Open dialog box, if desired. The default value is “Open.”
ArrayLength	When a screen object is associated with an array data structure, this property specifies the size of the scrolling region that is used to view array entries. For example, if a field is assigned an Array Length of 5, the form will stack five fields vertically to allow viewing of five array entries. A scroll bar appears beside the fields to allow viewing of the array entries. If Array Length is set to -1, the form dynamically creates as many fields as there are entries in the array. If the Input datatype is scalar, only a single Text Box is displayed. The default is 0, which means one vertical line of information is displayed.

Timer

Description: The Timer tool places a clock timer on a form. The timer object can initiate an event at a certain time, based on settings in the Input and Expiration Event fields.

Example:



Property	Value
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	This property is not supported for the Timer object.
Caption Condition	This property is not supported for the Timer object.
Input	The input field can be left blank or can take a variable. If the Input field is left blank, the timer functions as a digital timer, starting at zero and progressing in one second intervals. You can create and assign a variable to your form using Format Control. Be sure to assign a thread variable, as opposed to a local variable. If the variable is a positive number, then the timer will count up from that number in one second intervals. If the variable is a negative number, then the timer will count down to zero, at which point the event specified in the Expiration event field will activate. The timer will then begin to count up in one second intervals. In addition, you can use date/time to count the seconds to the input value. For example, set the variable to tod() + '1 00:00' to count down for one hour.

Property	Value
X	Horizontal position based on the left edge of the object.
Y	Vertical position based on the top edge of the object.
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.
Visible Condition	Enter a formula that will override the **Visible** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Elastic	Determines the width and height of an object in relation to its parent window. This property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.
Font	Select a font for your table from the drop-down list.
ForeColor	Color used for the table contents (text).

Property	Value
ForeColor Condition	Enter a formula that will override the **ForeColor** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
BackColor	The color of the table background. The Opaque property must be set to Yes.
BackColor Condition	Enter a formula that will override the **BackColor** property, if the conditions of the formula are met. See *Specifying property conditions* on page 129 for more information on constructing formulas.
Bold	When set to *Yes*, applies **bold** to the default font.
Bold Condition	Enter a formula that will override the **Bold** property, if the conditions of the formula are met
Italic	When set to *Yes*, applies *italics* to the default font.
Italic Condition	Enter a formula that will override the **Italic** property, if the conditions of the formula are met
Justification	Select from Left, Right or Center.
Opaque	Enables the BackColor property.
FontIncrease	Allows you to increase or decrease the standard font size.
FontIncrease Condition	Enter a formula that will override the **FontIncrease** property, if the conditions of the formula are met
OutputConv	This attribute allows you to specify the name of an Output conversion routine. If specified, the routine is called at runtime to convert the contents of the field before it is displayed on the screen.
ArrayLength	When a screen object is associated with an array data structure, this property specifies the size of the scrolling region that is used to view array entries. For example, if a field is assigned an Array Length of 5, the form will stack five fields vertically to allow viewing of five array entries. A scroll bar appears beside the fields to allow viewing of the array entries. If Array Length is set to -1, the form dynamically creates as many fields as there are entries in the array. If the Input data type is scalar, only a single Text Box is displayed. The default is 0, which means one vertical line of information is displayed.

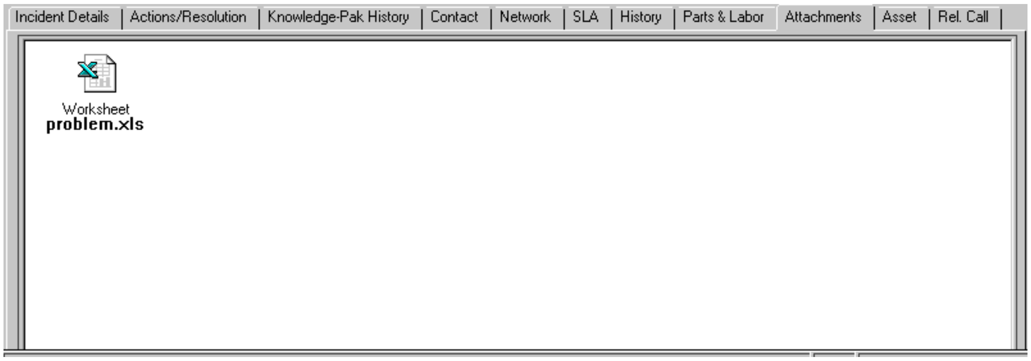
Property	Value
Expiration Color	The timer will change to the color specified in this field, when the timer reaches zero.
Expiration Event	The event sent to the server when the timer reaches zero. Enter a numeric value, for example, a button or event ID.

Attachment container

Description: The Attachment container tool allows the user to attach a non-ServiceCenter document (for example, from Microsoft Word or Microsoft Excel) to a ServiceCenter document such as an incident ticket or SLA agreement. In the example shown below, an Excel spreadsheet has been added to the Attachments tab of an incident ticket. Attachments are measured in bytes.

Note: In order to save attachments with Incident records using Qopen, the **Delay Assigning Problem Number** field in the **Incident Management Environment** profile must be clear. If **Delay Assigning Problem Number** is set to True, then attachments will not be saved with the incident record when performing a Qopen. Only the Qopen operation is affected by this setting. See the Application Administration guide for more information.

Example:



Property	Value
Name	Enter a unique identifier, if desired, for the object on the screen. This name is used by external applications, such as RAD, to change the properties of the object dynamically. This property is optional.
Caption	Enter the text for your Attachment tab in this field.

Property	Value
Caption Condition	Enter a formula that will override the **Caption** property, if the conditions of the formula are met.
Input	Enter the name of the appropriate database field or variable. This is not where the object is stored, but rather what field is used to store the *relationship* between the objects. Often this the unique key of the file the format is associated with For example, in the contacts file (*contacts.g*), the input value can be *contact.name*. This unique key allows each contact to have their own picture. For devices, you can use *devtype* to allow all PCs to have the same picture.
X	Horizontal position based on the left edge of the object.
Y	Vertical position based on the top edge of the object.
Height	Height of the object in alignment grid units.
Width	Width of the object in alignment grid units.
Visible	If Visible is set to *false*, the object is hidden from view when the form is displayed.
Visible Condition	Enter a formula that will override the **Visible** property, if the conditions of the formula are met.
Elastic	Determines the width and height of an object in relation to its parent window. his property can have one of four values: ■ None—the default. If the parent window is resized, the object will not resize with it. ■ Horizontal—automatically adjusts the width of the object in proportion to the width of the parent window. ■ Vertical—automatically adjusts the height of the object in proportion to the height of the parent window. ■ Both—automatically adjusts both the width and the height in proportion to the parent window.
Min Width	Used only when the Elastic property is set to Horizontal or Both. If Min Width is set to the default value (-1), then the smallest width of the object will be equal to the value set for the Width property. If Min Width is set to any other value (for example, 20), the object will use this value for the minimum width of the object instead of the value set for the Width property.

Property	Value
Min Height	Used only when the Elastic property is set to Vertical or Both. If Min Height is set to the default value (-1), then the smallest height of the object will be equal to the value set for the Height property. If Min Height is set to any other value (for example, 20), the object will use this value for the minimum height of the object instead of the value set for the Height property.
TabStop	Allows you to override the default tabbing order within a form. By default, the TAB key navigates the focus on a form from top to bottom and left to right. However, if some objects have a tabstop of non-zero, they are visited first, in ascending order.
ReadOnly	Disables editing capabilities, providing only viewing access.
ReadOnly Condition	Enter a formula that will override the **ReadOnly** property, if the conditions of the formula are met.
Focus Out Event	When this property is set to *true*, the event value is sent to the RAD application when the field loses focus (when you leave the field). Refer to the RAD documentation for further information.
AccessibleName	Allows names to be assigned to the component for use with accessibility software. This function is only supported by the Java client. This property is not required. If no name is present, the Java client defaults to looking for a nearby label to use for the name of the component. For buttons, check boxes or radio buttons, it will use the component's text. Accessibility software can query the Java Client to get this information, and use it to present it to the user in a variety of ways. For example, speech simulation software will say the name and the type of component when the component gets focus.
AccessibleDescription	Allows descriptions to be assigned to the component for use with accessibility software. This function is only supported by the Java client. This property is not required. If no description is present, the Java client defaults to looking for a nearby label to use for the name of the component. For buttons, check boxes or radio buttons, the Java client will use the component's text. Accessibility software can query the Java Client to get this information, and use it to present it to the user in a variety of ways. For example, speech simulation software will say the name and the type of component when the component gets focus, based on the settings in the AccessibleName and AccessibleDescription fields.

Property	Value
MaxAttachSize	A straight number field that sets the absolute size, measured in bytes, of an attachment for the form being modified.
MaxAttachSize Condition	A condition statement that varies the maximum allowed size of an attachment based on the capabilities of a user. For example, a system administrator could have a higher limit than a manager, who in turn could have a higher limit than a help desk user.

More about Attachment containers

Attachment containers allow you to drag and drop or copy and paste files into ServiceCenter. When you open a record that contains an attachment, you can double-click the attachment and open that file in the appropriate application.

Following are examples of how Attachment containers can be used:

- To attach a document describing the original agreement between parties involved in a Service Level Agreement to the SLA form.
- To attach a spreadsheet which is causing problems directly to the problem ticket opened to deal with the spreadsheet.

The Container is a separate input field that can contain only attachments. Attachments are stored in the ServiceCenter repository and are accessible to all ServiceCenter users.

5 CHAPTER

Dynamic View Dependencies

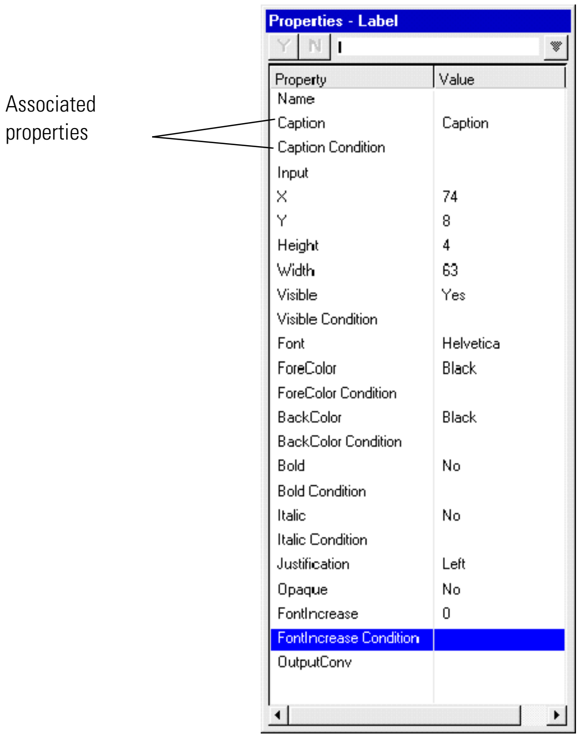
Dynamic View Dependencies involves a dynamic evaluation of view properties according to special dependencies set up at forms design time.

Using Forms Designer, you can create objects with attributes that are evaluated when the form is displayed and as the user interacts with the form. For instance, you can do things like:

- Specify that a field should be visible only when another field contains a certain value
- Specify that the choices in a drop-down list be dynamically calculated based on the choice in another combo box
- Specify that a field be drawn in a special font or color when it displays a certain value

ServiceCenter ships with a form called `dep.g` to demonstrate Dynamic View Dependencies. View this form using Forms Designer. Try interacting with the different fields before entering Design Mode to observe how View properties change dynamically, then inspect the form in Design Mode to see how different View properties are set.

A number of properties can be set up for dynamic evaluation. Field properties are normally set using the Properties window of Forms Designer. For instance, the Caption property for a Label specifies the text to display on the screen. The figure below shows a Property Window for a Label object.



Property	Value
Name	
Caption	Caption
Caption Condition	
Input	
X	74
Y	8
Height	4
Width	63
Visible	Yes
Visible Condition	
Font	Helvetica
ForeColor	Black
ForeColor Condition	
BackColor	Black
BackColor Condition	
Bold	No
Bold Condition	
Italic	No
Italic Condition	
Justification	Left
Opaque	No
FontIncrease	0
FontIncrease Condition	
OutputConv	

Figure 5-1: Properties Window

Notice that under the "Caption" property there is a property named "Caption Condition." This associated property allows you to specify a condition evaluated dynamically to compute a caption for the label.

Similarly, notice the other property pairing in the Property Window:

- "Visible" and "Visible Condition"
- "ForeColor" and "ForeColor Condition"
- "BackColor" and "BackColor Condition"
- "Bold" and "Bold Condition"
- "Italic" and "Italic Condition"
- "FontIncrease" and "FontIncrease Condition"

Other components may have these and other “property” and “property condition” pairs. In each case, you are able to specify a special conditional statement that specifies the run-time value for the property.

Specifying property conditions

There are three types of conditional statements that can be specified for a property:

- Field Comparisons
- Field Value Matching
- Dynamic Functions

Field comparisons

The first type of conditional statement is a field comparison. Field comparison statements have the following syntax:

```
FIELD (=|<>) VALUE
```

where:

FIELD is any variable or file field surrounded by brackets, for example, [$x], [name], and so forth.

VALUE is any quoted string or number.

For example, suppose you want to make a certain field visible only when the category of a ticket equals “network.” Assume that the category can be selected via a drop-down list whose Input is defined as *$category*. You can specify the visibility dependency by assigning the following to the Visible Condition property of the field:

```
[$category] = "network"
```

Note that it is also conceivable that instead of a variable, a drop-down list might use a record field as input. If the field is named “category,” you would then set up the “Visible Condition” as follows:

```
[category] = "network"
```

Important: In order for a property to depend on a variable (or record field), the variable *must* be assigned as an input to an object on the same screen.

Also, note that in addition to a comparison of equality, you can specify an inequality comparison, as in:

```
[$category] <> "network"  (i.e., category is not network)
```

Complex conditions

Conditions can be complex conditions as well. For example,

```
[$catgegory]="network" & [$subcatgegory]="modem"
```

will be only true if both parts of the condition are true.

```
[$catgegory]="network" | [$catgegory]="DEFAULT"
```

will be true if either part of the condition is met.

Field value matching

The second type of statement allowed in a property condition allows you to specify matching output values to various input values. The syntax is as follows:

```
FIELD ? ValueList : DisplayList
```

where:

`FIELD` is any variable or file field surrounded by brackets, for example, [$x], [name], and so forth.

`ValueList` and `DisplayList` are lists of comma-separated values

As an example, suppose you want the color of a label to change based on the value of a severity field. The severity is specified in an edit field whose input is *$severity*. You set the Color Condition of the label as follows:

```
[$severity]? "sev1","sev2","sev3" : 1,7,2
```

This statement tells the label to do the following:

- Use a color value of 1 (red) when $severity=”sev1”
- Use a color value of 7 (yellow) when $severity=”sev2”
- Use a color value of 2 (green) when $severity=”sev3”

You can also specify a value to use if no match is found. For instance:

```
[$severity]? "sev1","sev2","sev3" : 1,7,2, 0
```

This tells the system to use color 0 (black) if *$severity* is none of the listed "sev" values.

Note again that it is assumed that the *$severity* field has been assigned as input to at least one field on the screen.

Dynamic functions

The third type of property condition statement allows you to call a function that evaluates a result dynamically.

Select()

The select function was designed to work against many records and grab a scalar value from each one to make a list. You can also select just one record and use an array from within that record to make a display list.

For example, you can select all the login IDs from a department to make a list:

```
select("name","operator","department",[$dept])
```

or select all the approvers for a certain CM group:

```
select("approvers","cm3groups","name",[$cm3group])
```

The syntax is as follows:

```
select( return_field, file_name, select_field, select_field_value )
```

The parameters to the function can be any string or number. One parameter may also be a variable or record field. A number of functions may be defined in the future.

lang_preference(lang)

Returns the current language table selection. The function takes as a parameter the current language value. The function takes care of saving new language settings in the scuser preference for future reference. It also forces the user interface to update its language table and repaint as necessary.

6 Designer Tips

CHAPTER

This chapter provides additional tips for designing forms that meet your specific needs. The following topics are covered:

- Form style guidelines
- ServiceCenter and bitmaps
- Graphics and animation

Form Style Guidelines

This section outlines the style guidelines used by Peregrine Systems, Inc. in form design. Users who modify their forms according to these guidelines can ensure a consistent look and feel when compared with the system as shipped. These guidelines were established with the goal of generating forms which display well on a variety of platforms and with different screen resolutions and fonts.

Understanding ServiceCenter screen layouts

The size of ServiceCenter forms and the objects on them are defined in terms of grid units. For instance, a combo box may be defined to be 36 units wide and 2 units tall. Objects are also placed on forms using the same units, for example, the same combo box may have its upper left corner at a location 5 units to the right of the form edge and 4 units beneath the top of the form.

The size of a ServiceCenter grid unit varies depending upon the currently selected ServiceCenter font. The grid unit is always defined as being half as wide as the lower case **e** in the current font, and half as tall as the lower case **e** in the current font. Thus, in a font whose letter **e** is 8 pixels wide by 12 pixels tall, the ServiceCenter grid unit will be 4 pixels wide and 6 pixels tall.

Changing the system's font will change the size of a form on the screen. Just as importantly, it may change the relative shape of the objects therein. If a screen's grid unit goes from 4x4 to 6x8, then objects on the screen will become 50% taller, but will double in width. Thus, the screen will appear to stretch and will have different proportions than it did originally.

An important point to recognize is that Windows true-type fonts are non-deterministic. Each video driver manufacturer can bundle its own hardware mapping of common fonts, and most modern video cards do so. Many manufacturers improve upon the base Microsoft Windows definition of what constitutes a particular font. Thus, the letter **e** in Arial 8 pt. bold when displayed at 640x480 on one video card may have a different metric than the same letter displayed at the same resolution on a different video card.

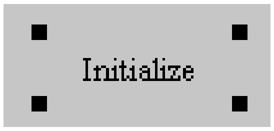
ServiceCenter default fonts

By default, ServiceCenter always launches in MS Sans Serif 8 pt. on a Microsoft Windows client. On other platforms, ServiceCenter uses Helvetica 8 pt. Bold. The only exception to this rule is that ServiceCenter will default to Arial 7 pt. Bold if launched on a 640x480 machine. Users can, of course, change this font after they log into ServiceCenter and their preference will be saved.

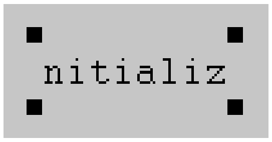
Using the Courier font

It is best to initially lay out your form using the Courier font. This font is a fixed font that offers the widest possible character spacing. Otherwise, you run the risk of creating objects that are too small or that overlap.

For example, suppose you create the label below using the Times font. The caption fits well within the label’s boundaries.



However, Times is a proportional font that compresses character spacing. Later, if a user views the label using a wider system font such as Courier, the caption will not fit.



Style guidelines

Form Size

The right edge of any form should be at grid unit 156. The bottom of any form should be at grid unit 42. Forms which follow this guideline will fit in the default fonts at 640x480 and 800x600. Additionally, these forms provide enough room along the edges for a scroll bar if required.

Label Text

All label text is to be black. Labels are to be left aligned and end in a full colon.

Combo Boxes

Combo boxes are to have 8 drop down lines unless the drop-down set contains a fixed number of elements <=7, in which case the combo box is to have box lines equal to the number of elements in the drop down set.

Check Boxes

Check boxes are to have label text on the right.

Find/Fill Boxes

Find and fill boxes are to use bitmap id=7. Under NO circumstances should a find or fill use the down array bitmap that makes the find or fill look like a combo box.

Tabbed Notebook

Forms with tabs should include a header section which is not part of the tab. This header should include important description information about the record. For example, problem tickets should show, at a minimum, the problem number.

Do not nest tabs within tabs.

Buttons

Navigational buttons on forms should be placed along the left edge or top border of the form. No button that is on the tool tray should have a replicate button on the form itself.

QBE Formats

All GUI mode QBE formats are to use the new table-type QBE format. Column headings should follow in book title capitalization rules, for example, *File Name* not *file name* or **file.name**.

Menu Labels

Labels on menu forms (for example, menu.gui.home) should have a font increase of 2.

Tab Stops

Read-only fields should have a tab-stop value of -1 to prevent users from tabbing into them.

Rules of good screen design

- Avoid excessive color. The human eye is keyed to look for edges in black and white, and whole images in color. If a form contains too many different colors, the human eye usually ceases to view it as text and instead tends to view it as art.
- Avoid pale colors. The human eye keys for contrast. Pale colors such as yellow, pink, and so forth do not contrast well enough to be readily visible.
- Line objects up, even if it means stretching some objects bigger than you think they should be. When the human eye scans an object, it instinctively follows edges. If most of the edges in your form line up vertically, then the form will appear neat and organized. In contrast, failure to line up the edges of your controls can make your forms appear sloppy.
- Cluster similar data together. If you have ten fields that all deal with problem ticket update history, place them near one another on the form. This simplifies the process of finding data for your user.

- Use tabs, frames, and bevels to group data into discreet units. Large forms with many fields can seem overwhelming to many users. By drawing enclosing shapes around groups of objects, you allow the human brain to process the set of container objects rather than the full set of on-screen objects. This makes the screen appear simpler.
- Use combo boxes, not Finds. If your users are to select from a set of *n* objects, and *n* is less than or equal to 100, then use the global list manager to build a list and use that list and a combo box to allow the user choices. Find and Fill are unfamiliar processes for most users and should be avoided where possible.
- Do not use a tabbed notebook unless necessary. The tabbed notebook breaks up data so that users cannot view it all at once. A single form is preferable to one with tabs.
- Use tabs instead of views. Views require that users change screens to see extra data. A tabbed form is preferable to a form with multiple views.

ServiceCenter and Bitmaps

ServiceCenter can store bitmaps directly in the ServiceCenter database system, regardless of the type of physical database being used. Thus, ServiceCenter can store bitmaps natively in Oracle, Sybase, SQL Server, DB/2 or the P4 system.

These bitmaps are stored inside of a file named SYSBLOB. The structure of the SYSBLOB file is as follows:

Field Name:	Description.
Application:	The table with which this bitmap is associated.
Topic:	The key value from the table with which this bitmap is associated. Usually the Unique key.
Type:	The type of data stored here. Bitmaps are type 3.
Segment:	The segment counter for this section of the bitmap.
Data:	The data block for this section of the bitmap.

Bitmaps are typically large objects, i.e., > 32 KB. In order to store large objects such as bitmaps inside the ServiceCenter system, ServiceCenter writes BLOB segments out at 30K each. ServiceCenter will create as many BLOB segments as necessary to hold the complete data.

Each bitmap is associated with a particular table and is further associated with a particular key value from the table. If this key value is unique, then the bitmap will display for only one record. However, it is possible to associate a bitmap with a non-unique key, in which case all records in a particular table which have that key value will share that bitmap. For example, a bitmap associated with the **Type** field in the device table would be stored only once, but displayed for every device of a particular type.

Adding a bitmap to an object

Use the following procedure to add a bitmap:

1. Access a form.
2. Open Forms Designer.
3. Using the Picture tool, draw a box on the form that is the size and shape of the bitmap you would like displayed.
4. Set the Input property of the box to the key field with which you would like to associate this bitmap (for example, to associate a bitmap with the **contact.name** field in the **contacts** file, set the image controls input to **contact.name**.
5. Save the form. Your form is now ready to accept bitmaps.
6. Bring up the form in Database Manager or another editor.
7. Place your cursor over the box you have drawn to contain the bitmap. Right click to bring up a menu and select **Insert bitmap** from the menu.
8. A new entry in the right click menu at the bottom will read **Insert bitmap**.
9. In the file location box that appears, select a bitmap image to insert.
10. Save the record, and the associated bitmap will be saved to the file as well.

Note: RLE or other compressed bitmaps are not supported. Only native Microsoft Windows DIB bitmaps are supported.

Graphics and Animation

You can use Microsoft Windows bitmaps to customize ServiceCenter with still images, 3D effects, and animation. Images can be mapped to pictures and buttons.

The BMP file format supports many different levels of color resolution. Forms Designer supports them all, however 4-bit color (16 colors) is the most efficient for Forms Designer to display. You can use bitmaps with 256 or more colors, but Forms Designer must dither them at runtime. This extra processing can degrade the appearance of the image and degrade client performance.

Optimum performance and image appearance

For optimum performance, use a graphics utility to convert 256 and higher-color bitmaps to 16 colors before using them in Forms Designer.

If the size of a bitmap exceeds the dimensions of its button or picture object, Forms Designer may resize the image at run time to fit the object. This extra processing at runtime can degrade the appearance of the image and degrade client performance.

For optimum performance, use objects that are at least as large as the images they display. Another option is to use the graphics utility to resize large images before using them in Forms Designer.

Text and graphics

To include text and graphics in an object, use a graphics editing package to add the text and graphics to the Microsoft Windows bitmap.

7 CHAPTER

The Form Wizard

The Form Wizard allows you to create forms based on a particular Database Dictionary. You can automatically create both table-type QBE lists and single-record display forms.

This feature is used only to create new forms and cannot be used to modify existing forms. The example steps shown are from a system administrator login.

This chapter describes how to:

- Access the Form Wizard.
- Create a form using the Form Wizard.

Accessing the Form Wizard

To access the Form Wizard:

1. Click the Toolkit tab located on the Main Menu. The selections available in the Toolkit are displayed.
2. Click the **Forms Designer** button. The initial Forms Designer screen appears.
3. Enter a name for the form in the **Form** field.
4. Click the **New** button.
5. The system asks if you want to use the Form Wizard to create your new form.
6. Click **Yes** to use the Form Wizard.

7 Enter the name of the format to add.

8 Click **OK** or press Enter.

9 Enter the name of the file for which you want to create a format when prompted, or select a file name from the drop-down list.

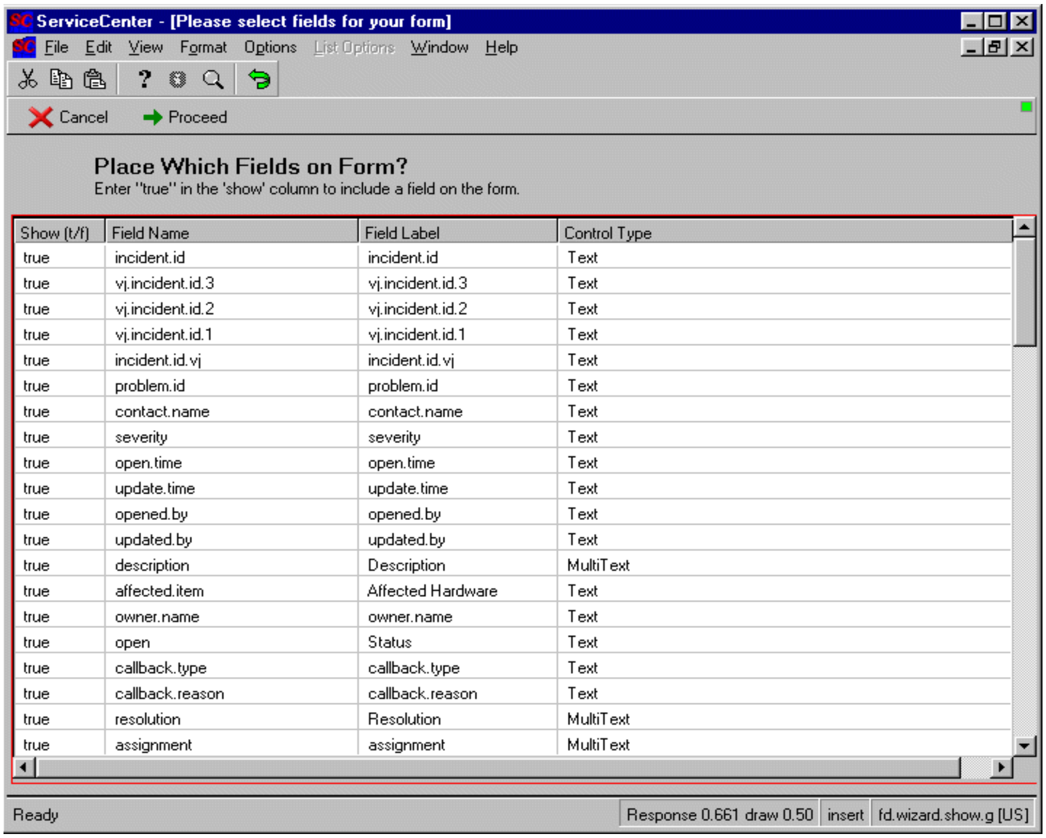
10 Click **OK** or press Enter.

The system asks what type of format you want to create.

11 Click the **down arrow** button and select a format from the drop-down list. Choose **List of Records** or **Detail of a Single Record**.

12 Click **OK** or press Enter.

The **fd.wizard.show.g** form appears.


ServiceCenter - [Please select fields for your form]
File Edit View Format Options List Options Window Help
Cancel
Proceed

Place Which Fields on Form?

Enter "true" in the 'show' column to include a field on the form.

Show (t/f)	Field Name	Field Label	Control Type
true	incident.id	incident.id	Text
true	vj.incident.id.3	vj.incident.id.3	Text
true	vj.incident.id.2	vj.incident.id.2	Text
true	vj.incident.id.1	vj.incident.id.1	Text
true	incident.id.vj	incident.id.vj	Text
true	problem.id	problem.id	Text
true	contact.name	contact.name	Text
true	severity	severity	Text
true	open.time	open.time	Text
true	update.time	update.time	Text
true	opened.by	opened.by	Text
true	updated.by	updated.by	Text
true	description	Description	MultiText
true	affected.item	Affected Hardware	Text
true	owner.name	owner.name	Text
true	open	Status	Text
true	callback.type	callback.type	Text
true	callback.reason	callback.reason	Text
true	resolution	Resolution	MultiText
true	assignment	assignment	MultiText

Ready | Response 0.661 draw 0.50 | insert | fd.wizard.show.g [US]

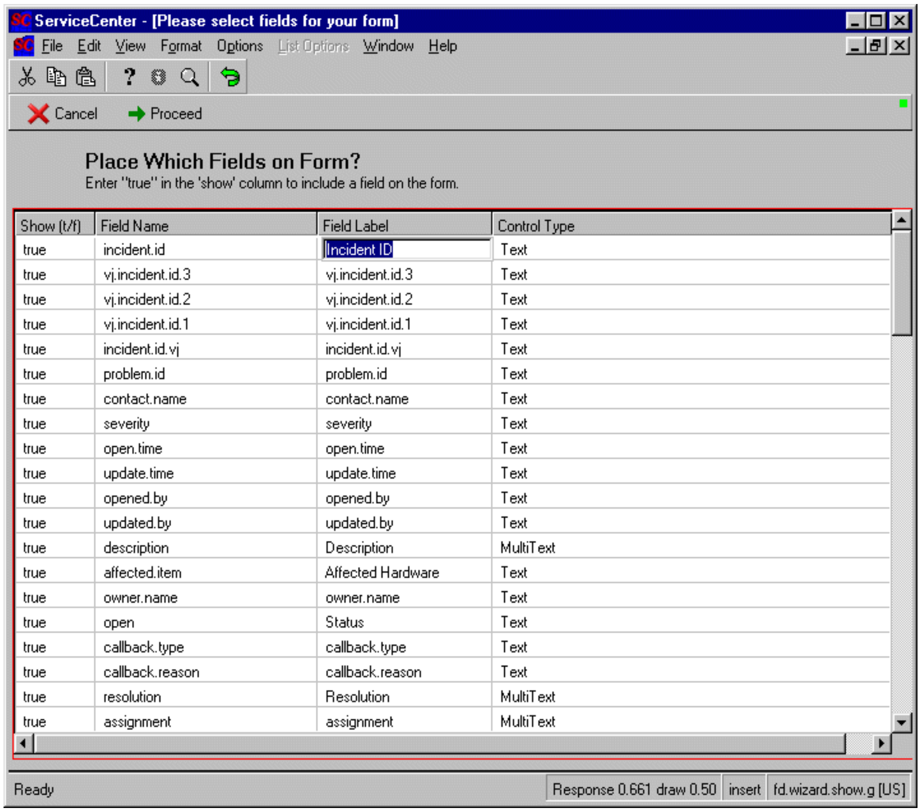
Figure 7-1: Fd.wizard.show.g Form

Creating a Form Using the Form Wizard

The following example continues the process of using the Form Wizard. In the example, Detail of a Single Record was chosen as the type of format to use.

To use the Form Wizard:

1 With the **fd.wizard.show.g** form displayed (refer to *Accessing the Form Wizard* on page 141), click in the fields you want to modify. All of the fields except **Control Type** are text boxes, which means you click in the table cell and type a new value to change the field.


ServiceCenter - [Please select fields for your form]
File Edit View Format Options List Options Window Help
Cancel
Proceed

Place Which Fields on Form?

Enter "true" in the 'show' column to include a field on the form.

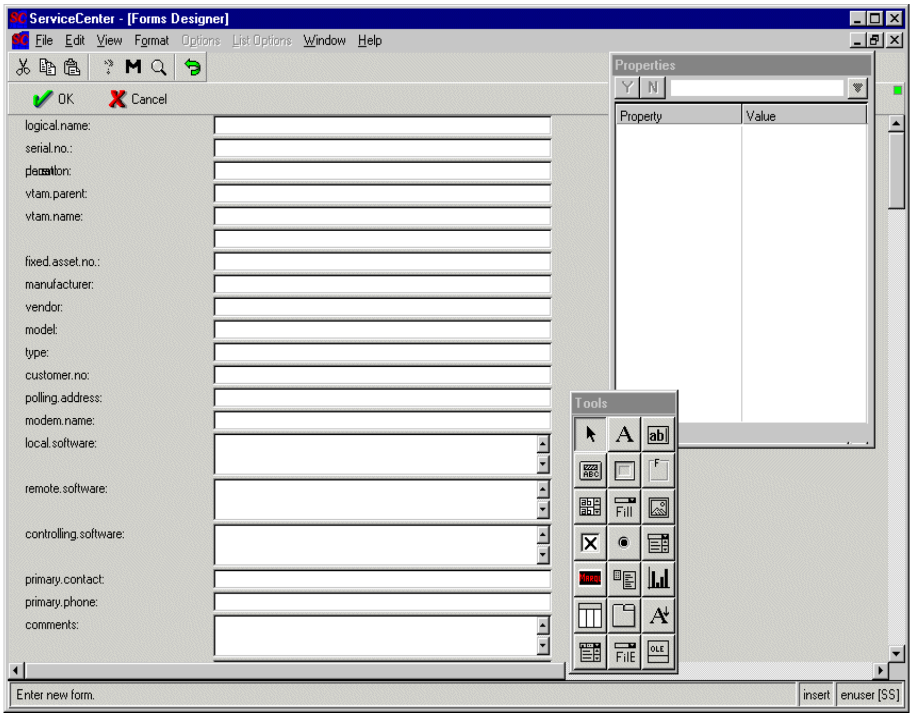
Show (t/f)	Field Name	Field Label	Control Type
true	incident.id	Incident ID	Text
true	vj.incident.id.3	vj.incident.id.3	Text
true	vj.incident.id.2	vj.incident.id.2	Text
true	vj.incident.id.1	vj.incident.id.1	Text
true	incident.id.vj	incident.id.vj	Text
true	problem.id	problem.id	Text
true	contact.name	contact.name	Text
true	severity	severity	Text
true	open.time	open.time	Text
true	update.time	update.time	Text
true	opened.by	opened.by	Text
true	updated.by	updated.by	Text
true	description	Description	MultiText
true	affected.item	Affected Hardware	Text
true	owner.name	owner.name	Text
true	open	Status	Text
true	callback.type	callback.type	Text
true	callback.reason	callback.reason	Text
true	resolution	Resolution	MultiText
true	assignment	assignment	MultiText

Ready | Response 0.661 draw 0.50 | insert | fd.wizard.show.g [US]

Figure 7-2: Fd.wizard.show.g Form

2 Change the **true** statement to **false** if you do not want a particular field included on the completed form.

3 Click **Proceed**. The form is displayed in Forms Designer.


ServiceCenter - [Forms Designer]
File Edit View Format Options List Options Window Help
OK
Cancel
Properties
Y N
Property
Value
Tools

logical.name:
serial.no.:
location:
vtam.parent:
vtam.name:
fixed.asset.no.:
manufacturer:
vendor:
model:
type:
customer.no:
polling.address:
modem.name:
local.software:
remote.software:
controlling.software:
primary.contact:
primary.phone:
comments:

Enter new form. insert enuser [SS]

Figure 7-3: New Form

4 Click **OK** to view the completed form.

5 Click **OK** again to save the form.

8 CHAPTER

What is Format Control?

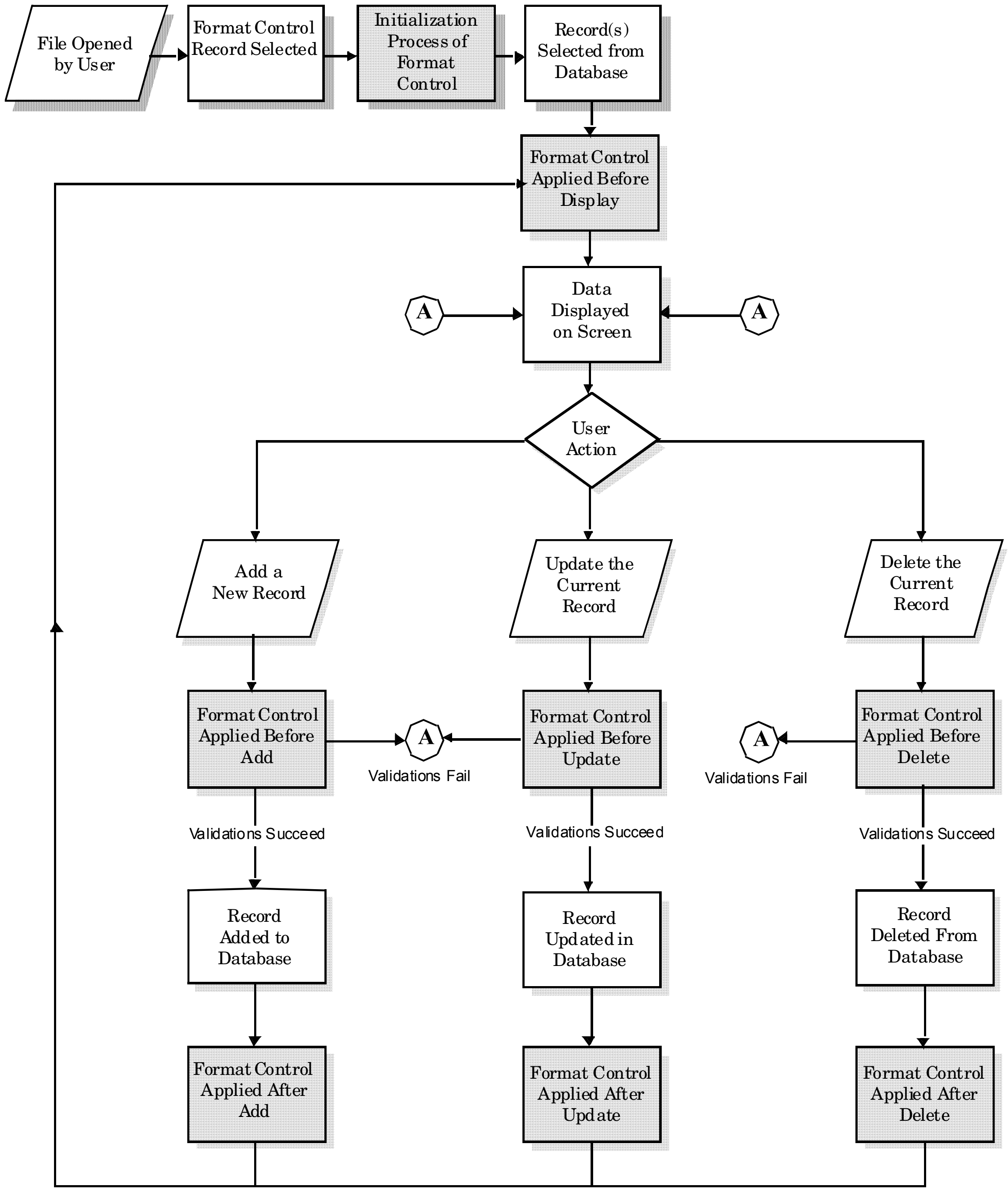
Format Control allows the System Administrator to apply the following special processes to ServiceCenter files through individual forms:

- Validate fields.
- Establish user privileges.
- Display alternate QBE forms.
- Do Calculations and Validations based on other fields in this file or other files.
- Call RAD routines.
- Define additional options and menu items.
- Automatically update or insert data in other parts of the database.

Format Control records can be attached to any form or file within ServiceCenter and do not require special programming skills to implement. Routines defined in Format Control can be user interactive or transparent and are performed when a record is displayed or when a user adds or updates a record or deletes a record from the database. Format Control is easily applied and changed.

Format Control is intended as a convenient utility and should not be overused. Excessive reliance on Format Control to modify your system can result in reduced system performance. If you intend to heavily customize your system, we recommend you find other, more efficient ways to implement your changes such as Data Policy or Data Validation and limit the use of Format control.

Format Control Process Flow


File Opened by User
Format Control Record Selected
Initialization Process of Format Control
Record(s) Selected from Database
Format Control Applied Before Display
Data Displayed on Screen
A
A
User Action
Add a New Record
Update the Current Record
Delete the Current Record
Format Control Applied Before Add
A
Validations Fail
Format Control Applied Before Update
A
Validations Fail
Format Control Applied Before Delete
Validations Succeed
Validations Succeed
Validations Succeed
Record Added to Database
Record Updated in Database
Record Deleted From Database
Format Control Applied After Add
Format Control Applied After Update
Format Control Applied After Delete

Format Control Processes

There are eight major functional *processes* in Format Control that define the actions to be taken on a record. Each of these processes is represented by a separate form within the Format Control Utility.

Main Information

This form is the entry point in Format Control and has several functions:

- Initializes fields or variables that will later be used in the Format Control record.
- Initially displays a value in a field when the record itself is displayed.
- Names special QBE and initial query forms to use, sets up default sort sequences, and runs scripts.

Views

This process allows the user to display multiple QBE views of ServiceCenter forms based on conditions evaluated at run time.

Queries

Queries are generally used to extract information from a file other than the primary file in order to perform calculations and validations and to report on information from more than one file.

Calculations

The Calculations process can be used whenever you want to perform a calculation on currently available fields or variables. The fields needed for calculations may be variables, fields in the primary file, or fields in any other secondary files which may have been queried.

Validations

The Validation process allows you to set up a logical expression for checking data in fields or variables on the form.

The validation expression you set up must evaluate to *true* upon the desired edit function for it to be successful. If the expression does not evaluate to *true*, a validation message will be displayed, and the specified operation will not be allowed.

Subroutines

This process allows you to call RAD subroutines.

Additional Options

Additional Options is used to define menu options you want available to users on any form associated with a particular Format Control record. This feature calls RAD subroutines and is available in Database Manager only.

Privileges

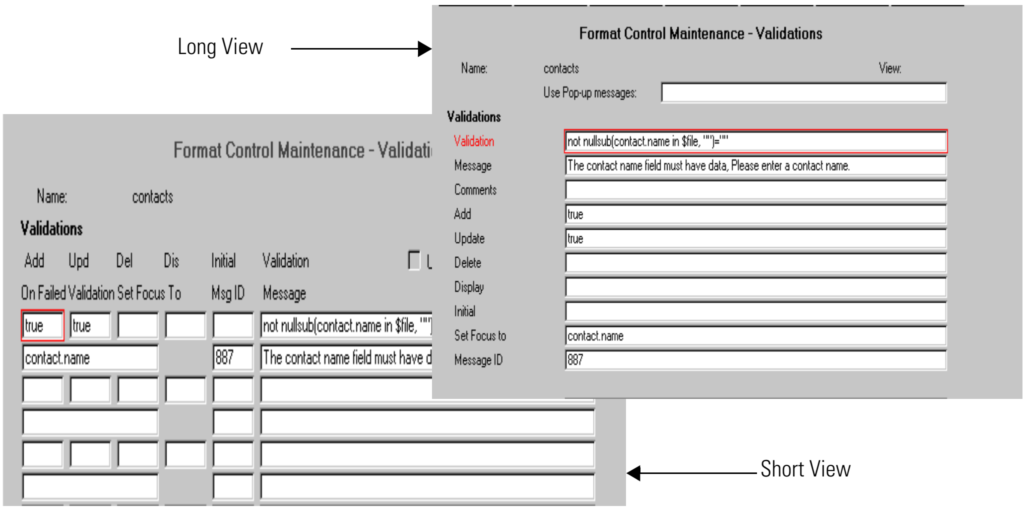
Security is used to control database options available to the user. If the contents of a field evaluate to *true* at processing time, the corresponding button will be available to the user.

Selecting a Process View

For most of the processes in Format Control, there is a *short* and a *long* view of the form. The short view is used when the Format Control information is relatively simple. The long view contains the same fields and is used when the information is more extended or complex. These views are simply two different ways of displaying the same Format Control record.

When you first select one of the Format Control processes, such as File Queries, the Short View of the format is displayed. To display the Long View, select **Show Expanded Form** from the Options menu.

If you select another process of the Format Control record (i.e. Calculations), the last view selected (long or short) is displayed as the default in the new category. A label appears on the right side of the form above the data fields indicating which view is being displayed.



Format Control Maintenance - Validations

Name: contacts View:

Use Pop-up messages:

Validations

Validation	not nullsub(contact.name in $file, "")=""
Message	The contact name field must have data, Please enter a contact name.
Comments	
Add	true
Update	true
Delete	
Display	
Initial	
Set Focus to	contact.name
Message ID	887

Format Control Maintenance - Validati

Name: contacts

Validations

Add	Upd	Del	Dis	Initial	Validation
true	true				not nullsub(contact.name in $file, ""

On Failed Validation Set Focus To	Msg ID	Message
contact.name	887	The contact name field must have d

Creating Expressions

Definitions

Primary File

The file attached to a form. This is referred to as **$file** in a Format Control record. Additional files are defined as $file1, $file2, etc. in the order of their entry in the secondary files section of the Additional File Queries screen of the Format Control record.

Semicolon

Semicolons(;) separate statements to allow them to be entered on one line.

Variable

A value that resides in memory only, rather than in a database record. Variable names always start with $. To ensure that a variable contains the proper value, initialize it in the Initializations Section of the Format Control record.

Statements

ServiceCenter supports some C-like statements. Their use will be demonstrated in various example in this chapter:

if then else

while do

for do

Note: For detailed instructions about ServiceCenter's System Language, refer to the System Language topic.

System Functions

Many of the ServiceCenter system functions are used to create expressions in Format Control. The following is a list of the most commonly used functions:

denull

Compresses an array by deleting all trailing NULL entries and returns the compressed array.

index

Returns the index or position number for a specific element value in an array or character in a string. If the target value is not in the array or string, it returns **0** (zero).

lng

Returns the number of elements in an array or structure and the number of characters in a string.

nullsub

Replaces a null value with a known value, eliminating the possibility that expressions will return null or unknown results.

operator()

Returns the logon id of the current operator.

str

Converts a non–string data type into a string.

substr

Extracts a substring from a string.

tod()

Returns the current date and time.

val

Converts a field into a numeric, logical, or date/time value.

Boolean (Logical) Fields

In the forms for Additional File Queries, Calculations, Validations, and Subroutines, you will find some or all of the following fields:

- add
- update
- delete
- display
- before

A value of *true* (or an expression that evaluates to *true*) in any of these fields will cause the Format Control functions to be executed. These functions will be executed before the edit is performed. If the fields are blank, the calculation, validation, etc. will not be performed for this file. For example, a value of *true* in the **add** field will cause Format Control functions to be performed before a record is added. If the other fields (update, delete, display) are blank, the functions will not be performed when the record is updated, deleted, or displayed.

Note: The **display** field is used for the initial display of records or when the Format Control record is used with reports.

Important: The **before** field must evaluate to *true* or *false* only. A variable WILL NOT work in this field.

Normally, these fields would contain *true*, *false*, or *blank*, or they could contain a variable that could be calculated in the Calculations process before being used in another process. When using either of these methods, use either version (short or long) of Format Control to enter an expression that will be evaluated to *true* or *false* at execution time.

This is an example of an expression that is frequently used:

```
index ("SysAdmin", $lo.ucapex)>0
```

This expression searches the user's capability array in the Operator Record to see if he/she has *SysAdmin* capabilities. The function will be executed if this evaluates to *true*, indicating that the operator has *SysAdmin* listed as a capability and therefore is authorized for the function.

File Variable

The *current* database record (the record being modified or displayed by the user) in Format Control is always identified by the variable **$file.** Elements of the current record being acted upon by Format Control are associated with **$file** by expressions. For example, <*field.name*> **in $file** might identify a particular field in the current record to which a Format Control process has been attached. The **$file** variable does NOT reference a *file,* which may use several forms to input data.

Important: In order for a property to depend on a variable (or record field), the variable *must* be assigned as an input in Format Control.

9 CHAPTER Getting Started With Format Control

Accessing Format Control

You can access the Format Control Utility from three locations in ServiceCenter:

- From Forms Designer.
- From the Database Manager (file=formatctrl).
- Directly from a menu.

Access from Forms Designer

The best method for accessing Format Control is through Forms Designer. This method reduces the chances of accessing the incorrect record because you are accessing the record directly from the form with the same name as the Format Control record.

To access Format Control from Forms Designer:

1. Click the **Forms Designer** button in the Toolkit tab in the administrator's home menu.
2. Enter the name of the form you wish to see in the **Form** field of the Forms Designer dialogue box.
3. Use the Search function to find the form.
4. Select **Format Control** from the Options menu.

   The Format Control record is displayed for the selected form.

Direct Access

You can access Format Control directly without going through Forms Designer. This method allows you to search for an existing record or create a new record.

To access Format Control directly:

1 Click the **Tools** button in the Utilities tab in the administrator's home menu.

The Tools menu is displayed.

2 Click on Format Control in the Tools tab.

A blank Format Control Maintenance - Main Information form is displayed. You can either search for an existing Format Control record or create a new record from this form.

Adding a Record

To add a record to Format Control:

1 Access the form to which the Format Control record will be associated.

2 Access Format Control from Forms Designer.

A new Format Control record is created for the form. ServiceCenter automatically places the form's name in the Name field. By default, a value is automatically entered in the **System** field (a required field) and the **Use Default Sort** check box is selected.

Note: When accessing Format Control in this manner, the name of the new record is defaulted to the same name as the form used to access it. In the rare cases where a different name is needed, you can override the default name.

3 Add the desired data. These fields are discussed in the next chapter.

4 Click on the **Add** button to add the record to the database.

5 Use the Options menu to select the appropriate Format Control process. These processes are discussed in the next chapter.

10 CHAPTER Array Maintenance

Two types of arrays exist within ServiceCenter: simple arrays, consisting of fields of similar data types; and arrayed structures, composed of related fields of similar or dissimilar data types. A series of applications exists in ServiceCenter that can be called from the Additional Options process of Format Control to perform various maintenance functions on arrayed structures. A single application, called from the Subroutines process, allows the user to sort simple arrays automatically when adding or updating records.

Arrayed Structures

The arrayed structure maintenance functions (copy, move, delete, insert) are similar to those used in the array editor found in the **Expand Array** option in the Options menu. However, in Format Control, these functions act against the entire arrayed structure element, not against a single field within the structure.

The most significant difference between the array editor and the arrayed structure editor is that the array editor re–displays the field in question in an expanded, dynamic array field. This does not occur with arrayed structures. The current form displayed when starting the function is the same one displayed during execution. The routines that perform these functions are available collectively and individually to any RAD routine or Format Control Additional Options definition.

The following general rules apply to arrayed structure maintenance:

- None of the routines perform update functions to the database. In order to save the changes, you must select the appropriate update function within the application that called the maintenance functions.
- The applications are cursor dependent. For example, the *insert* function inserts a blank element at the cursor position within the arrayed structure. If the cursor is placed in any field within an element of the arrayed structure, the function is performed against that entire element.
- If the functions are executed against a non-arrayed structure field, the procedure normally fails and either issues an invalid data-type message or produces unpredictable results.

The Arrayed Structure maintenance applications perform tasks such as moving, copying, deleting or sorting lines in arrayed structures. They are used with the Additional Options process to define special button features for files opened in Database Manager. Users working with forms containing arrayed structures will find these options useful in editing files.

The name of each application, its basic function, and its available parameters are defined. To call these applications from Format Control, refer to *Additional Options* on page 250 for a detailed explanation of how to define Additional Options and how to pass data to these subroutines.

Users should have a thorough knowledge of Forms Designer and the use of *subformats* in the creation of arrayed structures.

Note: Under normal circumstances, the application **as.options** is called. However, you can call each of the other applications as needed. RAD users should follow normal subroutine call procedures to execute these applications. (Please refer to the *RAD User Guide*).

as.options

The **as.options** application displays all available arrayed structure options on one **rio** (Record Display/Input) Command panel.

Note: Use of this application in arrayed structure maintenance provides all the controls needed to copy, insert, delete and move fields. It is not necessary to run **as.copy**, **as.move**, **as.insert**, and **as.delete** in conjunction with this application.

Parameters

file

The file variable currently being edited. This is always *$file* when called from Format Control.

name

The name of the current form when the arrayed structure maintenance applications are displayed. If left blank, the default is the result of executing the function **current.format**(). Under normal circumstances, you do not pass data to this parameter. However, you could pass the name of a special form that contains an expanded version of the arrayed structure.

cond.input

A boolean field that allows the user to edit the data record when the arrayed structure maintenance options are displayed. The default is *true*.

Note: When calling the application from Format Control, the value must be passed as a data type of **4** (logical) by using the syntax:

val("false", 4)

index

The number of lines in the *subformat* that displays the arrayed structure. For instance, if the arrayed structure subformat has data on lines 1 and 2, then 2 should be passed to this parameter. The default is 1.

Note: This is NOT the window size that displays the arrayed structure. Also note that when calling the application from Format Control, the value must be passed as a number by using the syntax:

val("*n*"**, 1)** where **n** = the number of lines in the subformat, and **1** indicates a data type of *number*.

The results of the maintenance routines are unpredictable if this number does not correctly reflect the number of data lines in the subformat.

Example

Create the Format Control record that attaches the as.options application to the service form.

1 Click the **Tools** button in the Utilities tab in the administrator's home menu.

2 Click on the **Format Control** button in the Tools menu.

3 Type **service** in the **Name** field of the blank Format Control record.

4 Press **Enter**.

5 If a Format Control record does not already exist for **service**, click on the **New** button to create one. (See *Adding a Record* on page 156 for information about adding records.)

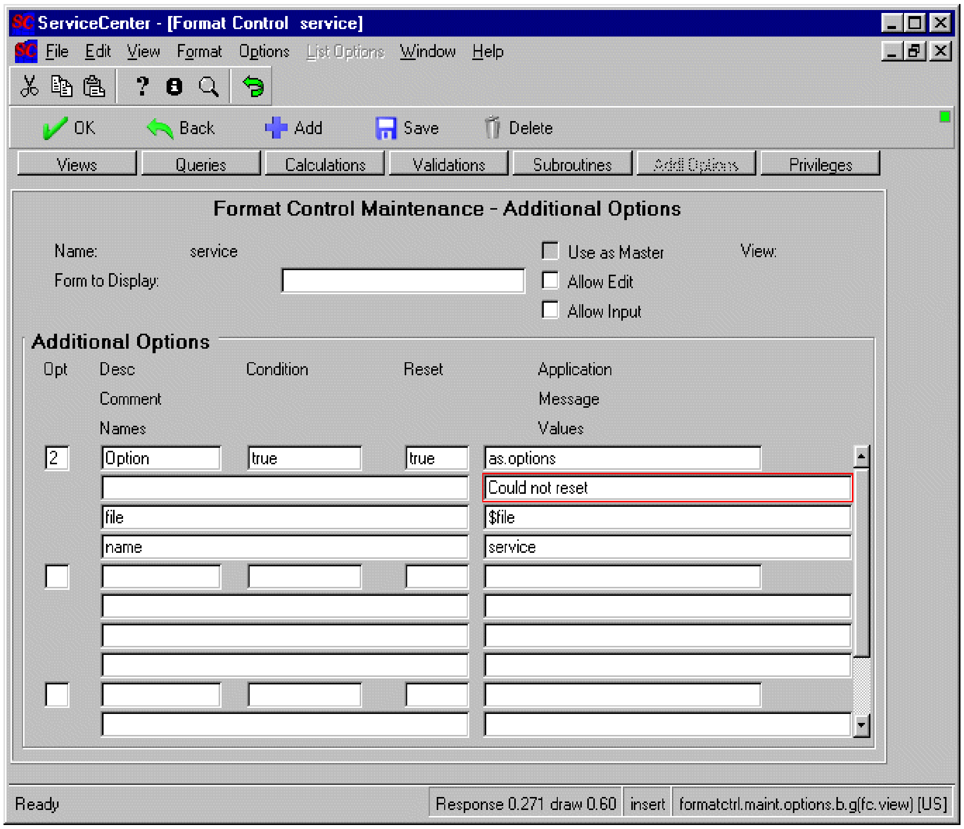
6 Select **Additional Options** from the Options menu.

Note: Refer to *Additional Options* on page 250, for details on Additional Options.

7 Enter the following values in the fields of the Additional Options process form.

Option #	2
Desc	Option
Condition	true
Reset	true
Application	as.options
Message	Could not reset
Names	file name
Values	$file service

The record appears as follows:


ServiceCenter - [Format Control service]
File Edit View Format Options List Options Window Help
OK Back Add Save Delete
Views Queries Calculations Validations Subroutines Add Options Privileges

Format Control Maintenance - Additional Options

Name: service

Form to Display:

Use as Master

Allow Edit

Allow Input

View:

Additional Options

Opt	Desc / Comment / Names	Condition	Reset	Application / Message / Values
2	Option	true	true	as.options
				Could not reset
	file			$file
	name			service

Ready | Response 0.271 draw 0.60 | insert | formatctrl.maint.options.b.g(fc.view) [US]

Figure 10-1: Format Control Record for *service* Form

Test the option you have defined for the *service* form.

1 Select the Toolkit tab in the Home menu.

2 Click on the **Database Manager** button.

3 Enter **service** in the **Form** field of the Database Manager dialogue box.

4 Press **Enter** or click on the **Search** button.

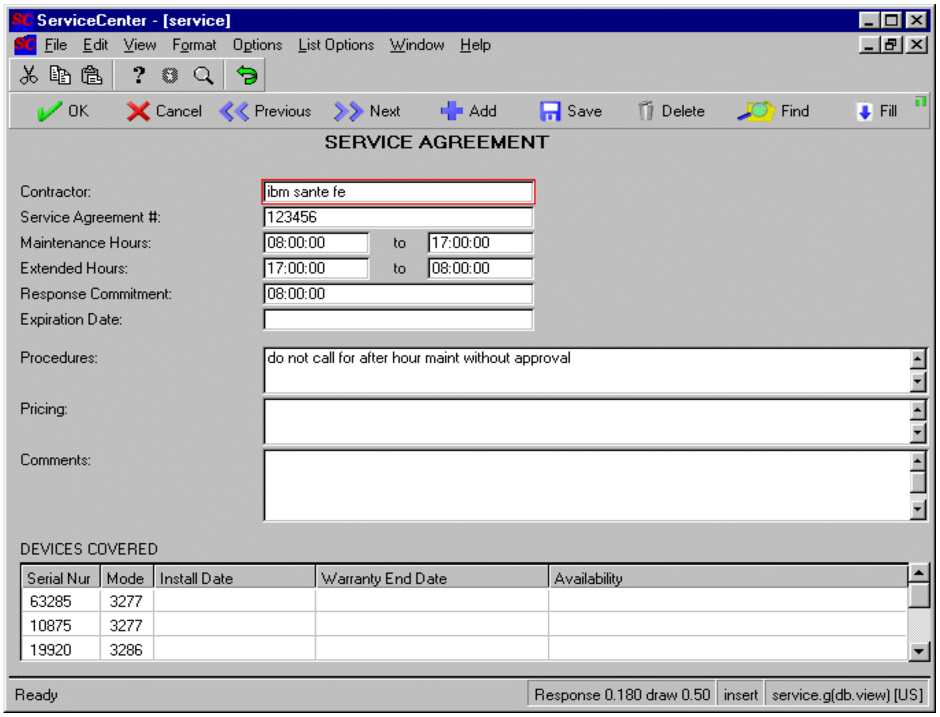
5 Select **service** from the QBE list.

6 A blank Service Agreement form is displayed.

7 Click on the **Search** button

8 Select a record from the QBE list. For this example, we have chosen **ibm santa fe** – Service Agreement # 123456.

9 The IBM Santa Fe Service Agreement record is displayed.


ServiceCenter - [service]
File Edit View Format Options List Options Window Help
OK Cancel Previous Next Add Save Delete Find Fill

SERVICE AGREEMENT

Contractor:	ibm sante fe		
Service Agreement #:	123456		
Maintenance Hours:	08:00:00	to	17:00:00
Extended Hours:	17:00:00	to	08:00:00
Response Commitment:	08:00:00		
Expiration Date:			
Procedures:	do not call for after hour maint without approval		
Pricing:			
Comments:			

DEVICES COVERED

Serial Nur	Mode	Install Date	Warranty End Date	Availability
63285	3277			
10875	3277			
19920	3286			

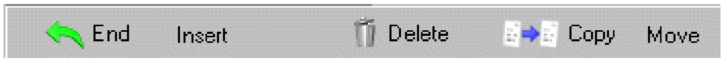
Ready | Response 0.180 draw 0.50 | insert | service.g(db.view) [US]

Figure 10-2: Service Agreement with Normal Button Options

10 Enter data in blank fields so that the records for all three devices are complete.

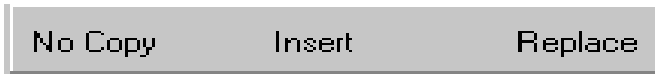
11 Select **Option** from the Options menu.

12 The System Tray in the **service** form displays the buttons associated with the as.options subroutine.



- The **Insert** button inserts a blank line in an arrayed structure above the cursor position.
- The **Delete** button deletes the line in which the cursor is placed.
- The **Copy** button copies the entry from the line selected by the cursor. The original element is left in place when the copy is inserted into the arrayed structure.

- The **Move** button copies the entry from the line selected by the cursor. The original element is deleted when the copy is inserted into the arrayed structure.
- The following System Tray buttons are displayed for completing the **copy** and **move** functions:



- The **Insert** button inserts the copied element into the arrayed structure above the line selected by the cursor.
- The **Replace** button replaces the element selected by the cursor with the copied element.

13 Click on the **End** button to restore the normal tray buttons.

14 Click on OK or **Save** to save any changes you have made to the Service Agreement record.

Important: You must use the OK or **Save** button to update your record. Clicking on the **Enter** button will NOT save any changes you have made.

as.copy

The **as.copy** application copies an element from one position to another, leaving the original in place.This application is used in conjunction with **as.delete**, **as.insert**, and **as.move.**

Parameters

file

The file variable currently being edited. This is always *$file* when called from Format Control.

name

Under normal circumstances, data is not passed to this parameter. The default is the result of executing the function **cursor.field.name**(1). The application **as.get.name** is then executed against this result.

Note: You can pass the name of a specific arrayed structure. If the data passed to this parameter does not contain any commas, it is considered to be an arrayed structure name. If a comma is found, the application *as.get.name* is executed against the data.

index

The number of lines (height property) in the *subformat* that displays the arrayed structure. For instance, if the arrayed structure subformat has data on lines 1 and 2, then 2 should be passed to this parameter. The default is 1.

Note: This is NOT the window size that displays the arrayed structure. When calling the **as.copy** application from Format Control, the value must be passed as a number by using the syntax:

val("*n*"**, 1**) where **n** = the number of lines in the subformat, and **1** indicates a data type of *number*.

The results of the maintenance routines are unpredictable if this number does not correctly reflect the number of data lines in the subformat. For more information, refer to the *System Language Guide*.

as.delete

The **as.delete** application deletes an element from an array. This application is used in conjunction with **as.copy**, **as.insert**, and **as.move**.

Parameters

file

The file variable currently being edited. This is always *$file* when called from Format Control.

name

Under normal circumstances, data is not passed to this parameter. The default is the result of executing the function **cursor.field.name**(**1**). The application **as.get.name** is then executed against this result.

Note: You can pass the name of a specific arrayed structure. If the data passed to this parameter does not contain any commas, it is considered to be an arrayed structure name. If a comma is found, the application **as.get.name** is executed against the data.

index

The number of lines (height property) in the *subformat* that displays the arrayed structure. For instance, if the arrayed structure subformat has data on lines 1 and 2, then 2 should be passed to this parameter. The default is 1.

Note: This is NOT the window size that displays the arrayed structure. When calling the **as.copy** application from Format Control, the value must be passed as a number by using the syntax:

val(*"n"*, **1**) where **n** = the number of lines in the subformat, and **1** indicates a data type of *number*.

The results of the maintenance routines are unpredictable if this number does not correctly reflect the number of data lines in the subformat.

as.insert

The **as.insert** application inserts a NULL element. This application is used in conjunction with **as.copy**, **as.delete**, and **as.move**.

Parameters

file

The file variable currently being edited. This is always *$file* when called from Format Control.

name

Under normal circumstances, data is not passed to this parameter. The default is the result of executing the function **cursor.field.name(1)**. The application **as.get.name** is then executed against this result.

Note: You can pass the name of a specific arrayed structure. If the data passed to this parameter does not contain any commas, it is considered to be an arrayed structure name. If a comma is found, the application **as.get.name** is executed against the data.

index

The number of lines (height property) in the *subformat* that displays the arrayed structure. For instance, if the arrayed structure subformat has data on lines 1 and 2, then 2 should be passed to this parameter. The default is 1.

Note: This is NOT the window size that displays the arrayed structure. When calling the **as.copy** application from Format Control, the value must be passed as a number by using the syntax:

val(*"n"*, **1**) where **n** = the number of lines in the subformat, and **1** indicates a data type of *number*.

The results of the maintenance routines are unpredictable if this number does not correctly reflect the number of data lines in the subformat.

as.move

The **as.move** application moves an element from one position in an array to another by creating a copy and then deleting the original. This application is used in conjunction with **as.copy**, **as.insert**, and **as.delete**.

Parameters

file

The file variable currently being edited. This is always *$file* when called from Format Control.

name

Under normal circumstances, data is not passed to this parameter. The default is the result of executing the function **cursor.field.name**(1). The application **as.get.name** is then executed against this result.

Note: You can pass the name of a specific arrayed structure. If the data passed to this parameter does not contain any commas, it is considered to be an arrayed structure name. If a comma is found, the application **as.get.name** is executed against the data.

index

The number of lines (height property) in the *subformat* that displays the arrayed structure. For instance, if the arrayed structure subformat has data on lines 1 and 2, then 2 should be passed to this parameter. The default is 1.

Note: This is NOT the window size that displays the arrayed structure. When calling the **as.copy** application from Format Control, the value must be passed as a number by using the syntax:

val("**n**", **1**) where **n** = the number of lines in the subformat, and **1** indicates a data type of *number*.

The results of the maintenance routines are unpredictable if this number does not correctly reflect the number of data lines in the subformat.

Example

Create a Format Control Record

1. Click the **Tools** button in the Utilities tab.
2. Click on the **Format Control** button in the Tools menu.
3. Type **service** in the **Name** field of the blank Format Control record.
4. Press **Enter**.
5. If a Format Control record does not already exist for **service**, click on the **New** button to create one. (See *Adding a Record* on page 156 for information about adding records.)
6. Select **Additional Options** from the Options menu.

   Note: Refer to *Additional Options* on page 250, for details on Additional Options.

7. Enter the following values in the fields of the Additional Options process form:.

Option #	Condition	Desc	Application	Reset on Return	Names	Values
1	true	Copy	as.copy	true	file index	$file val("18", 1)
2	true	Delete	as.delete	true	file index	$file val("18", 1)
4	true	Insert	as.insert	true	file index	$file val("18", 1)
5	true	Move	as.move	true	file index	$file val("18", 1)

Error Message for each subroutine call above:

Could not execute.

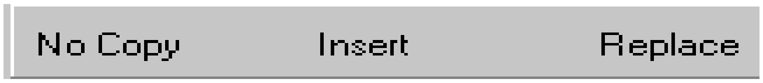
8. Click **OK** when finished.
9. Exit Format Control.

Test the Form

Test the effects of the Format Control options you have defined:

1 Select the Toolkit tab in the Home menu.
2 Click on the **Database Manager** button.
3 Enter **service** in the **Form** field of the Database Manager dialogue box.
4 Press **Enter** or click on the **Search** button.
5 Select **service** from the QBE list.
6 A blank Service Agreement form is displayed.
7 Click on the **Search** button
8 Select **ibm santa fe** (Service Agreement # 123456) from the QBE list.
9 The IBM Santa Fe Service Agreement record is displayed.
10 Enter data in blank fields so that the records for all devices are complete.
11 Position the cursor within the arrayed structure to identify an insertion, deletion, move or copy point.
12 From the Options menu, select any of the standard options or any of the four new options defined in the Format Control record.

- The **Insert** option inserts a blank line in an arrayed structure above the cursor position.
- The **Delete** option deletes the line in which the cursor is placed.
- The **Copy** option copies the entry from the line selected by the cursor. The original element is left in place.
- The **Move** option copies the entry from the line selected by the cursor. The original element is deleted after the copy is inserted into the arrayed structure.
- The following System Tray buttons are displayed for completing the **copy** and **move** functions:



- The **Insert** button inserts the copied element into the arrayed structure above the line selected by the cursor.
- The **Replace** button replaces the element selected by the cursor with the copied element.

13 Click **OK** or **Save** to save any changes you have made to the Service Agreement record.

Note: You must use the **Save** button to update your record. Clicking the **Enter** button will NOT save any changes you have made.

as.get.name

The **as.get.name** application returns the structure name of the arrayed structure in which the cursor is located.

Parameters

name

The structure name of the arrayed structure. The default is the result of executing the function **cursor.field.name**(1).

prompt

The returned structure name of the arrayed structure. This is a data return parameter. Data should not be passed to it.

as.sort

The **as.sort** application sorts an arrayed structure based on the contents of a particular field within the arrayed structure.

This application supports *major*, *intermediate*, and *minor* sorting by allowing you to sort multiple arrayed structure fields with one pass of the application. You can also specify an *ascending* or *descending* sort order. You must call this application from Format Control or RAD on an as–needed basis.

Parameters

file

The file variable currently being edited. This is always *$file* when called from Format Control.

name

The name of the arrayed structure you wish to sort. This name corresponds to the **input** field of the subformat object in the form to which the Format Control record is attached. (For information on creating subformats, refer to the *Forms Designer* documentation.)

Note: If the data passed to this parameter does not contain any commas, it is considered to be an arrayed structure name. If a comma is found, the application **as.get.name** is executed against the data.

numbers

An array that specifies the field numbers on which to sort within the arrayed structure. This is a required field. If left NULL, the sort routine ends. The data must be passed as an array of numbers. In Format Control, you must first define a *$variable* in the Initializations form for the array of numbers, then pass that variable to the **numbers** input field in the Additional Options definition. There are three levels of sorting: *major*, *intermediate*, and *minor*. For example, if you pass the array {6, 1, 3}, field 6 is the major sort, field 1 is the intermediate sort, and field 3 is the minor sort.

Important: Field numbers correspond to the *index numbers* listed in the Database Dictionary and NOT to the sequence in which they appear on the form itself.

condition

A boolean array that specifies either *ascending* or *descending* order for each field being sorted in the arrayed structure (as specified in the *numbers* parameter). To specify an ascending order, pass a value of *true* (the default). To specify a descending order, pass a value of *false*. Each element in this array has a direct correlation with the elements of the *numbers* parameter. For example, to sort the fields, 3, 1, and 6 of an arrayed structure in ascending, descending, and ascending order, the *condition* parameter is {true, false, true}. In Format Control, you must first define a *$variable* in the Initializations screen that defines the boolean array, then pass the variable to this input field in the Additional Options definition.

Example

1. Click the **Tools** button in the Utilities tab.
2. Click on the **Format Control** button in the Tools menu.
3. Type **service** in the **Name** field of the blank Format Control record.
4. Press **Enter**.
5. If a Format Control record does not already exist for **service**, click on the **New** button to create one.(See *Adding a Record* on page 156 for information about adding records.)

6 Enter the following expressions in the **Initialization Expressions** field of the Initializations form:

$numbers={4, 2, 3}

$array={false, true, false}

Note: You must leave a space after each comma in the expressions above.

The **Warranty** field (4) sorts first in descending order, the **Model** field (2) sorts in ascending order, and the **Install** field (3) sorts in descending order.

7 Select **Additional Options** from the Options menu.

Note: Refer to*Additional Options* on page 250 for details on the Additional Options function.

8 A prompt box will appear asking if you want to save the changes to the Format Control record. Click on the **Yes** button to save your changes.

9 Select **Show Expanded Form** from the Options menu.

10 Enter the following values in the fields of the Additional Options process form. To enter the third and fourth parameter values, position the cursor on the first values field (where you entered the value $file) and press PgDn.

Option #	1
Condition	true
Command	Sort
Application	as.sort
Reset	true
Names	file
	numbers
	name
	condition
Values	$file
	$numbers
	devices.covered
	$array

Error Message

Could not execute.

Format to Display: Leave this field blank. (If left blank, the value of this field will default to the current form—the form you want to sort.)

Note: The value, **devices.covered**, is the name of the arrayed structure to be sorted. This value must match the **input** value of the subformat object in the form to which the Format Control record is attached. (For more information about creating subformats, refer to the *Forms Designer* documentation.)

11 Click **OK.**

12 Exit Format Control.

Test the sorting order you have defined for the *service* form.

1 Select the Toolkit tab in the Home menu.

2 Click the **Database Manager** button.

3 Enter **service** in the **Form** field of the Database Manager dialogue box.

4 Press **Enter** or click on the **Search** button.

5 Select **service** from the QBE list.

6 A blank Service Agreement form is displayed.

7 Click the **Search** button.

8 Select **ibm santa fe** (Service Agreement # 123456) from the QBE list.

9 The IBM Santa Fe Service Agreement record appears.

10 Enter data in blank fields so that the records for all devices are complete.

11 Place the cursor in the arrayed structure.

12 Select **Sort** from the Options menu to execute the sort.

13 Click on the **Save** button to save any changes you have made to the Service Agreement record.

Important: You must use the **Save** button to update your record. Clicking on the **Enter** button will NOT save any changes you have made.

Sorting Simple Arrays

The **sort.array** application is a utility called from Format Control to sort simple arrays in either ascending or descending order. Supported array data types are:

- number
- character
- data/time

The **sort.array** utility is called from the Subroutines process of Format Control. (Refer to *Subroutines* on page 240 for information about Subroutines.) Use this utility to sort simple arrays in ascending or descending order

Example

This example has three parts:

- Create the form.
- Create a Format Control record.
- Create a data record.

Create a Form

Create a form called **demo.sort** with three simple arrays of different data types: numbers, characters, and dates. Provide a **Name** field for the record and create a file in the Database Dictionary called **demosort**. (For information about creating forms, refer to the *Forms Designer* documentation.)

Use the following values to construct your form:

Field Name	**Input Value**
Name	name
Numbers	number
Characters	character
Dates	date

Your form should resemble the following:

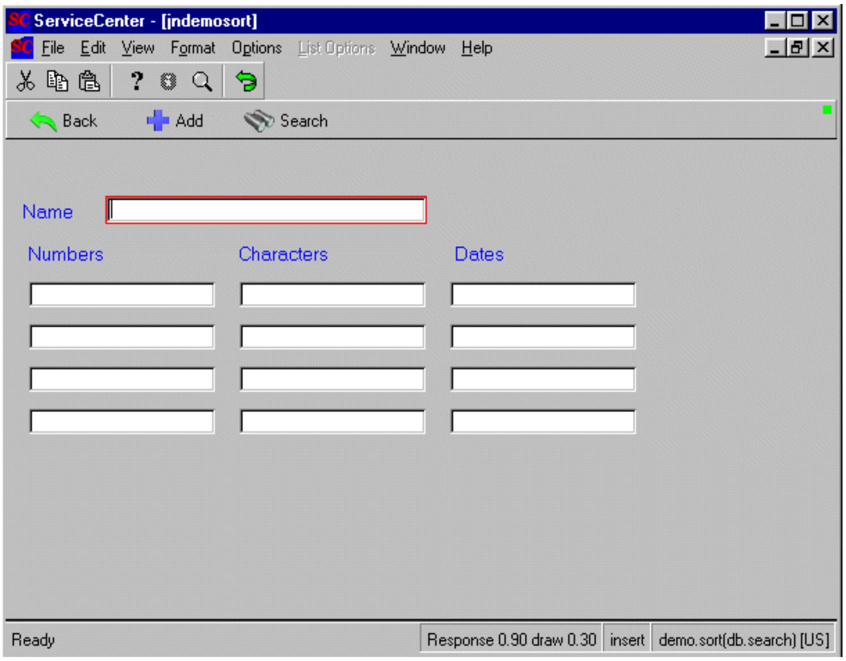

ServiceCenter - [indemosort]
File Edit View Format Options List Options Window Help
Back
Add
Search
Name
Numbers
Characters
Dates
Ready
Response 0.90 draw 0.30
insert
demo.sort(db.search) [US]

Figure 10-3: Demonstration Form for *sort.array* Utility

Create a Format Control Record

Call the sort.array application from the Subroutines process of Format Control.

1 Access the Format Control Initialization form.

2 Type demo.sort in the Name field.

3 Click **New** to create a new Format Control record for your demo form.

4 Select **Subroutines** from the Options menu or click the Subroutines button.

5 Select **Show Expanded Form** from the Options menu in the Subroutines process.

6 Enter the following values in the long form view of Subroutines:

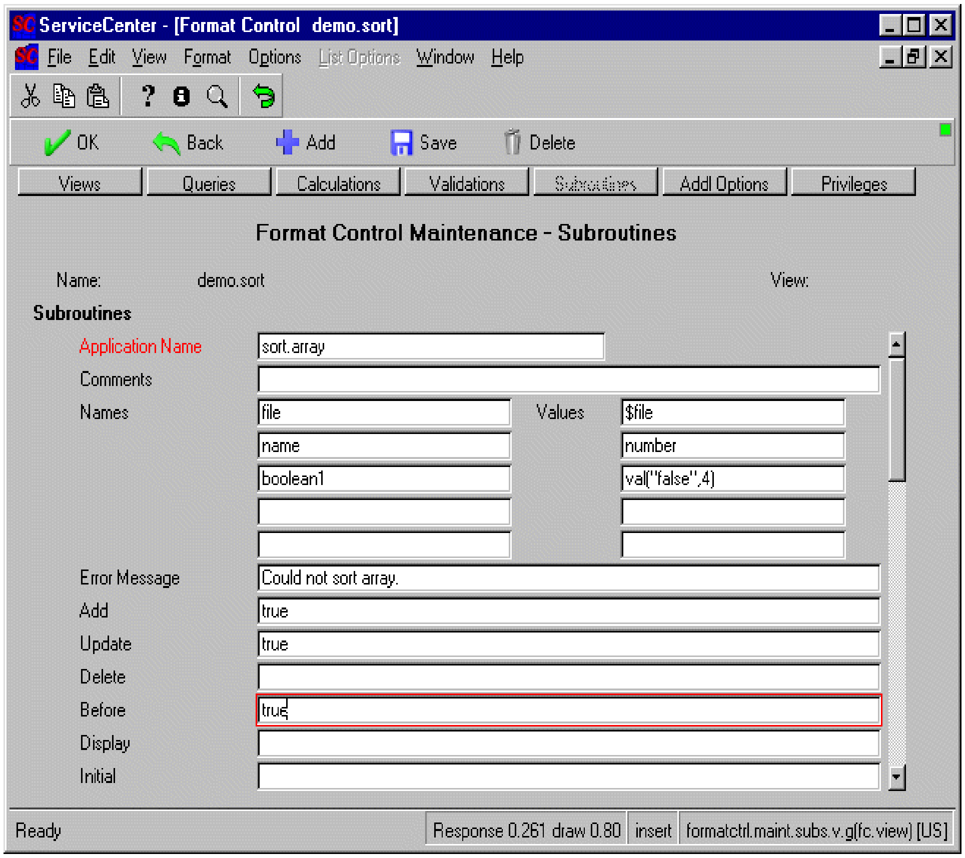
Application Name	Add	Update	Before	Error Message
sort.array	true	true	true	Could not sort the array.

Parameter values:

Names	Definition	Values
file	When calling sort.array from Format Control, always pass *$file* to this parameter.	$file
name	The name of the input field (array) in $file you wish to sort.	number
boolean1	The parameter that controls the sort order. A value of *true* sorts in ascending order. If you pass *false*, the sort order is descending. The default is *true*.	val ("false", 4) *

* For a detailed discussion of the **val** function, refer to the *System Language* guide.

7 The completed Subroutines record looks like Figure 10-4 on page 176:


ServiceCenter - [Format Control demo.sort]
File Edit View Format Options List Options Window Help
OK Back Add Save Delete
Views Queries Calculations Validations Subroutines Addl Options Privileges

Format Control Maintenance - Subroutines

Name: demo.sort View:

Subroutines

Application Name	sort.array		
Comments			
Names	file	Values	$file
	name		number
	boolean1		val("false",4)
Error Message	Could not sort array.		
Add	true		
Update	true		
Delete			
Before	true		
Display			
Initial			

Ready Response 0.261 draw 0.80 insert formatctrl.maint.subs.v.g(fc.view) [US]

Figure 10-4: Long Version of the Subroutine Form

8 Click **OK** to save your record.

Add Data to Your Test Form

1 Select the Toolkit tab in the Home menu.

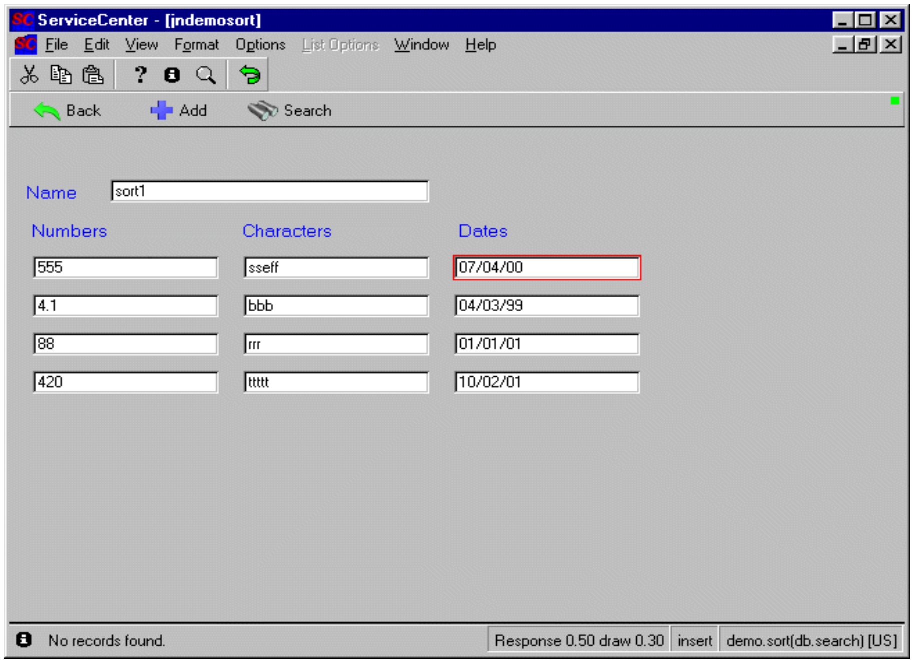
2 Click **Database Manager**.

3 Type demo.sort in the **Form** field of the Database Manager dialogue box.

4 Press **Enter** or click **Search**.

5 Type sort1 in the **Name** field of your test form.

6 Enter data in the arrays in random order (numbers in the **Numbers** array, characters in the **Characters** array, etc.)


ServiceCenter - [indemosort]
File Edit View Format Options List Options Window Help
Back Add Search

Name	sort1	
Numbers	Characters	Dates
555	sseff	07/04/00
4.1	bbb	04/03/99
88	rrr	01/01/01
420	ttttt	10/02/01

No records found. Response 0.50 draw 0.30 insert demo.sort(db.search) [US]

Figure 10-5: Test Data Record

7 Click **Add** to add the record to the database and sort the array.

Note: Since **Update** evaluates to *true* in the Format Control record for this form, you may also sort an array by clicking on the **Save** button.

The **Numbers** array is sorted in *descending* order.

8 Repeat the process for the other arrays by changing the name parameter in the Format Control record to reflect the array being sorted.

9 Remove the boolean1 parameter, and the sort order defaults to *true*. The array is sorted in *ascending* order.

11 CHAPTER Special Processing Considerations

Incident Management

In Incident Management, each action taken on an incident ticket has the option of adding an additional *page*; therefore, **update** and **delete** operations are not truly performed. Even though there is a **Save** button, the application is really *adding* a new record to the problem database. Furthermore, when you close an incident, a page can be added to the file with a status of *closed*. Consequently, when using Format Control on Incident Management, the **add** field must evaluate to *true*.

Format Control records are only executed on the *initial*, *browse*, *open*, *update*, and *close* forms for Incident Management.

Format Control could be placed on the following forms:

- problem.abends.open
- problem.abends.update
- problem.abends.close

Format Control will NOT be executed on the following forms (as they are subforms):

- problem.hdr
- problem.hdr1

- problem.open
- problem.update
- problem.close

Change Management

Change Management allows you to define three types of Format Control records:

- Detail
- Master
- Approval.

The *detail* and *master* records are executed during standard **add**, **update** and **close** processing. The *approval* record is executed during Change Management approval processing.

Change Management uses the four processing functions as follows:

Add

Processed at open time.

Update

Processed at update time (update, reopen).

Delete

Processed at close time.

Display

Processed when an item is displayed.

The Format Control components are executed *after* the process is invoked but *before* the record is permanently updated. For instance, the **add** function is executed *after* the user clicks on the **Open** button in either the Request or Task structure, but *before* the Request/Task is added to the database. The **display** function is executed *after* a record has been selected from the QBE list but *before* the record is actually displayed.

You can execute both a *master* Format Control record and a *detail* Format Control record for each Request/Task process. The *master* Format Control record executes before the *detail* Format Control Record. If any of the Format Control components fail, for any reason, the user is always returned to the previously displayed screen with the appropriate error messages.

Master Format Control Record

Change Management supports a *master* Format Control record which allows you to define in one record the Format Control statements that apply to all Change Management Request Phases. The name of the master Format Control record is *cm3r* (*cm3t* for Tasks). The options on this Format Control record are executed during all Request/Task processing except *approval* and background processing. The master format control record is processed *before* the *detail* Format Control record.

Detail Format Control Record

To enforce processing rules that are unique to a Phase, define a format control record that is the same name as the *default* View of that Phase. The **add, update, delete,** and **display** features are processed just as they are with *detail* Format Control definitions.

Approval Format Control Record

The Change Management approval process checks for a Format Control record that is associated with the *approvals* view of the Phase.

- The **add** features are processed when a Request is *approved*.
- The **update** features are processed when a Request is *disapproved*.
- The **delete** features are processed when a Request is *unapproved*.

The *master* Format Control record is not executed during approval processing.

Format Control and Eventout Records

When incidents are opened, updated or closed by Event Services, a record may be written to the *eventout* file. This record contains information from the incident (described in the output **eventmap** record for the event) that will be passed to an external process via the SCAuto/IPAS external interface. You can elect to write to the **eventout** file when Help Desk operators open and close tickets so that the information is passed to the external interface.

The **axces.write** application creates a character string of fields from a structure and writes them to **eventout**. An Event Registration record identifies the event type and the name of the Event Map records used to define which fields will be selected from the record. The application should be called as a Format Control Subroutine passing two parameters:

- The record from which data will be mapped.
- The Event Type, as defined in the Event Register.

To write to **eventout** on incident close, the Format Control record would be attached to the **problem.equipment.close** form using the following values:

Field	Value
Application	axces.write
Add	true

Name	Value
record	$file
name	pmc

Note: This procedure is not specific to Incident Management. You can write **eventout** records for other applications, such as Inventory Control or Change Management.

The Subroutine form for the **problem.equipment.open** form looks like this::

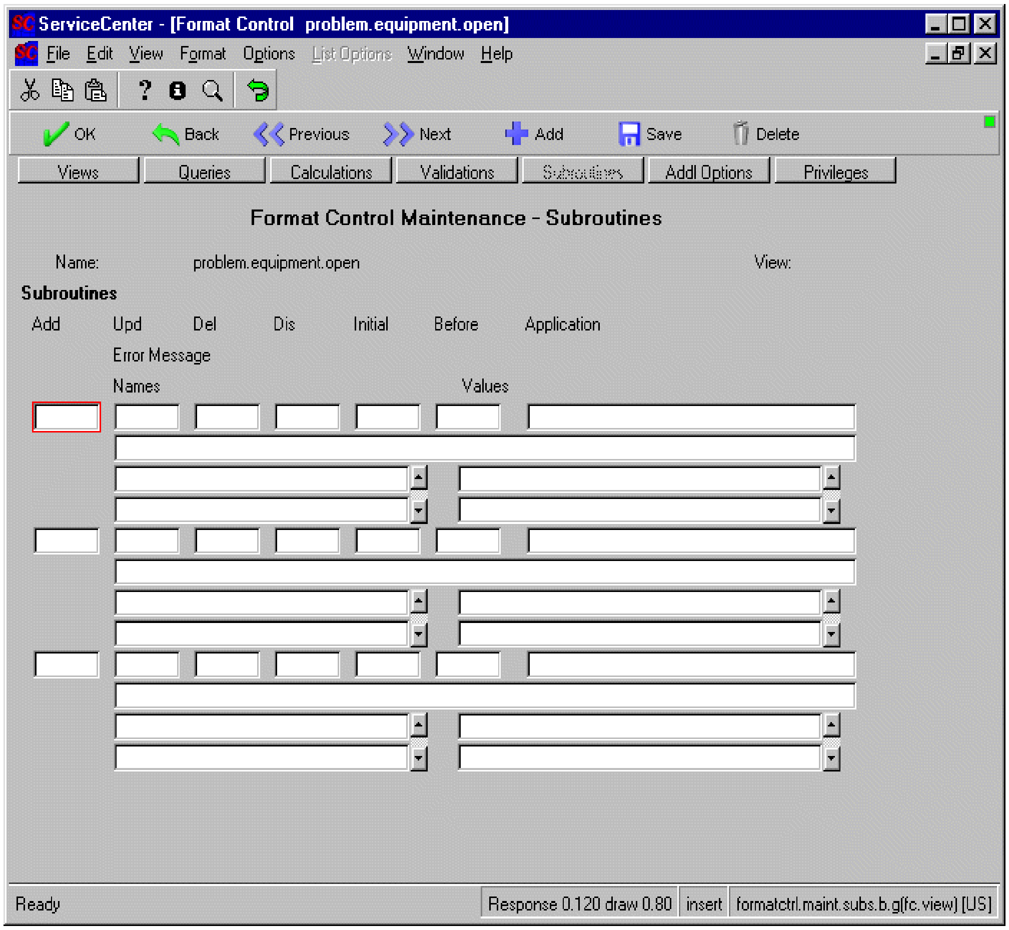


Figure 11-1: Subroutine for problem.equipment.open

Page Messages

SCAuto supports a generic **page** function. Page events can be written to the **eventout** file via a Subroutine call to **scauto.page** from Format Control. The parameters to pass are:

Name	Value
name:	**contact.name** in **contacts** file or **name** in **operator** file
prompt:	numeric message
text:	alphanumeric message
string1:	separation character
query:	user sequence number
values:	list of contact.name or name values
names, 1:	a pager phone number (ignored if value is passed in *name* or *values*
names, 2:	pager PIN number
names, 3:	the name of a record in the *distgroup* file that contains a list of **names** or **contact.names** (optional–SP4 or >)

Incident Management Alert Records

How can you determine what Format Control record is running within Incident Management at any given time? The normal rules of Format Control apply during Text and GUI mode operations. For example, when you update a form named **problem.abends.update.g**, Incident Management looks for a Format Control record called **problem.abends.update**.

However, when background alerts are processed, Incident Management uses a different set of Format Control records named **problem.<*category*>.alerts**, where <*category*> is a defined Incident Management category. For example, for an incident ticket of an *abends* category, Incident Management will look for a Format Control record called **problem.abends.alerts**. A ticket in the *TCP* category would access a Format Control record called **problem.TCP.alerts**.

Important: Alerts are processed in the background and cannot ask the user for information. Any **Subroutines** which require screen input/output should be avoided in *alert* Format Control operations, as should **Validations** requiring user input.

12 CHAPTER

Sequential Numbering for Format Control

Sequential numbering is a Subroutine process of Format Control that automatically adds identifying numbers to database records as they are created. Sequential numbering can be used for inventory control, incident tickets, or employee records, for example.

Sequential numbers can be defined for appropriate fields in any data record. You may create alphanumeric strings (prefixes and suffixes) that identify records by task or item.

Sequential numbering allows you to:

- Increment/decrement numbers in sequence.
- Reset values at definable intervals.
- Define values for increment/decrement sequence.
- Define the length of the number.
- Append *prefixes* and *suffixes* (characters) to sequential numbers.

The **getnumb.fc** application is called from the Subroutines process of Format Control to establish the sequential numbering sequence.

Data Types

There are two data types of sequential numbers:

- Simple **numbers** (1, 2, 3, etc.).
- Complex **character** strings (DEV00001/WS).

Note: If you wish to use the **prefix** and **suffix** capabilities of sequential numbering, the field being incremented or decremented must be defined in the Database Dictionary as type **character.** In all other cases, the field must be of type **number**.

Parameters

The following parameters are used for passing sequential numbering values to the **getnumb.fc** application in the Subroutines process:

record

The data record in which the number will be placed. The value of this parameter must be *$file*.

name

The sequential number Class (category) of the record to which the Format Control record is attached. This value appears in the **Class** field of the Sequential Number File.

prompt

The field in the data record in which the sequential number will appear.

string1

(Optional). Defines the prefix to use. This is the identifying character string that appears before the sequential number.

query

(Optional). Defines the suffix to use. This is the identifying character string that appears after the sequential number.

text

(Optional). Defines the data type of the sequential number (**number** or **string**). The default is **number**.

number1

(Optional). Defines the length of the sequential number. The subroutine will add as many zeroes to the number as necessary to match the required number of digits. Pass the RAD function **val**() to the number1 parameter. For example, a four digit number would have the value **val**("4", **1**) where **4** is the length of the number and **1** is the data type (number).

numbers,1

(Optional). Determines the reset point for the sequential number. When the value passed to the application is reached, the sequential number is reset to its starting value.

numbers,2

(Optional). Establishes a new starting value for the sequential number when the reset point has been reached.

numbers,3

(Optional). Sets the increment/decrement value. For example, you may want to increment by 2, displaying only odd or even numbers.

numbers,4

(Optional). Allows the user to increment or decrement the sequential number independent of the *number* file. Use this parameter to attach a special number to your records without updating the *number* file.

Simple Sequential Numbers

This example illustrates the process of setting up sequential numbering using Format Control.

In the process, we will:

- Create a form.
- Create a database file.
- Create a Format Control record for the form.
- Open the number file.
- Add a numbered record to the database file.

Create the Form

Create a form in Forms Designer called **employees** that records simple employee data such as name, address, and telephone number. Be sure to include a field labeled **Employee ID** with an input value of **employee.id**.

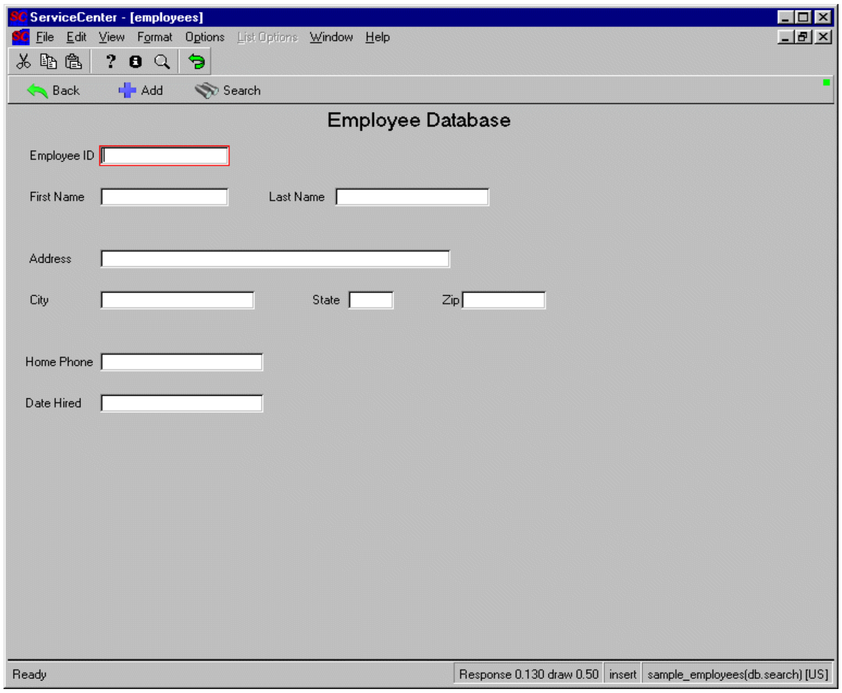


Figure 12-1: Blank Employee Record

Create a Database File

1 After you have finished creating your form, select **Create File** from the Options menu in Forms Designer.

2 Enter **employees** as a file name in the dialogue box.

3 Click **OK** to create the new datafile.

4 Change the **Employee ID** field of the *employee* file to a **number** data type.

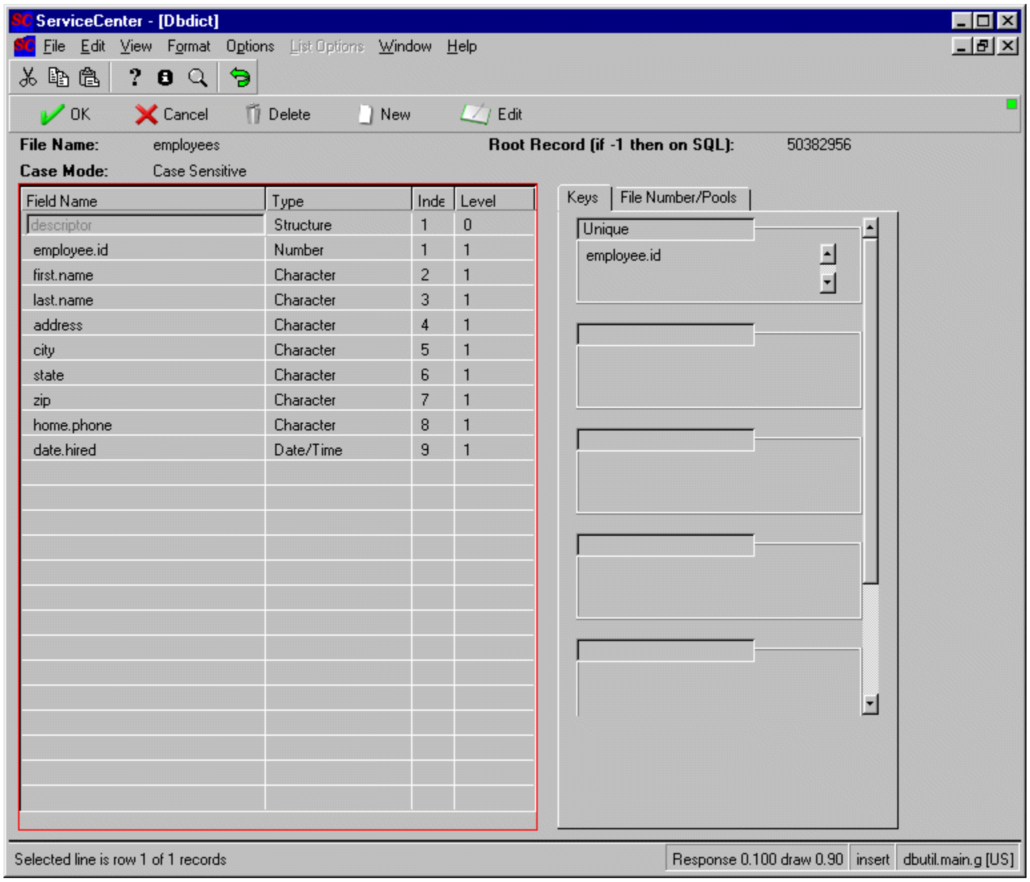
  a Select the **employee.id** field with the cursor.

  b Click **Edit**.

  c Select **number** from the drop–down menu in the **Type** field.

Important: A data type of **number** is required for all simple sequential numbering.

5 The edited file appears as shown in Figure 12-2 on page 191:



File Name: employees

Case Mode: Case Sensitive

Root Record (if -1 then on SQL): 50382956

Field Name	Type	Inde	Level
descriptor	Structure	1	0
employee.id	Number	1	1
first.name	Character	2	1
last.name	Character	3	1
address	Character	4	1
city	Character	5	1
state	Character	6	1
zip	Character	7	1
home.phone	Character	8	1
date.hired	Date/Time	9	1

Unique: employee.id

Selected line is row 1 of 1 records

Response 0.100 draw 0.90 | insert | dbutil.main.g [US]

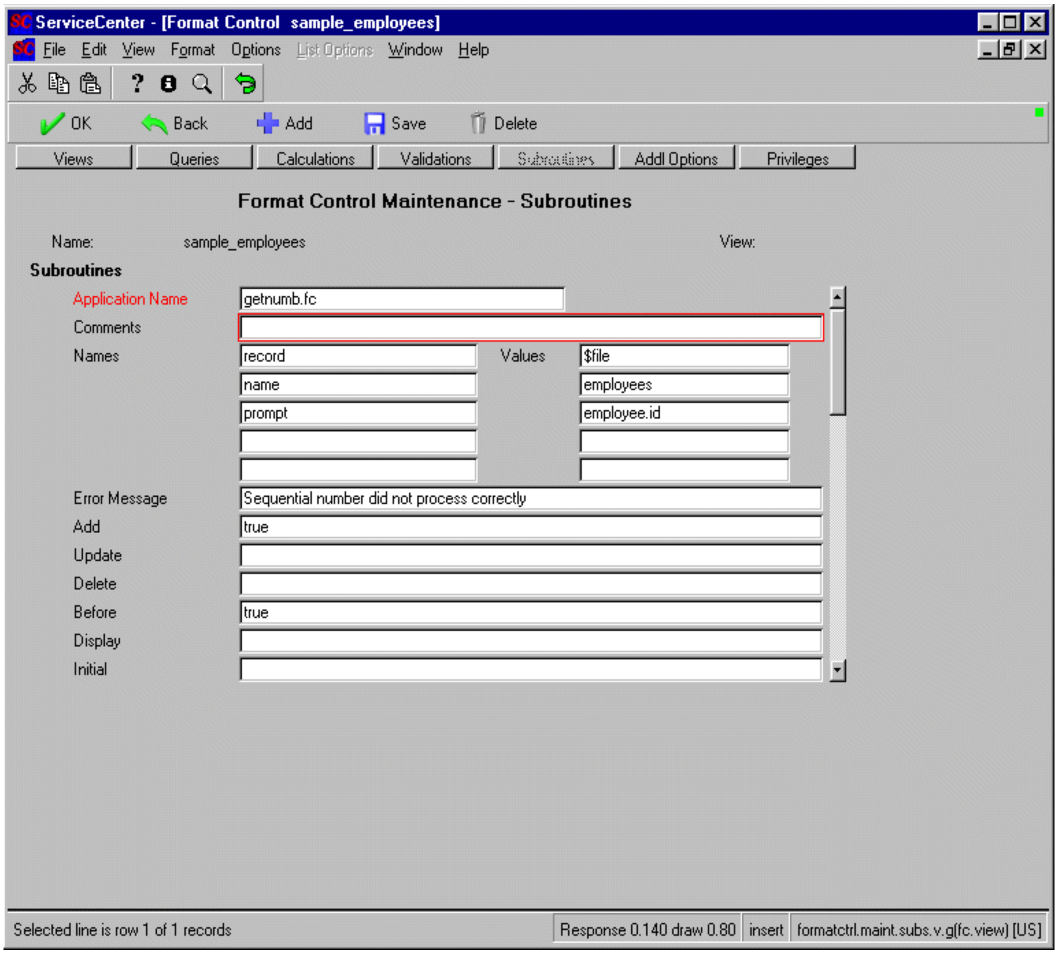
Figure 12-2: Create a File

Create a Format Control Record

1. Click the **Tools** button in the Utilities tab in the administrator's home menu.
2. Click **Format Control** in the Tools menu.
3. Type **employees** in the Name field and click **New**.
4. Select **Subroutines** from the Options menu or click the Subroutines button.
5. Select **Show Expanded Form** from the Options menu in Subroutines.
6. Use the following values to complete the Subroutines process:

Application Name	Add	Before	Error Message
getnumb.fc	true	true	Sequential number did not process correctly!

Names	Definition	Values
record	The record in which the number will be placed. In this example, the employees datafile.	$file
name	The name to appear in the **Class** field of the Number file generated by the Format Control record.	employees
prompt	The field within the record receiving the sequential number.	employee.id


ServiceCenter - [Format Control sample_employees]
File Edit View Format Options List Options Window Help
OK Back Add Save Delete
Views Queries Calculations Validations Subroutines Addl Options Privileges

Format Control Maintenance - Subroutines

Name:	sample_employees	View:	

Subroutines

Application Name	getnumb.fc		
Comments			
Names	record	Values	$file
	name		employees
	prompt		employee.id
Error Message	Sequential number did not process correctly		
Add	true		
Update			
Delete			
Before	true		
Display			
Initial			

Selected line is row 1 of 1 records | Response 0.140 draw 0.80 | insert | formatctrl.maint.subs.v.g(fc.view) [US]

Figure 12-3: The Long Version of the Subroutine Form

Add a Record to the Database

1. Select the Toolkit tab in the Home menu.
2. Click the **Database Manager** button in the Home menu.
3. Enter employees in the **Form** field of the Database Manager dialogue box.
4. Press **Enter** or click **Search**.
5. Add data to the blank Employee Database record, but leave the Employee ID field blank.
6. Click **Add**. The sequential number is added to the **Employee ID** field before the record is added to the database.

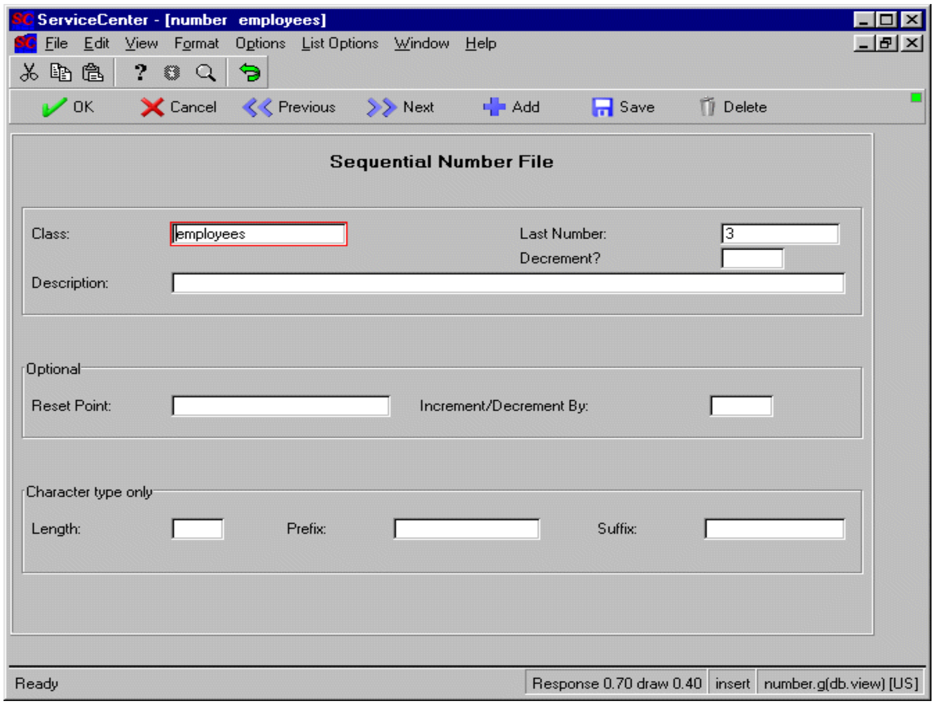
Number File

A Sequential Number File record is created by the Subroutines process of Format Control when the application **getnumb.fc** is called. Values in the number file fields are defined by the parameters passed to the application from within Subroutines. These values determine how the sequential number is displayed and what other information is contained within it. You can define number file values either directly in the number file or via parameters passed from Format Control. However, if you use the number file to change values *also* defined in the Format Control Subroutines process, they will be overridden by the Format Control values.

To view the Number file:

1. Select the Toolkit tab in the Home menu.
2. Click on the **Database Manager** button in the Home menu.
3. Enter number in the **Form** field of the Database Manager dialogue box.
4. Press **Enter** or click on the **Search** button.
5. When the blank Sequential Number File form appears, press **Enter** or click on the **Search** button to display a QBE list.
6. Select **employees** from the list of records.

7 The Sequential Number File for the *employees* file is displayed.


ServiceCenter - [number employees]
File Edit View Format Options List Options Window Help
OK
Cancel
Previous
Next
Add
Save
Delete

Sequential Number File

Class: employees
Last Number: 3
Decrement?
Description:

Optional
Reset Point:
Increment/Decrement By:

Character type only
Length:
Prefix:
Suffix:

Ready | Response 0.70 draw 0.40 | insert | number.g(db.view) [US]

Figure 12-4: Number File for *employees* Datafile

Fields

Class

The class or category of data records to which the Format Control record is attached.

Last Number

Displays the number of the last record in the defined sequence.

Decrement?

Boolean field determining whether or not the numbering sequence is decremented. If this field evaluates to *true*, the sequence will be decremented.

Description

Plain text description of the file to which the Format Control record is attached. This field is for reference only.

Reset Point

Defines the point (number) at which the sequential number will return to its starting value.

Increment/Decrement By

Determines the numbering sequence. For example, you may want to increment a sequence by two, producing odd numbers only. The default is 1.

Length

Determines the length (number of digits) of the number. For example, a sequential number with a starting value of 1 and a length value of 5 would be displayed as 00001.

Prefix

Prefix to appear before the sequential number. For example, you may want to identify a device record with the prefix DEV.

Suffix

Suffix appended to the end of the sequential number. For example, the sequential number identifying a workstation record might be followed by /WS.

Prefixes and Suffixes

Prefixes and suffixes are used to embed information in the sequential number and provide additional levels of identification. For example, each device in the database might be described by a logical name in the form of **DEV**(*number*)/(*device type*). For workstations the device type might be **WS**. In this case, DEV is the *prefix* for the sequential number and /WS is the *suffix*.

In this example, we will add a subroutine call to the **device.workstation** form to assign a unique device number to all records added to the file from this form.

In the process, we will:

- Create a Format Control record for the **device.workstation** form.
- Add a record to the **device.workstation** file.

Create a Format Control Record

1 Click the **Tools** button in the Utilities tab.

2 Click **Format Control**.

3 Enter device.workstation in the **Name** field of the blank Format Control Initializations form.

4 Press **Enter** or click **Search**.

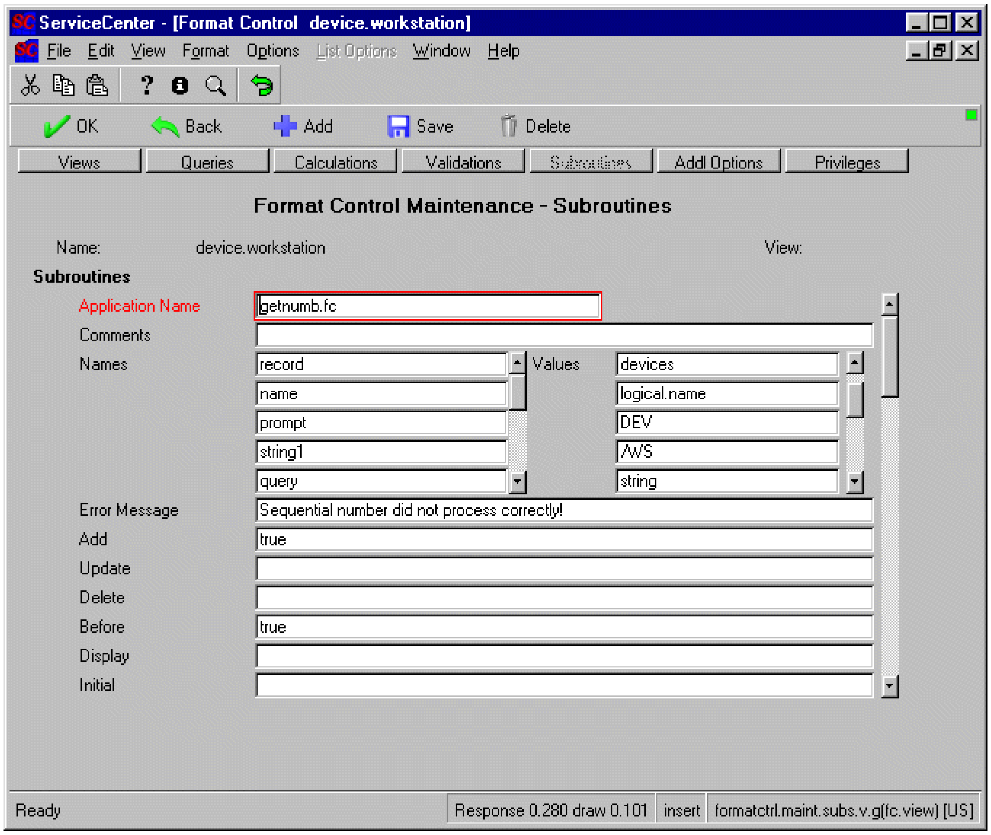
5 Select **Subroutines** from the Options menu or click the Subroutines button.

6 Use the following values to complete the Subroutines process:)

Application Name	Add	Before	Error Message
getnumb.fc	true	true	Sequential number did not process correctly!

Names	Definition	Values
record	The record in which the number will be placed. In this example, the device.workstation record.	$file
name	The sequential number class to be used.	devices
prompt	The field within the record receiving the sequential number.	logical.name
string1	The string (prefix) to be added to the front of the sequential number when creating a new number.	DEV
query	The string (suffix) to be added to the end of the sequential number when creating a new record.	/WS
text	Defines the sequential number type (number or string). Since characters occur in the number, this must be **string**.	string
number1	Determines the length of the sequential number.	val("4", 1)

The Format Control record appears as follows:


ServiceCenter - [Format Control device.workstation]
File Edit View Format Options List Options Window Help
OK Back Add Save Delete
Views Queries Calculations Validations Subroutines Addl Options Privileges

Format Control Maintenance - Subroutines

Name:	device.workstation
View:	

Subroutines

Application Name	getnumb.fc		
Comments			
Names	record	Values	devices
	name		logical.name
	prompt		DEV
	string1		/WS
	query		string
Error Message	Sequential number did not process correctly!		
Add	true		
Update			
Delete			
Before	true		
Display			
Initial			

Ready | Response 0.280 draw 0.101 | insert | formatctrl.maint.subs.v.g(fc.view) [US]

Figure 12-5: Completed Subroutine Form

Add a Data Record

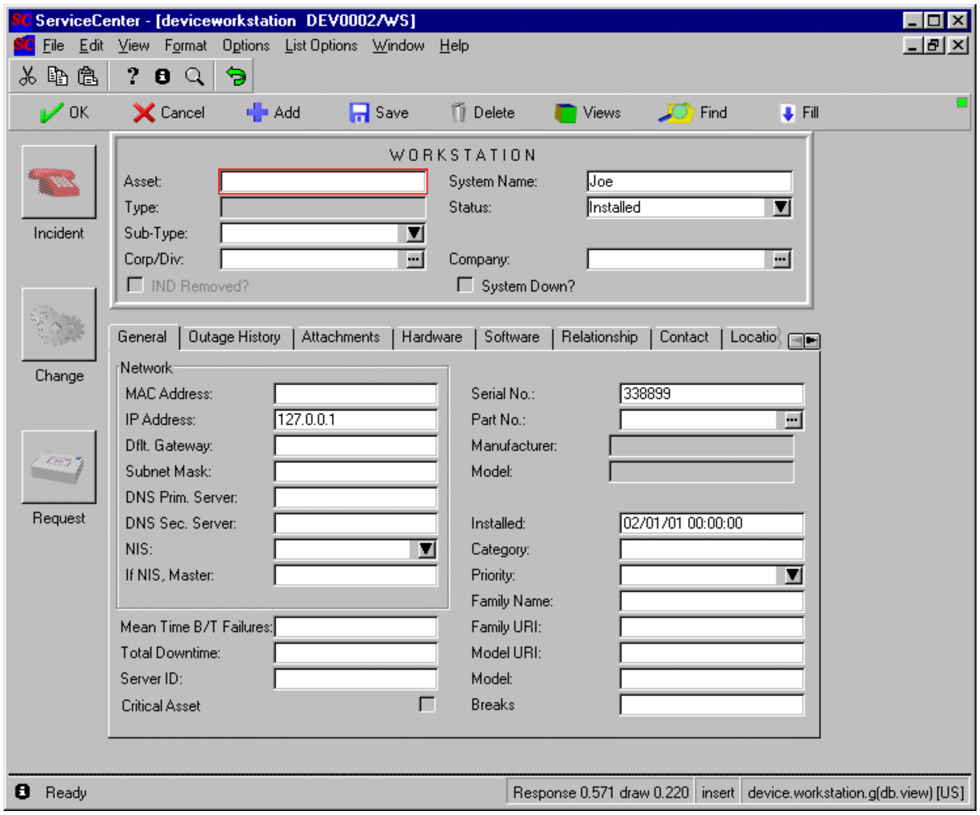
1 Select the Toolkit tab in the System Administrator's Home menu.

2 Click **Database Manager**.

3 Enter **device.workstation** in the **Form** field of the Database Manager dialogue box.

4 Press **Enter** or click on the **Search** button.

5 Add data to the blank Workstation record.


ServiceCenter - [deviceworkstation DEV0002/WS]
File Edit View Format Options List Options Window Help
OK Cancel Add Save Delete Views Find Fill
Incident
Change
Request
General Outage History Attachments Hardware Software Relationship Contact Locatio

WORKSTATION

Asset:		System Name:	Joe
Type:		Status:	Installed
Sub-Type:			
Corp/Div:		Company:	
IND Removed?		System Down?	

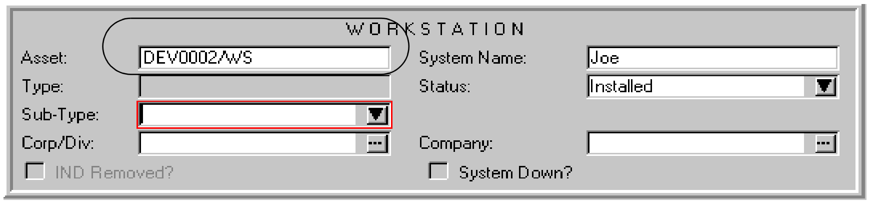
Network

MAC Address:		Serial No.:	338899
IP Address:	127.0.0.1	Part No.:	
Dflt. Gateway:		Manufacturer:	
Subnet Mask:		Model:	
DNS Prim. Server:			
DNS Sec. Server:		Installed:	02/01/01 00:00:00
NIS:		Category:	
If NIS, Master:		Priority:	
		Family Name:	
Mean Time B/T Failures:		Family URI:	
Total Downtime:		Model URI:	
Server ID:		Model:	
Critical Asset		Breaks	

Ready   Response 0.571 draw 0.220   insert   device.workstation.g(db.view) [US]

Figure 12-6: Workstation Record

6 Click **Add.** A complex sequential number is added to the **Asset** field before the record is added to the database.



WORKSTATION

Asset:	DEV0002/WS	System Name:	Joe
Type:		Status:	Installed
Sub-Type:			
Corp/Div:		Company:	
IND Removed?		System Down?	

Notice that the complex sequential number DEV0002/WS appears in the **Asset** field (**logical.name**). Add another record, and the sequential number will advance to DEV0003/WS.

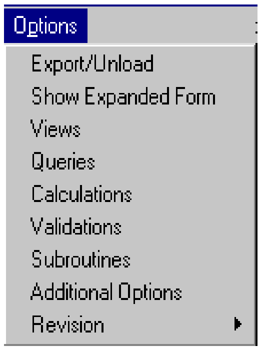
13 CHAPTER Format Control Processes

When Format Control is called, it evaluates user input from seven *processes* to modify the way data is delivered to the database. These processes are evaluated in the following order.

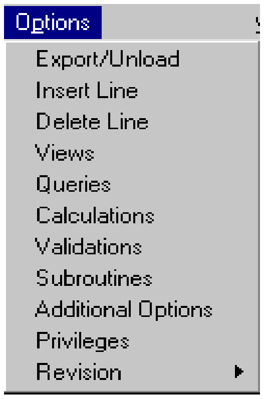
- Views
- Queries
- Calculations
- Validations
- Subroutines
- Additional Options
- Privileges

Note: The order in which the processes are presented in this chapter reflects the frequency with which each process is used and not the processing order.

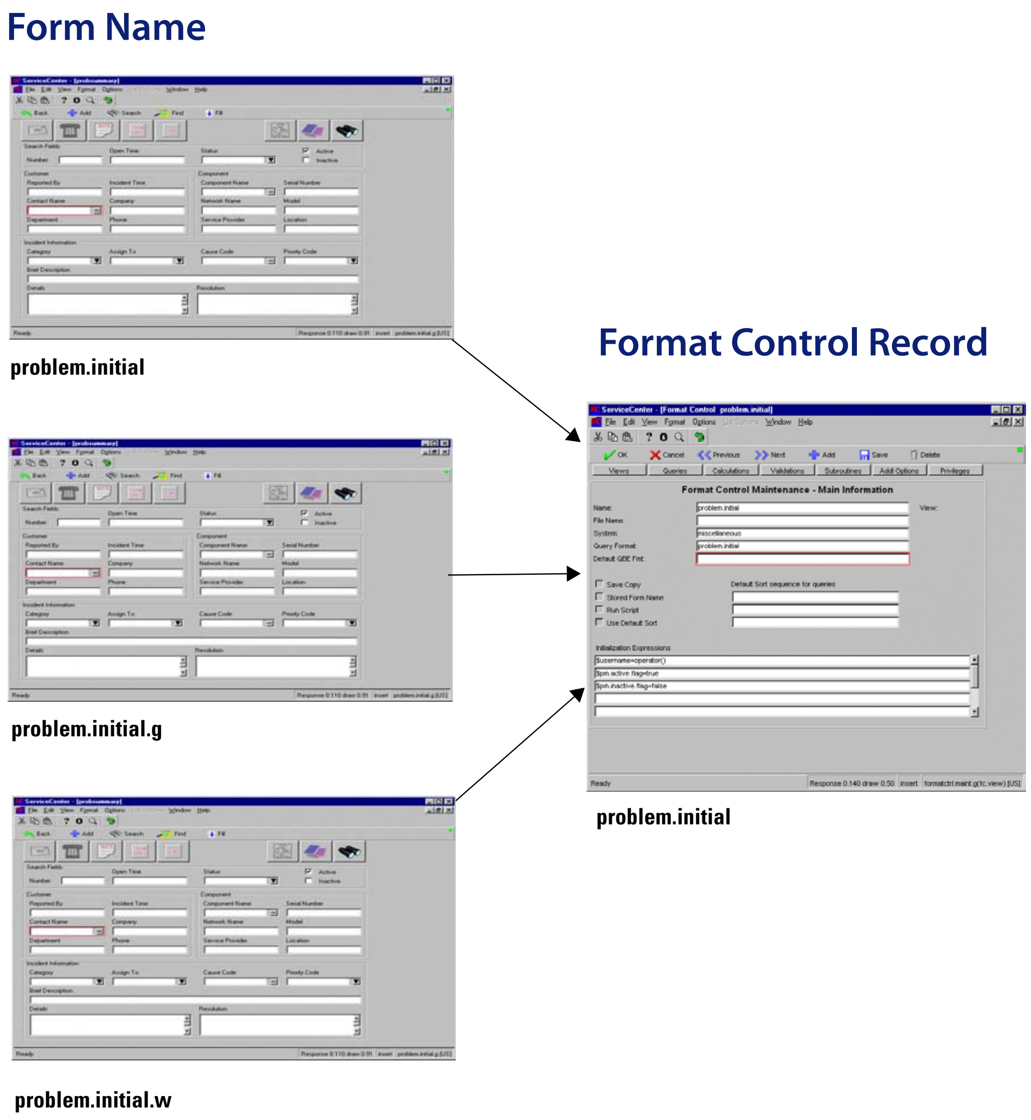
Format Control processes are accessed through the Options menu from anywhere within a Format Control record or through the corresponding buttons. See Menu 1 below. As you move between processes, the Options menu changes to reflect the available choices. For processes containing arrays, editing tools (**Insert Line**, **Delete Line**) are included in the menu. See Menu 2 below.



Menu 1



Menu 2



Format Control Maintenance - Main Information

Name: problem.initial
File Name:
System: miscellaneous
Query Format: problem.initial
Default QBE Fmt:
View:

☐ Save Copy
☐ Stored Form Name
☐ Run Script
☐ Use Default Sort

Default Sort sequence for queries

Initialization Expressions

```
$username=operator()
$pm.active.flag=true
$pm.inactive.flag=false
```

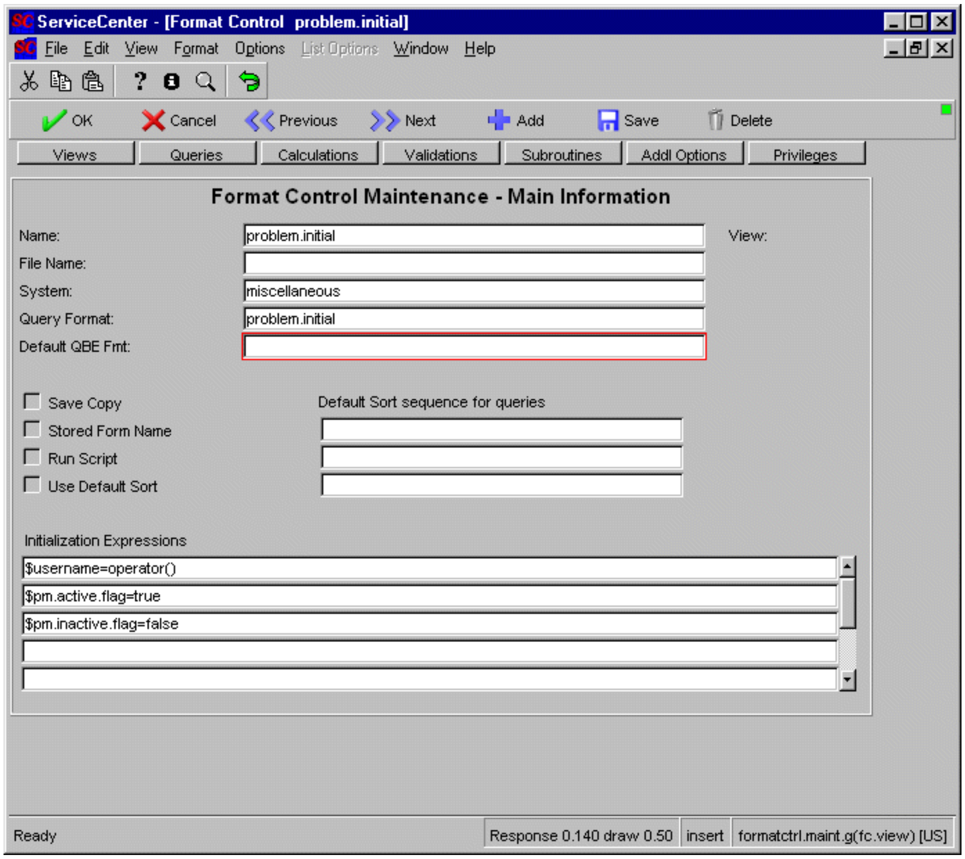
Ready | Response 0.140 draw 0.50 | insert | formatctrl.maint.g(fc.view) [US]

Figure 13-1: Naming Process for Format Control Records

Main Information

The first form of a Format Control record is the Main Information form or Initializations process.

The Initializations process of the Format Control record is used to store general information and to initialize variables to be used in other processes of the Format Control record. Initialization expressions are the first operation performed for each evaluation of a Format Control record.


ServiceCenter - [Format Control problem.initial]
File Edit View Format Options List Options Window Help
OK Cancel Previous Next Add Save Delete
Views Queries Calculations Validations Subroutines Addl Options Privileges

Format Control Maintenance - Main Information

Name: problem.initial
File Name:
System: miscellaneous
Query Format: problem.initial
Default QBE Fmt:
View:

☐ Save Copy
☐ Stored Form Name
☐ Run Script
☐ Use Default Sort

Default Sort sequence for queries

Initialization Expressions

```
$username=operator()
$pm.active.flag=true
$pm.inactive.flag=false
```

Ready | Response 0.140 draw 0.50 | insert | formatctrl.maint.g(fc.view) [US]

Figure 13-2: Format Control Main Information Form

Fields

Name

Name of the Format Control record. Format Control uses the base name of the displayed form to locate the Format Control record. The base name has the GUI (**.g**) or Web (**.w**) suffix removed; therefore, the same Format Control record is used for Text, GUI, and Web users File Name

Do not use this field. Values in this field differ in meaning between applications, and unexpected results could occur.

System

Free form text field that allows you to group your Format Control records by functionality. You may query by System name to display a QBE list of records in that group. Select your own unique name or use the system default, **miscellaneous.**

Query Format

This optional field is used to name an abbreviated form designed to limit the scope of queries against large database files by reducing the number of fields available for user queries. Query forms typically contain only keyed fields for the file, and appear only in preparation for querying.

Note: Limiting users' query fields only to keyed fields reduces system load and speeds the querying process.

For example, if you have a large database file with only three keys defined in its database dictionary, you should limit your users' ability to query non–keyed fields. When a record is selected using this query, the normal form (the form to which the Format Control record was attached) is displayed with all its fields.

Default QBE Fmt

Enter the name of the QBE form to use when displaying record lists, i.e., lists of records selected as a result of a query. You can create different QBE forms for different users and establish a default for each group.

Note: To create a standard, default QBE form to be used for all forms and users that access a particular file, create a form called (filename*).qbe.* For example, *contacts.qbe*.

This feature allows you to name different QBE forms for querying the same file. Create different forms based on user capabilities and attach them by name to the appropriate Format Control Record.

For example, you could use the Forms Designer to create a custom QBE form for the **contacts** file using the following values:

Form name: **contacts.view**

Column Captions	Input Values
Contact Name	contact.name
First Name	first.name
Company	company
Phone	phone
Email	email

Note: The Format Control Views process allows you to define many different QBE forms for use with a single data form. For specific information on this process, refer to *Views* on page 258.

Save Copy

If this option is selected, a copy or image of the primary file's record will be saved before the record is displayed. The "before" copy is referenced as *$file0, e.g., name in $file0* to reference the original value of the name field, before any changes resulting from user input.

For example, if you want to be able to check whether a user changed a field, select this feature. Then, you can create calculations to set a flag, based on whether or not the value changed, for example:

```
if (originator in $file=originator in $file0) then $nochange=true else
$nochange=false
```

Based on the results of this calculation, other Format Control activities can occur or not occur.

Stored Form Name

If this option is selected, the Database Manager application will store the name of the form used to add each record to a file in a field called **format**. This **format** field must exist in the file's Database Dictionary record.

For example, Incident Management and Change Management store the form name in each record so the system knows which form to use when displaying records. Use Format Control to do the same thing in other database files.

Run Script

Note: The Run Script functionality is designed to work with the Database Manager only and not within applications like Incident Management, Service Management, Call Management, Request Management or Root Cause Analysis. There are facilities within each of those applications for script execution.

If this option is selected, the system will execute the script identified in the Initialization Expressions as **$script.name**. For example, to collect information in a form before it is displayed for querying the database, you can build and execute a script called *datacollection* to prompt for information. The name of the script would be entered in the Initializations Expressions as:

```
$script.name="datacollection"
```

Refer to the *Scripting* chapter in System Tailoring Volume 3 for information about building scripts.

Use Default Sort

If this option is selected, the specified Default Sort Sequence for queries will *always* be used.

Default Sort Sequence for Queries

The value in this field determines which key will be used to select data from *all* records. This sort sequence overrides the system's decision about the most efficient key to use for queries. If the sort sequence is not a key in the dbdict for the file, you receive the messages: *Records cannot be selected from the* (filename)*file* and *Sort order must be a key*.

Warning: Improper key selection can result in an inefficient query, which can adversely affect system performance.

Initialization Expressions

Set initial values for variables and fields in this array. These expressions are used in subsequent calculations or to display values in forms upon initial display. They are evaluated before any other Format Control processes are evaluated against the current record. Initialization expressions are evaluated each time the form is used, either to view a record in Database Manager or to write a record to a report.

The following table demonstrates how to initialize variables to different data types:

Data Type	Syntax
character	$x=" "
number	$number=0
date/time	$dt= '00:00:00'
boolean (true/false)	$end=false or $end=f
arrays	$array={}

Note: You can set the value of variables to NULL after establishing their types.

Examples:

- If you want to total a variable dependent upon a value in a field in the Calculations process of a Format Control record, use Initialization Expressions to set the variables as follows:

  ```
  $open=0;$update=0;$close=0
  ```

  In the Calculations process of Format Control, add the following calculations:

  ```
  if (open.time in $file>=tod() - '7 00:00') then ($open+=1);
  if (update.time in $file>=tod() - '7 00:00') then ($update+=1);
  if (close.time in $file>=tod() - '7 00:00') then ($close+=1)
  ```

This could be used in a report to calculate the number of problem tickets opened, updated, and closed within the last week.

If you do not set these variables to `0` in the **Initialization Expressions** field, the system will not know these variables are numeric. In our example, the system will reset the variables to zero for each record selected before executing the other Format Control operations.

- If the **manufacturer** field in an inventory file is usually **ibm**, use Format Control to initialize a default value of **ibm**. Either leave the default value or replace it with a different manufacturer as appropriate.

  The syntax would be as follows:

  ```
  if manufacturer in $file=NULL then manufacturer in $file="ibm"
  ```

  or

  ```
  manufacturer in $file=nullsub(manufacturer in $file, "ibm")
  ```

- If a variable is to be initialized *only once* during evaluation of a ServiceCenter Report Writer report, the operation should be performed in a Format Control record connected to the **header** form. If the variable is set in the **detail** form, it will be reset each time a record is selected.

  To initialize a field or variable only once for a Report Writer report, put Initialization Expressions in a Format Control record attached to the **header** form:

  ```
  $total=nullsub($total,0)
  $total now contains a value of 0 if it was previously NULL
  ```

  To initialize a field or variable for a Report Writer report each time a record is read, put statements in a Format Control record attached to the **detail** form. Use the following statement:

  ```
  $finished=false;$severity1=0;$name="ibm";$date='00:00:00'
  ```

Modifying a Format Control Record

Move through a Format Control record by selecting the various processes (Views, Calculations, Additional Queries, etc.) from the Options menu or by clicking the buttons. Add or delete data in any form of the record.

You may exit the record at any time:

Click **Back**. A prompt box is displayed asking you if you want to save any changes you have made.

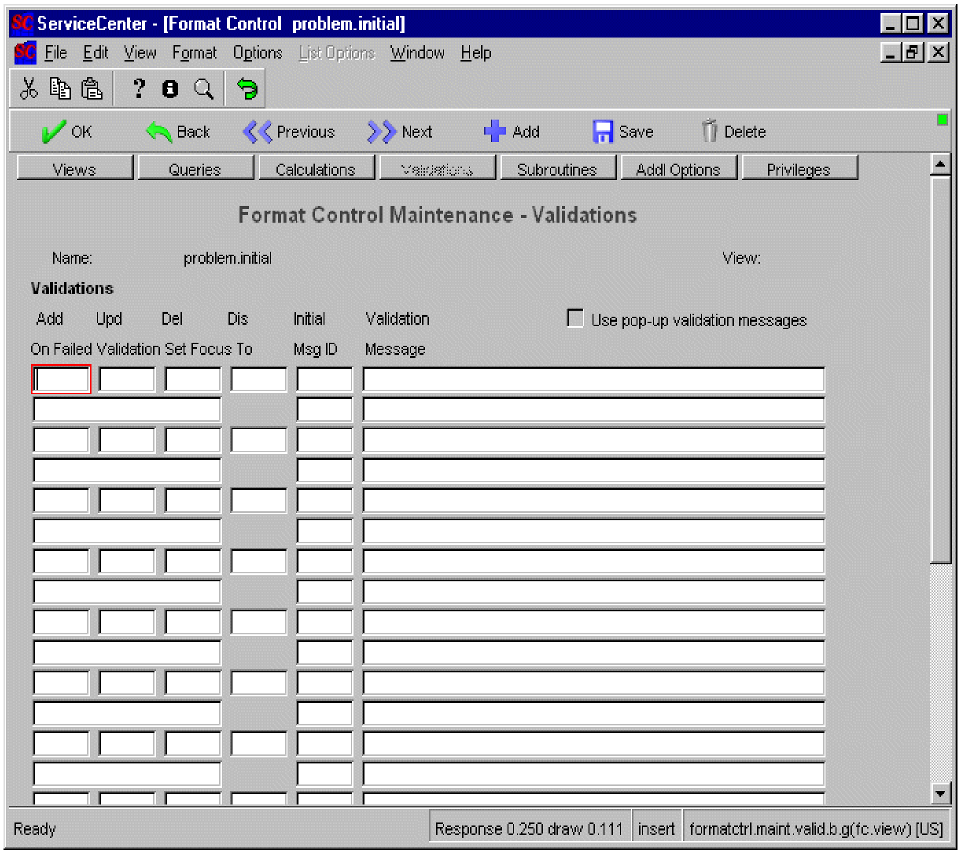
- Click on **OK** to save your changes and return to a blank Initializations form.
- Click **No** to return to the Initializations form without saving the changes to the record.
- Click **Cancel** to return to the current form without taking any action.
- Click **Save** to save the current record.

Validations

The Validations process of the Format Control Record is used for forcing a field to be required (not NULL), to be a specific value, or to be within a range of values during data entry. This is a commonly used section of Format Control because it is a simple way of field editing.

Note: **Validations** should not be confused with *Validity Tables*, a different ServiceCenter utility with more flexibility and better suited to complex validations. The Format Control **Validations** process uses expressions in a Format Control record to establish field validity and is well suited to single field validations.

Select **Validations** from the Options menu to display the Validations form.


ServiceCenter - [Format Control problem.initial]
File Edit View Format Options List Options Window Help
OK Back Previous Next Add Save Delete
Views Queries Calculations Validations Subroutines Addl Options Privileges

Format Control Maintenance - Validations

Name: problem.initial

View:

Validations

Add Upd Del Dis Initial Validation

Use pop-up validation messages

On Failed Validation Set Focus To Msg ID Message

Ready Response 0.250 draw 0.111 insert formatctrl.maint.valid.b.g(fc.view) [US]

Figure 13-3: Validations—Short Form

Fields

Add, Update, Delete, Display, Initial

These are Boolean fields that control Format Control processing. Processes evaluating to *true* apply Format Control before completing the specified operation. For example, when a user clicks **Save**, and before the record updates, Format Control executes every section or expression where the Update field evaluates to *true*.

Validation

An expression that evaluates to True or False. If this expression evaluates to True, then the validation is successful and processing continues. If this expression evaluates to False, then the validation fails and the validation message appears on the screen.

On Failed Validation Set Focus To

The field where the cursor should be positioned. When a Validation fails, ServiceCenter displays the input form again, with the error message displayed at the bottom and the cursor positioned in the field specified here (typically the field that just failed its Validation).

Message

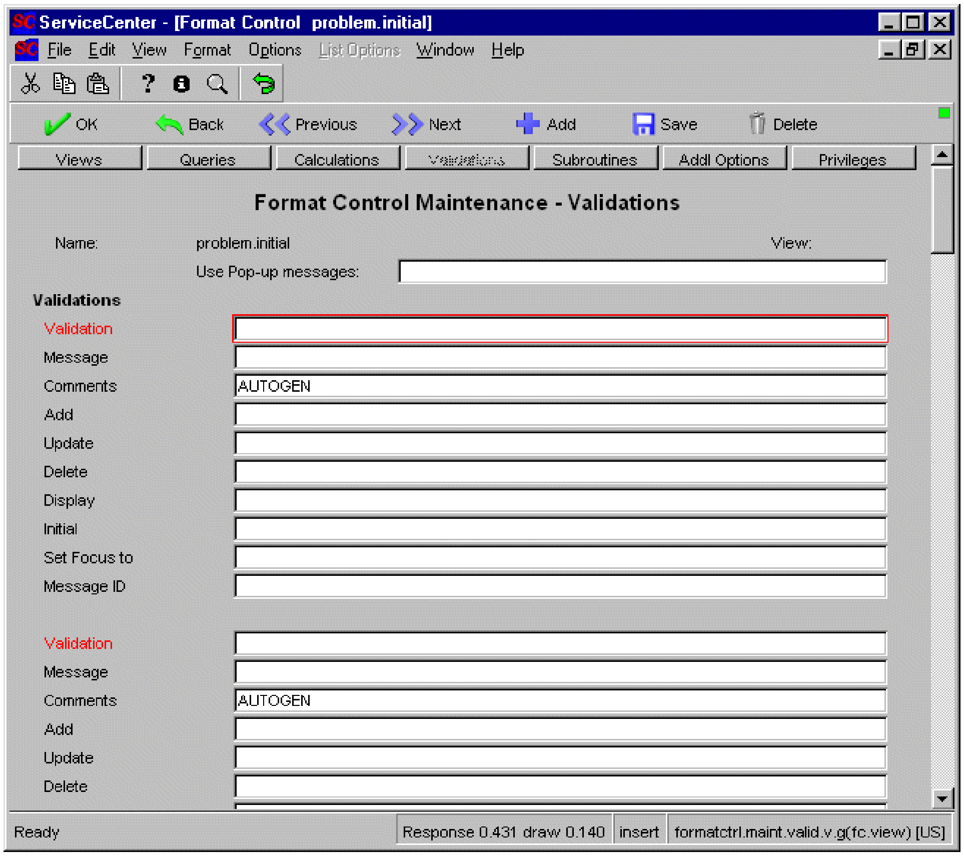
Error message displayed when a field value, defined in the Validation expression, fails to evaluate to *true*.

Message ID

Error message number to display instead of the Message when the validation fails. If a Message Number is specified, a corresponding message from the *scmessage* file displays when the Validation fails, rather than displaying the text in the Message field discussed above. Since the *scmessage* file can hold messages in many languages, this is the preferred method for multi-lingual implementations.

Comments

Brief comment about the field being validated.


ServiceCenter - [Format Control problem.initial]
File Edit View Format Options List Options Window Help
OK Back Previous Next Add Save Delete
Views Queries Calculations Validations Subroutines Addl Options Privileges

Format Control Maintenance - Validations

Name:	problem.initial
View:	
Use Pop-up messages:	

Validations

Validation	
Message	
Comments	AUTOGEN
Add	
Update	
Delete	
Display	
Initial	
Set Focus to	
Message ID	
Validation	
Message	
Comments	AUTOGEN
Add	
Update	
Delete	

Ready | Response 0.431 draw 0.140 | insert | formatctrl.maint.valid.v.g(fc.view) [US]

Figure 13-4: Validations—Long Form

Examples

Required Fields

Use the following values to make a field (**location**) required when adding a record or updating a database file:

Field	Value
Add	true
Update	true
Validations	not null(location in $file) or location in $file~=NULL or location in $file<>NULL
On Failed Validation Set Focus To	location
Message	The location field is required.

Specific Values

To make a field begin with a specific value when adding or updating a record:

Field	Value
Add	true
Update	true
Validations	type in $file #“US”
On Failed Validation Set Focus To	type
Message	The type field must begin with US.

Range of Values

To validate a value within a range:

Field	Value
Add	true
Update	true
Validations	amount in $file<=100000
On Failed Validation Set Focus To	amount
Message	The amount cannot be greater than 100000.00

Conditional Values

Some validations are conditional, i.e., you only want to perform the validation if other specific conditions occur. In the example below, the Sales dept. in Atlanta has only two valid cost centers: 2284 and 3401. The example shows a cost.center validation for the contacts file that occurs only if the location and dept fields contain Atlanta and Sales, respectively. This type of conditional validation works well if you have only one or two exceptional situations. However, if you have many combinations to check and verify, setting up separate conditional validations for each is very inefficient. Instead, use a more appropriate method such as:

- Set up a new file listing the various acceptable combinations.
- Use a Format Control File Query to read the appropriate record from the new secondary file.
- Setup a single Format Control validation to verify that the current value matches the secondary file value.

In the **Calculations** tab, add a new line

Field	Value
Add	True

Field	Value
Update	True
Calculation	if (location in $file=”Atlanta” and dept in $file=”Sales”) then $atlsales=true else $atlsales=false

Note: These conditions can also go directly into the Add, Update, Delete, and Display fields on the Validations tab, but they are harder to read and require slightly more processing since they must be evaluated twice, for both the Add and Update conditions.

In the **Validations** tab, add a new entry:

Field	Value
Add	$atlsales
Update	$atlsales
Validation:	cost.center in $file isin{”2284”, “3401”}
Message:	The cost center for Atlanta Sales personnel must be 2284 or 3401
On Failed Validation Set Focus To:	cost.center

To validate a value based on a condition:

1 Enter the following expression in the Calculations section of the Format Control record:

```
if (manufacturer in $file=“ibm”) then ($charge=false) else ($charge=true)
```

2 Enter the following in the Validations section of the Format Control record.

Field	Value
Add	$charge
Update	$charge

Field	Value
Validations	repair.charge in $file>0
On Failed Validation Set Focus To	repair.charge
Message	When the manufacturer is IBM, there cannot be a repair charge.

Fields in an Arrayed Structure

The Validations process, in conjunction with the Calculations process, can be used to validate fields in an arrayed structure. In this example, we are validating the **Serial Number** field in the **devices.covered** array of the **service** form. If a user enters data in the **Model**, **Install Date** or **Warranty** fields, there must also be a valid Serial Number. Use the following values to set up the Format Control record:

- **Calculations Process**

Field	Value
Add	true
Update	true
Calculation expression	$errorlist={};if (not null(devices.covered in $file)) then for $i = 1 to lng(denull(devices.covered in $file)) do (if (not null($i in devices.covered in $file)) then if null(1 in $i in devices.covered in $file) then ($i in $errorlist=str($i)))

Note: The Calculation expression must be a single line.

Validations Process: Use either version A or version B.

Example A:

Field	Value
Add	true
Update	true
Validation	lng(denull($errorlist))=0
Message	For each DEVICE COVERED you must supply a serial number

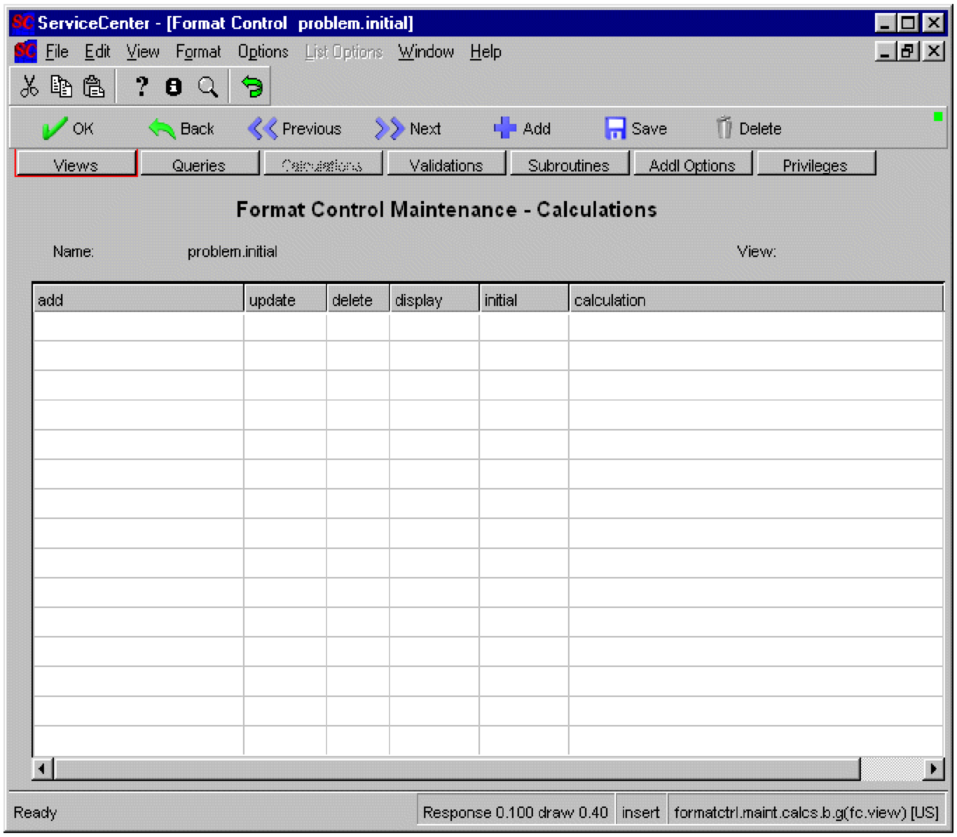
Example B:

Field	Value
Add	lng(denull($errorlist))>0
Update	lng(denull($errorlist))>0
Validation	false
Message	For each DEVICE COVERED you must supply a serial number

Calculations

Select **Calculations** from the Options menu to open the Calculations process of the Format Control record, or click the **Calculations** button.

The Calculations process in Format Control is similar to a **process** Command panel and can perform the same functions without involving RAD programming.


ServiceCenter - [Format Control problem.initial]
File Edit View Format Options List Options Window Help
OK Back Previous Next Add Save Delete
Views Queries Calculations Validations Subroutines Addl Options Privileges

Format Control Maintenance - Calculations

Name: problem.initial View:

add	update	delete	display	initial	calculation

Ready | Response 0.100 draw 0.40 | insert | formatctrl.maint.calcs.b.g(fc.view) [US]

Figure 13-5: Calculations—Short Form

Fields

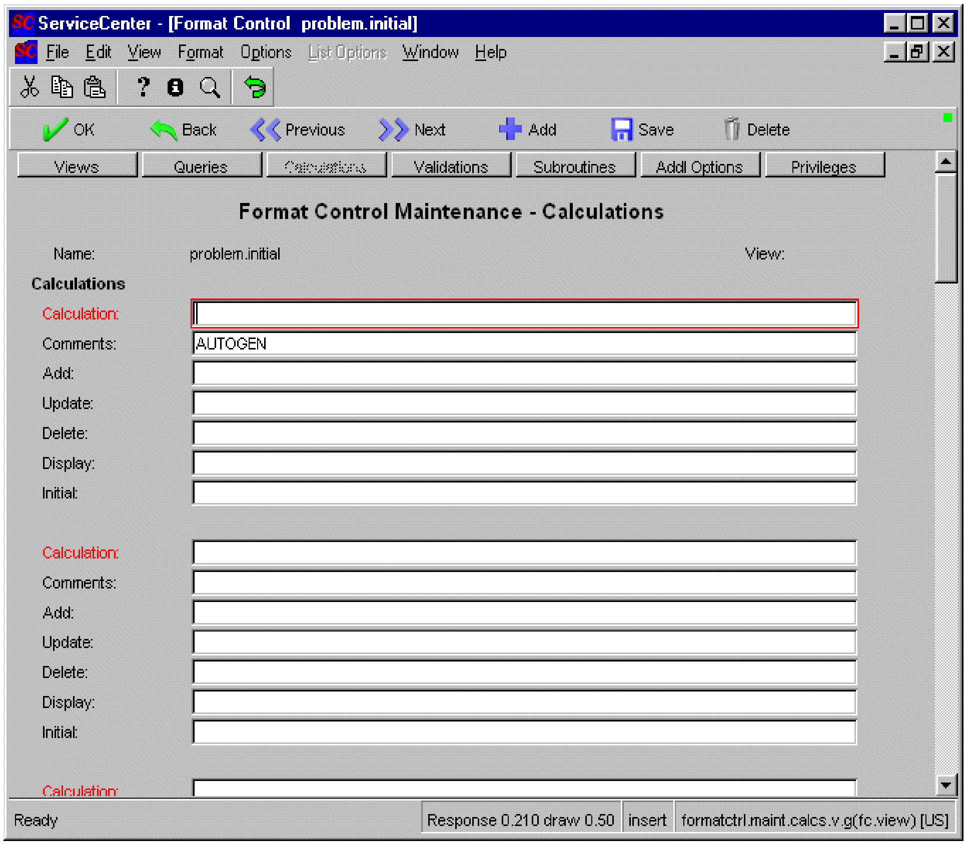
Add, Update, Delete, Display, Initial

These are Boolean fields that control Format Control processing. Processes evaluating to *true* apply Format Control before completing the specified operation. For example, when a user clicks **Save**, and before the record updates, Format Control executes every section or expression where the Update field evaluates to *true*.

Calculation

The Calculation expressions are only performed on variables or fields that are currently available, either from the primary file or any additional file that has already been accessed from Additional File Queries.

Note: If the Format Control record is to be used with reports, it is important to remember that the processes you set up will be performed every time the form is accessed. If the Format Control record is on a **detail** form, the calculations will be performed on every record. If the Format Control record is on the **header** form, calculations will be performed on every page.


ServiceCenter - [Format Control problem.initial]

Format Control Maintenance - Calculations

Name:	problem.initial
View:	

Calculations

Calculation:	
Comments:	AUTOGEN
Add:	
Update:	
Delete:	
Display:	
Initial:	
Calculation:	
Comments:	
Add:	
Update:	
Delete:	
Display:	
Initial:	
Calculation:	

Ready | Response 0.210 draw 0.50 | insert | formatctrl.maint.calcs.v.g(fc.view) [US]

Figure 13-6: Calculations—Long Form

Examples

Simple calculations

- Calculate the total amount from price and tax given below:

  price in $file=120.00;$tax=0.05;

Calculation:

  $amount=price in $file*$tax

Result:

  $amount=6.00

- Perform expressions (math) on variables given below:

  $a=8, $b=4, $c=5, $d=2

calculations	results
$x=$c*$b+1	$x=21
$x=$c*($b+1)	$x=25
$x=$a*$c/$b - $d	$x=8

Note: Multiplication (*) and division(/) are calculated first—addition (+) and subtraction (-) next. Subtraction must be separated by spaces (9 - 5).

Moving Characters in a Field

Strip the first three characters from a field and insert them into another field. If the first three bytes of a serial number were the same as the manufacturer's name:

serial.no.="ibm173605"

Calculation Expression:

manufacturer in $file=substr(serial.no. in $file, 1 ,3)

Result:

manufacturer="ibm"

Calculations on Fields in Arrayed Structures

Perform calculations on fields in arrayed structures.

The format:

(1)manufacturer	(2)model	(3)quantity	(4)price	(5)amount
(1) ibm	123	2	100.00	200.00
(2) dec	222	4	200.00	800.00
(3)				

The dbdict:

- arrayed.structure.example
- arrayed.structure.example
- manufacturer
- model
- quantity
- price
- amount

The operator enters **manufacturer**, **model**, **quantity** and **price** in a database file. Calculate the total amount of all items in the array: (amount=quantity*price)

Calculation Expression:

```
5 in 1 in arrayed.structured.example in $file=3 in 1 in arrayed.structure.example in
$file*4 in 1 in arrayed structure.example in $file
```

In this example, the expression would evaluate to:

amount=2*100.00

amount=200.00

This would have to be repeated for each possible element in the arrayed structure. To do so, you must use a **for** loop:

```
for $i=1 to lng(denull(arrayed.structure.example.in$file)) do 5 in $i in
arrayed.structure.example in $file=3 in $i in arrayed.structure.example in $file* 4 in
$i in arrayed.structure.example in $file
```

Important: This statement must be entered as one continuous line.

Calculation Expression:

```
5 in 2 in arrayed.structured.example in $file=3 in 2 in arrayed.structure.example in
$file*4 in 2 in arrayed.structured.example in $file.
```

In this example, the expression would evaluate to:

amount=4*200.00

amount=800.00

Note: The reference of a field (in an element) located in a structured array is based on how it is defined in the database dictionary, not how it is displayed in the form. Therefore, if *quantity* was the fifth entry in the arrayed structure in the dbdict, it would be referenced as 5 in 1 in arrayed.structure.example in `$file`, rather than 3 in 1 in arrayed.structure.example in `$file` as shown in this example.

Counting Tickets of Differing Severity Levels in a Report Writer Report

Count the number of problem tickets opened over a period of time based on severity level in a ServiceCenter Report Writer report:

```
if(severity in $file=1) then ($sev1+=1)
if(severity in $file=2) then ($sev2+=1)
if(severity in $file=3) then ($sev3+=1)
```

At the end of the processing `$sev1`, `$sev2`, and `$sev3` will contain the total number of problems for each severity level.

Note: `$sev1`, `$sev2`, and `$sev3` would have to be initialized once to zeros using an assignment expression ($sev1=0) in the Initializations form of the Format Control record.

Counting Open Problem Tickets in a Report Writer Report

Create a report that counts the number of problem tickets that are still open over a period of time. Expressions defining tickets still open for periods of one week, two weeks, three weeks, and over four weeks:

```
if (open.time in $file>tod() - ‘7 00:00’) then ($week1+=1)
if (open.time in $file>tod() - ‘14 00:00’ and open.time in $file<tod() - ‘7 00:00’) then
($week2+=1)
if (open.time in $file>tod() - ‘21 00:00’ and open.time in $file<tod() - ‘14 00:00’) then
($week3+=1)
if (open.time in $file>tod() - ‘21 00:00’) then ($greater.than3+=1)
```

In the totals format of the report:

```
Number Of Problems Still Open Within The Last Week: ($week1)
Number Of Problems Still Open Between 1 & 2 weeks: ($week2)
Number Of Problems Still Open Between 2 & 3 Weeks: ($week3)
Number Of Problems Still Open Over 3 Weeks: ($greater.than4)
```

Note: The Format Control record would be attached to the **detail** form of the report in order for the calculation to be performed for each record selected by the report. The variables would have to be initialized to zeros one time only using the nullsub function in the Initializations form of the Format Control record.

Denulling an Array

Denull an array before printing in a report, so the underscores in blank lines are not printed.

```
comments in $file=denull(comments in $file)
```

Using Calculations in Reports

- Create a report to calculate the percentage of problem tickets that were opened and closed within the same day:

Attach the Format Control record to the **detail** form of the report.

```
display: true
if (date(open.time in $file)=date(close.time in $file)) then
($resolved+=1);$all.recs+=1
```

Note: :`$resolved` and `$all.recs` must to be set to zeros using the nullsub function in a Format Control record attached to the report **header** form. This would be a one-time initialization.

Attach a Format Control record to the **totals** form:

```
$perc=(round(($resolved/$all.recs)*100, 1)
```

Note: The **1** at the end of the expression is the number of positions after the decimal point (43.7).

- Create a report to calculate the down time for specific device types over a period of time:

Attach the Format Control record to the **detail** form of the report.

```
$down=up.time in $file - down.time in $file
$dt=(1*$down)/60
```

This calculates downtime in hours.

Note: In this case, the result of the calculation (1*$down) converts the type of $down (which is now date/time) to numeric. The system stores this numeric value as the number of minutes since year 01/01/0001 00:01.

- Combine all the information from the **Description** and **Update Description** arrays of an incident ticket, regardless of array size, and attach the following Format Control calculation to the **detail** form of the report:

```
$action={"DESCRIPTION"}+denull(action in $file)+{"UPDATE ACTIVITY"}+denull(update.action in $file)+denull(resolution
in $file)
```

Note: These calculations denull the action and update.action arrays and store their lengths. The lengths are then used to check and concatenate the information into one readable block.

Put the following label and input field on the **detail** form where the descriptions are to begin:

DESCRIPTION:

_

($action) window of (-1)

An example of what the output would look like is:

DESCRIPTION:

description line 1

description line 2

UPDATE ACTIVITY

update line 1

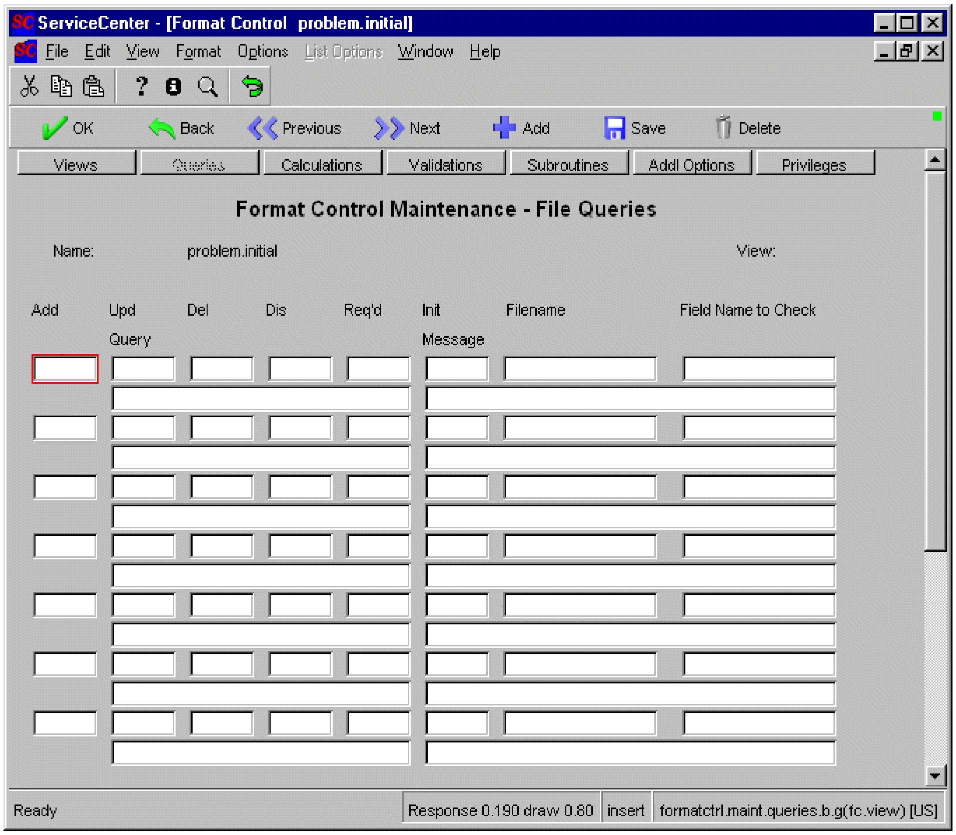
update line 2

update line 3

Additional File Queries

Select **Queries** from the Options menu to open the Additional File Queries process of the Format Control record, or click the **Queries** button. This process validates data in the current file against data in other ServiceCenter files or counts records in an associated file and returns the results.

The Additional File Queries process has four boolean fields that, if evaluated to true, will be evaluated BEFORE any task is performed on a record.



Format Control Maintenance - File Queries

Name: problem.initial

View:

Add Upd Del Dis Req'd Init Filename Field Name to Check

Query Message

Ready | Response 0.190 draw 0.80 | insert | formatctrl.maint.queries.b.g(fc.view) [US]

Figure 13-7: Additional File Queries—Short Form

Fields

Filename

Enter the name of the database file to be used for the secondary file query in this field. You may use either a *variable* or a *literal* for the file name.

The **primary file** (named in the **Name** field) is referred to in the Format Control record as **$file**. Each additional file that is called in this section will be referred to as **$file1**, **$file2**,....**$filen** depending upon their position on the additional file query line.

For example, the **contacts** file (the additional file to be queried from this form) will now be referred to as **$file1**.

Query

This is the expression used to select records from the database file specified in the **Filename** field. Once a match has been made, the data in the additional file is available for reference.

For example:

> **contact.name=contact.name in $file**

contact.name is the input field name from the **contacts** dbdict. **contact.name** does not have to reference a **$filename** here, but refers instead to the name in the **Filename** field. You could interpret this expression as:

> **contact.name in contacts file=contact.name in problem file.**

Comments

A brief comment about the use of this query or an indication that the file query should return a count of the number of records found by the query.

To indicate that a count should be returned, begin your comment with "count." For example, "count location records". The count is returned as a number in a global array variable called *$file.count*. The position in the list is the same as the file number—that is, the count for **$file2** would be **2** in **$file.count**.

Add, Update, Delete, Display,Initial

These are Boolean fields that control Format Control processing. Processes evaluating to *true* apply Format Control before completing the specified operation. For example, when a user clicks **Save**, and before the record updates, Format Control executes every section or expression where the Update field evaluates to *true*.

Required Query? (short form: Req'd)

This field is used for validations in additional files. If the field evaluates to *true*, the indicated function (**Add**, etc.) will not be performed unless the system finds an appropriate record to match the specified query. A suitable message is displayed to the user.

If this field evaluates to *true* in our example, the additional file, *contacts,* would be queried to ensure that a valid entry was made in the problem ticket for the contact's name. For example, if **JACKSON** is entered in the problem ticket, but no record for **JACKSON** is found in the **contacts** file, the problem ticket is not added to the database and the user receives an error message.

Required Field Name (short form: Field Name to Check)

This is an optional field allowing you to check for a NULL state in a primary record field when no records are found as a result of executing the query against the specified file. At run time, Format Control first performs the requested query against the specified file. If one or more records are found, processing continues normally to the next Format Control definition.

If no records are found, Format Control checks for a value in the Required Field Name field. If a value is defined, it then checks the contents of that field in the primary record. The sequence of events is as follows:

- If the **Required Field Name** field is NULL and the **Required Query** condition evaluates to *true*, Format Control issues the specified error message. If an error message is not defined, it issues a default message.
- If the **Required Field Name** field is not NULL, Format Control checks for a NULL state in the contents of *that* field in the primary file. For example, if the **Required Field Name** field contains the value *location*, Format Control examines *location in $file* for a NULL condition. If Format Control does not detect a NULL condition, and the **Required Query** condition evaluates to *true*, Format Control issues the specified error message.

Note: If you need to check for multiple complex conditions in order to control the evaluation of the secondary file query, we recommend that you use the *add*, *update*, and *delete* conditions to specify the complex condition.

Error Message

This is the message to be displayed to the user if no records are found with the query. The value must be valid in the secondary file.

For example:

Vendor is not a valid name in the vendor file—please re–enter.

Privileges

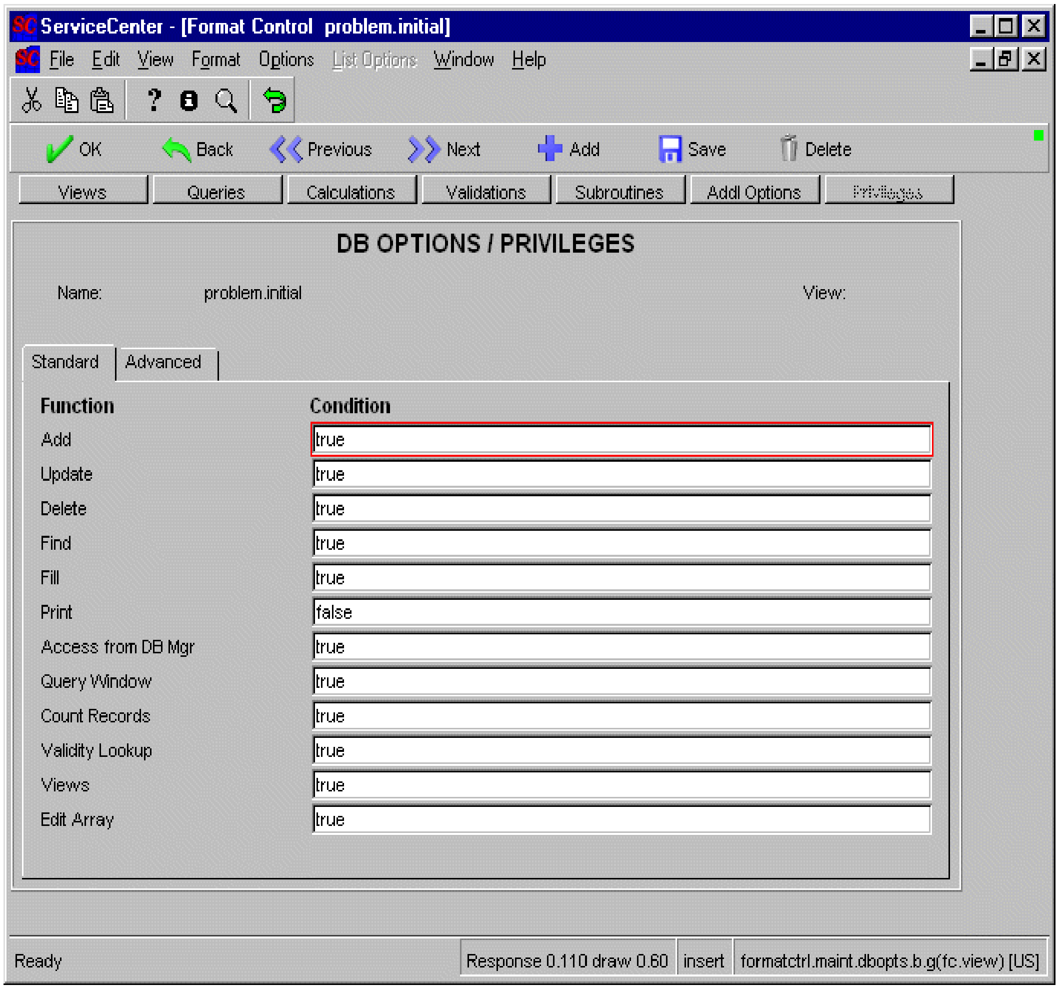
To access the Security process of the Format Control record, select **Privileges** from the Options menu or click **Privileges**. There is only one form associated with Privileges.

The Security process of the Format Control record is used to control which options (System Tray buttons) will be available to the user.

All the options in this process of the Format Control record are boolean. If they evaluate to *true*, the options will be available to all users. If they evaluate to *false*, the option will not be offered to any user.

Expressions can also be used in all Security fields to make the options available to users based on their capabilities as defined in their Operator Records or based on values in the data at runtime. Note that in order to reference the `$file` variable and have it evaluated, you must have initialized the variable in some other area of the format control. Refer to the *Expressions in Boolean (Logical) Fields* earlier in this chapter.

Format Control Security options are not available for Incident Management, Change Management and the Device Inventory portion of Inventory and Configuration Management. These applications have their own security written into the applications.


ServiceCenter - [Format Control problem.initial]
File Edit View Format Options List Options Window Help
OK Back Previous Next Add Save Delete
Views Queries Calculations Validations Subroutines Addl Options Privileges

DB OPTIONS / PRIVILEGES

Name: problem.initial View:

Standard | Advanced

Function	Condition
Add	true
Update	true
Delete	true
Find	true
Fill	true
Print	false
Access from DB Mgr	true
Query Window	true
Count Records	true
Validity Lookup	true
Views	true
Edit Array	true

Ready | Response 0.110 draw 0.60 | insert | formatctrl.maint.dbopts.b.g(fc.view) [US]

Figure 13-8: Security Process: Standard Tab

Standard Tab Fields

Add

Controls a user's ability to add a record to the database. This applies to Add and Add/Retain. The default value is *true*.

Update

Controls a user's ability to update a record. The default value is *true*.

Delete

Controls a user's ability to Delete a record. The default value is *true*.

Find

Determines if the **Find** button will be displayed, allowing the user to link to another file and view the data within a desired record. The default value is *true*.

Fill

Determines if the **Fill** button will be displayed, allowing the user to link to another file and extract data from within a desired record to be placed in the current record. The default value is *true*.

A Link record must be set up for the **Find** and **Fill** options to work correctly. If a link record is not set up, the **Find** and **Fill** buttons will not be displayed.

Print

Determines whether or not the Print option (mass and record level) appears in the Options menu of a file opened in Database Manger. The default value is *false*.

Access From Db Mgr

Determines whether or not the user can access the Database Manager using the form to which the Format Control record is attached. If this field evaluates to *false*, users will be denied access to the database using this form. The default value is *true*.

Users who are denied access will see the error message: *This file CANNOT be accessed by the DB Mgr. due to formatctrl restrictions.*

Query Window

Determines whether or not the user is allowed to open the Advanced Search Window. (Refer to the *Database Manager* documentation for information on this option.) The default value is *false*.

Count Records

Determines whether or not the **Count Records** button appears in the System Tray of the QBE list. The default value is *true*.

Validity Lookup

Determines whether or not the **Validity Lookup** option appears in the Options menu of a blank form in Database Manager. The default value is *true*.

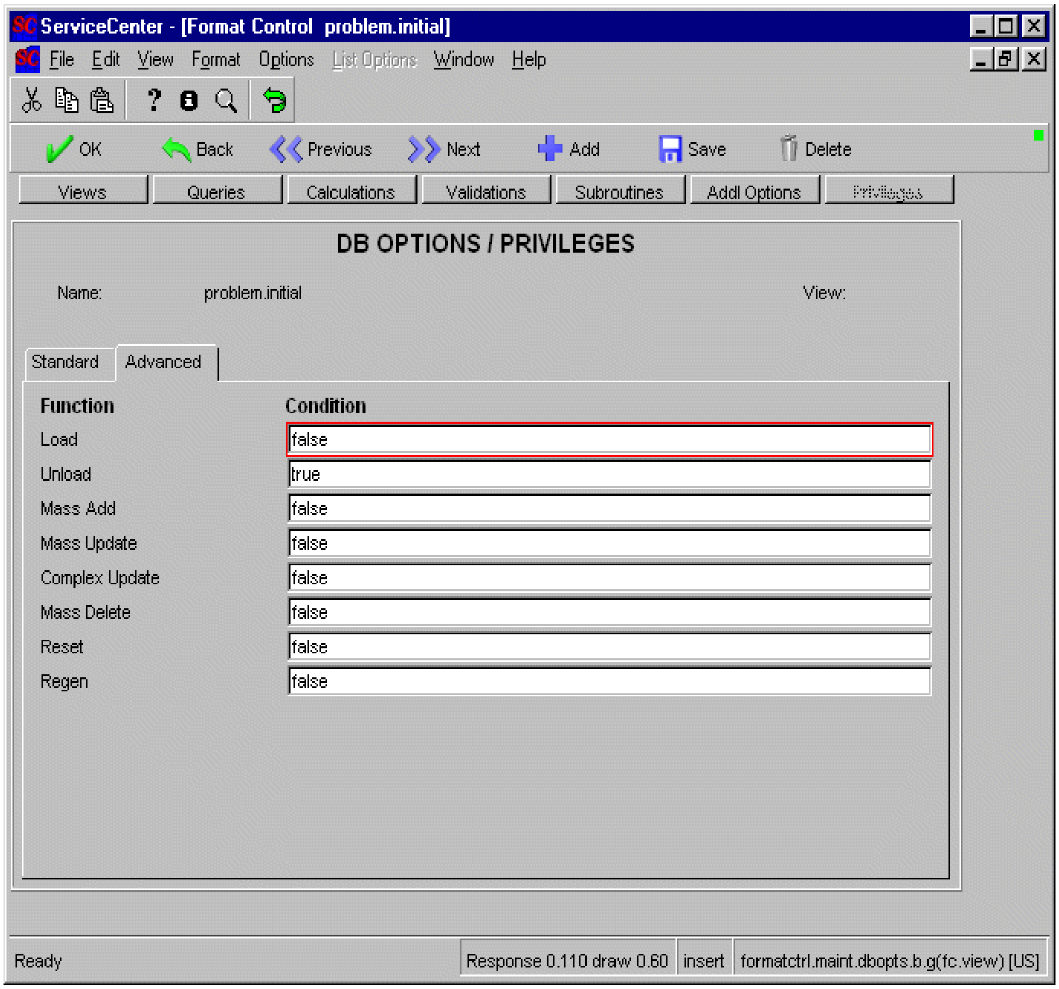
Views

Determines whether or not the Views button appears in the System Tray providing access to alternate QBE forms. These are the forms defined in the Views process of Format Control. The default is *true*.

Note: If no Views are defined for a particular form, the **Views** button will not be enabled, regardless of privilege.

Edit Array

Determines the users' ability to edit array structures. The default value is *true*.


ServiceCenter - [Format Control problem.initial]
File Edit View Format Options List Options Window Help
OK Back Previous Next Add Save Delete
Views Queries Calculations Validations Subroutines Addl Options Privileges

DB OPTIONS / PRIVILEGES

Name: problem.initial View:

Standard Advanced

Function	Condition
Load	false
Unload	true
Mass Add	false
Mass Update	false
Complex Update	false
Mass Delete	false
Reset	false
Regen	false

Ready | Response 0.110 draw 0.60 | insert | formatctrl.maint.dbopts.b.g(fc.view) [US]

Figure 13-9: Security Process: Advanced Tab

Advanced Tab Fields

Load

Determines the user’s ability to load data from an external file into the current file. The default value is *false*.

Unload

Controls a user's ability to Unload records to an external file. The default value is *true*.

Mass Add

Determines whether or not users have access to the Mass Add feature. The **Add** option appears in the Options menu of a QBE list. The default value is *false*.

Mass Update

Determines whether or not users have access to the Mass Update feature. The **Update** option appears in the Options menu of a QBE list. The default value is *false*.

Complex Update

Determines whether or not the **Complex Update** button appears in the System Tray during a Mass Update procedure. The default value is *false*.

Mass Delete

Determines whether or not the **Delete** option appears in the Options menu of a QBE list. The default value is *false*.

Reset

Determines if the **Reset** option appears in the Options menu.

Warning: A **reset** deletes ALL records from the current database file. Make this option unavailable to most users. The best practice is to set the value of the field to *false* and change it to *true* only when you need to delete all records from a file. The default value is *false*.

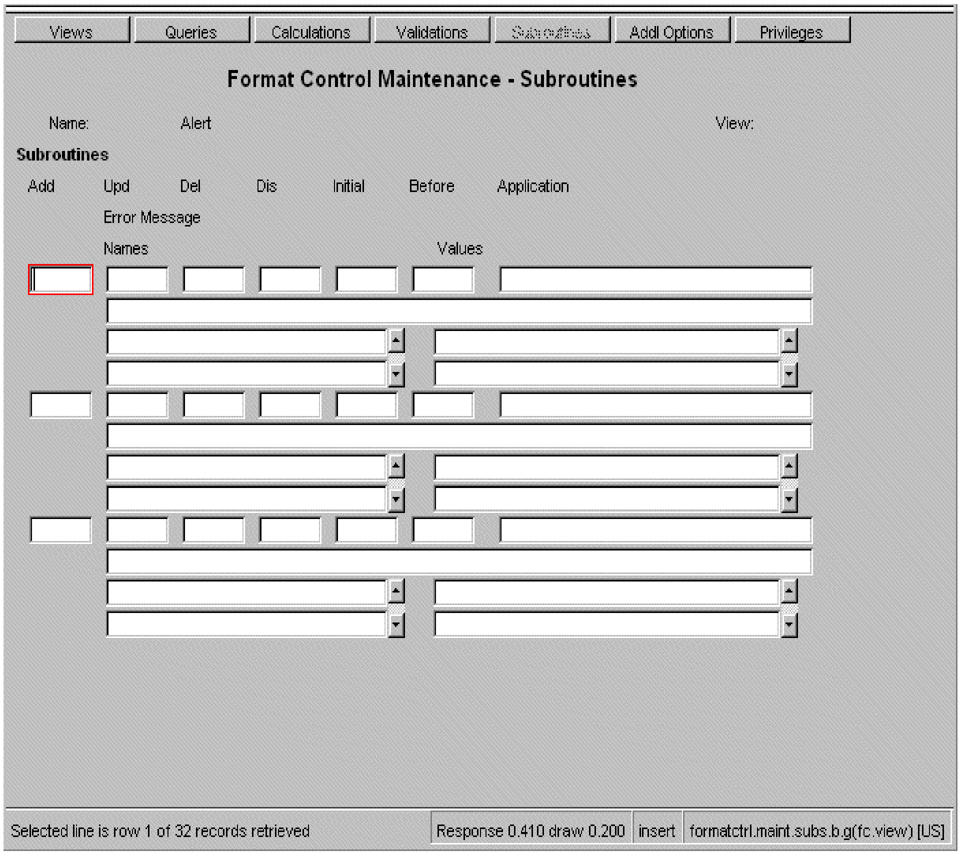
Regen

Determines if the **Regen** option appears in the Options menu.

Warning: A **regen** will rebuild all indexes for every record in the current database file. All users will be locked out of this file while the regen is executing. If the regen is stopped before completion, records remaining to be processed may not be available to the users. The best practice is to set the value of the field to *false* and change it to *true* only when you need to *regen* a file. The default value is *false*.

Subroutines

Select **Subroutines** from the Options menu to display the Subroutines form. The Subroutines process allows you to call subroutines that can do anything available through RAD. The subroutine can be run at **add**, **update**, **delete**, and **display/print** time, either before or after the process is performed.



Format Control Maintenance - Subroutines

Name: Alert View:

Subroutines

Add Upd Del Dis Initial Before Application

Error Message

Names Values

Selected line is row 1 of 32 records retrieved | Response 0.410 draw 0.200 | insert | formatctrl.maint.subs.b.g(fc.view) [US]

Figure 13-10: Subroutines—Short Form

Fields

Add, Update, Delete, Display

These are Boolean fields that control Format Control processing. Processes evaluating to *true* apply Format Control before completing the specified operation. For example, when a user clicks **Save**, and before the record updates, Format Control executes every section or expression where the Update field evaluates to *true*.

Initial

This field sets initial values for variables and fields to be passed to the RAD application specified in the **Application** field.

Before

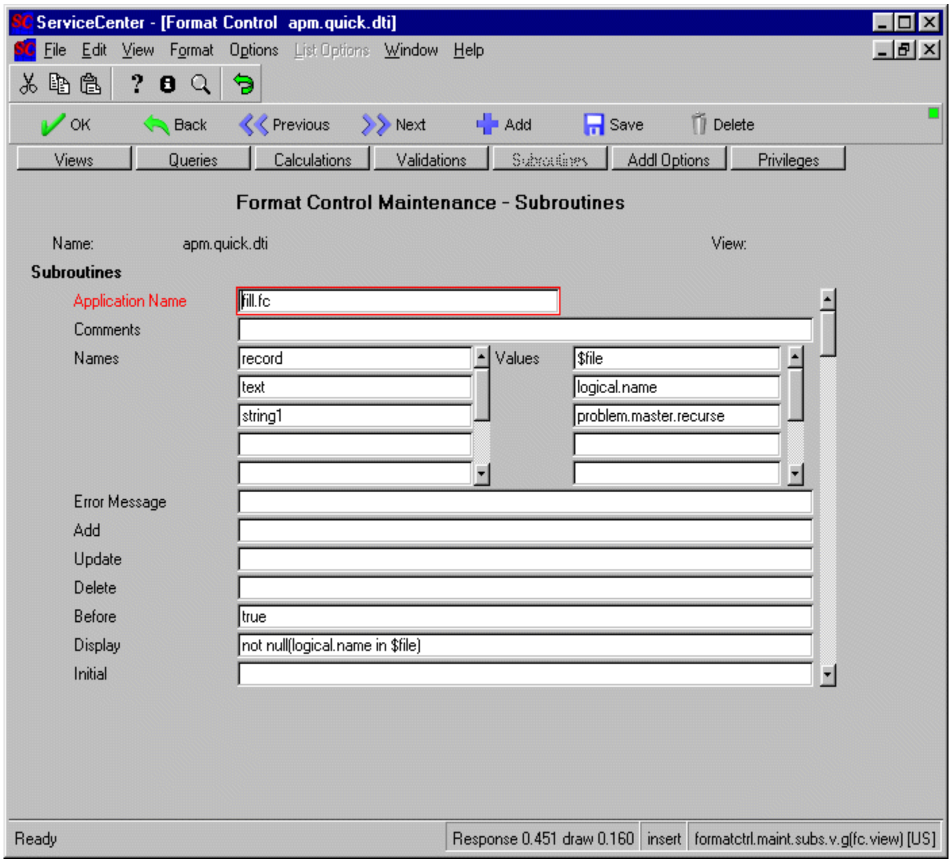
Determines whether the subroutine will be performed before or after the specific **add**, **update**, **delete**, or **display** operation. The default is *false* (blank), meaning the subroutine will be performed after the specified operation. In most cases, you should change this to *true*.

Application

The name of the RAD application to be executed when the specified operation is performed. This must be an existing application.

Error Message

This message will be displayed if there is an error executing the application.


ServiceCenter - [Format Control apm.quick.dti]
File Edit View Format Options List Options Window Help
OK Back Previous Next Add Save Delete
Views Queries Calculations Validations Subroutines Addl Options Privileges

Format Control Maintenance - Subroutines

Name: apm.quick.dti View:

Subroutines

Application Name	fill.fc		
Comments			
Names	record	Values	$file
	text		logical.name
	string1		problem.master.recurse
Error Message			
Add			
Update			
Delete			
Before	true		
Display	not null(logical.name in $file)		
Initial			

Ready | Response 0.451 draw 0.160 | insert | formatctrl.maint.subs.v.g(fc.view) [US]

Figure 13-11: Subroutines—Long Form

Parameters

Name List

The names of the parameters used to pass data to the subroutine application if necessary. The name must be a valid Peregrine defined parameter input field name. Refer to the *RAD User Guide* for a list of these parameter names and their uses.

For example:

record (*type of record*)

Value List

The names of the fields or values to be passed to the subroutine application. The *type* of the field must match the *type* of the parameter defined in the **Name** field.

For example:

> $file (*to pass the whole record to the subroutine*)

Examples

Fingerprinting

Fingerprinting in Format Control allows a System Administrator to monitor the activity of a specific file. An application (**fingerprint**) called through Subroutines automatically records the names of users adding or updating records of fingerprinted files and the time each action is performed. A separate form, made available in the Views process, displays this information on demand. (Refer to the discussion of Views below.)

Fingerprinting is set up in four steps:

- Edit the dbdict.
- Create a form.
- Set up an Alternate View of the form created.
- Set up a Subroutine call of the **fingerprint** application.

Note: This example demonstrates how two Format Control processes can work together

Edit the dbdict

In order for the **fingerprint** field of the secondary file to be populated with data from the **contacts** file, **fingerprint** must appear as an **array** field of data type *character* in the dbdict record for the **contacts** file.

Add fingerprint as an array:

1. Select the **Toolkit** tab in the Home menu.
2. Click on the **Database Dictionary** button.
3. Enter contacts in the **File Name** field of the Database Dictionary dialogue box.

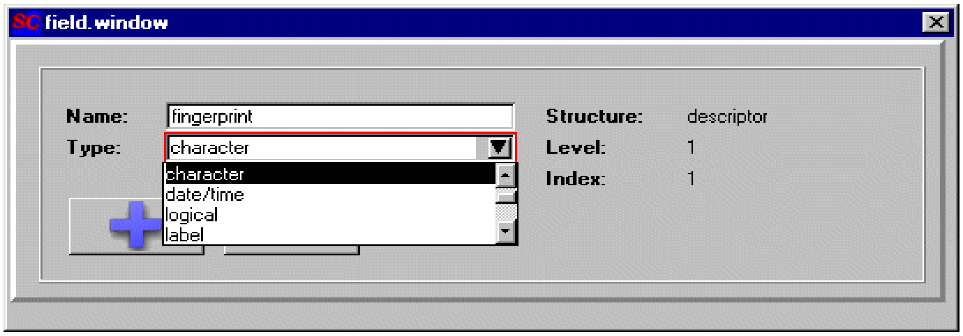
4 Press **Enter** or click on the **Search** button.

5 The dbdict record for the contacts file is displayed.

6 Place the cursor in the **descriptor** field of the dbdict.

7 Click on the **New** button.

8 Enter **fingerprint** in the **Name** field of the edit box and select **array** from the drop–down list.



Name:	fingerprint	Structure:	descriptor
Type:	character	Level:	1
		Index:	1

9 Click the **OK** button.

10 Enter **fingerprint** in the **Name** field of the edit box and select **character** from the drop–down list.

11 Click the **Add Field** button.

12 Exit the Database Dictionary.

Create a Form

Create a secondary form called **show.fingerprint** that will display the update data from the contacts file. This form will be referenced in the Views process of the Format Control record for the contacts file.

To create the form:

1 Select the Toolkit tab in the Home menu.

2 Click **Forms Designer**.

3 Enter show.fingerprint in the **Form** field of the Forms Designer dialogue box.

4 Click **New**.

5 Create a form consisting of a single Text field with the following values:

Note: Refer to the *Forms Designer* documentation for details on creating forms.

Property	Value
Input	fingerprint
X	2
Y	2
Height	2
Width	56
Array Length	10

1 Select the Label tool and name the field **Last Updated.**

2 Click **OK** to save your form.

3 Select **Create File** from the Options menu.

4 Enter `fingerprint` as the filename for your form in the prompt box.

5 Click **OK** to create the file.

6 Exit Forms Designer.

Set up an Alternate View in the Views Process

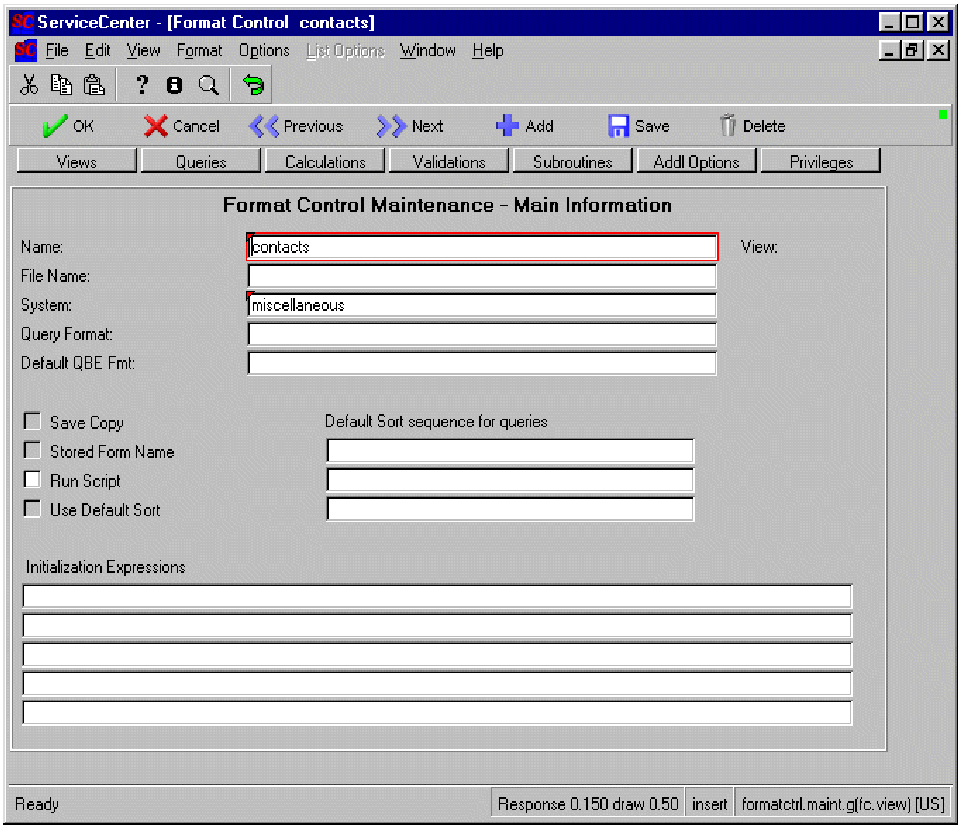
1 Click the **Tools** button in the Utilities tab.

2 Click **Format Control** in the Tools menu.

3 Enter `contacts` in the **Name** field of the Format Control Initializations form.

4 Press **Enter** or click **Search**.

5 The Initializations form is displayed.


ServiceCenter - [Format Control contacts]
File Edit View Format Options List Options Window Help
OK Cancel Previous Next Add Save Delete
Views Queries Calculations Validations Subroutines Addl Options Privileges

Format Control Maintenance - Main Information

Name: contacts

View:

File Name:

System: miscellaneous

Query Format:

Default QBE Fmt:

Save Copy

Stored Form Name

Run Script

Use Default Sort

Default Sort sequence for queries

Initialization Expressions

Ready | Response 0.150 draw 0.50 | insert | formatctrl.maint.g(fc.view) [US]

Figure 13-12: Initializations Form of Format Control

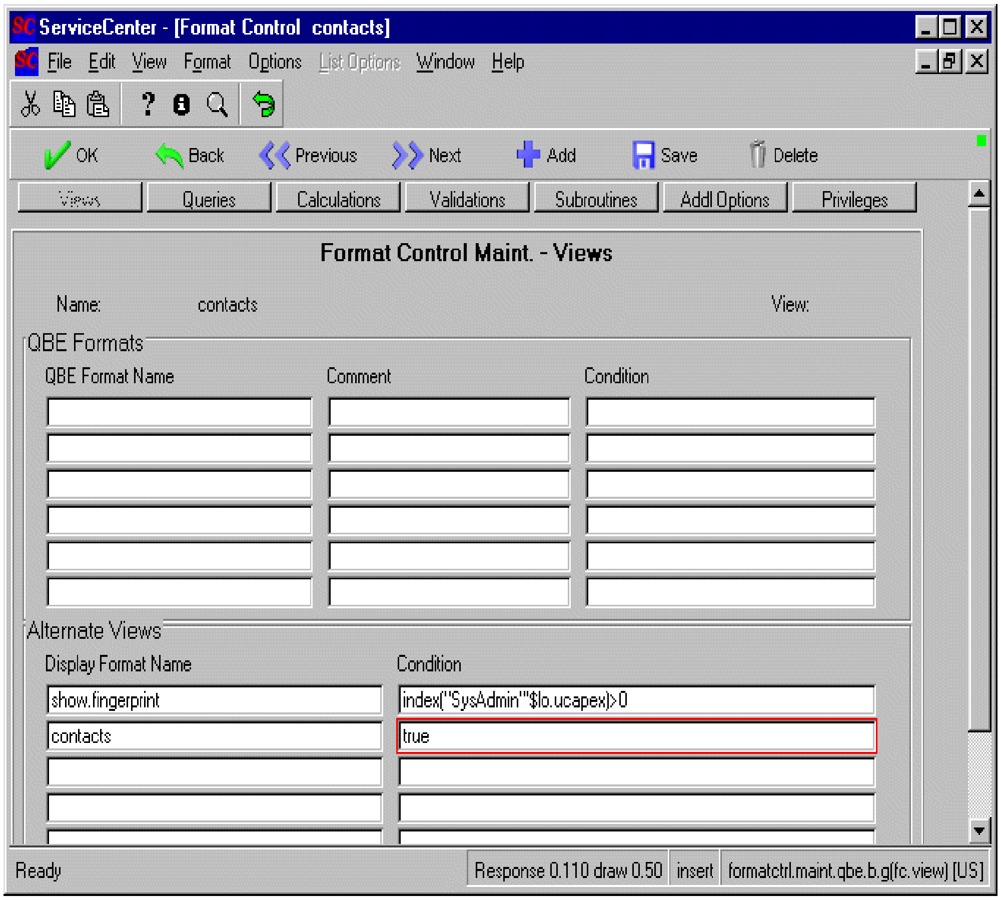
6 Select **Views** from the Options menu or click **Views**.

Note: For a complete discussion of the Views process, refer to *Views* on page 258

7 Enter the following data in the Alternate Views structure of the Views form:

Display Format Name	Condition
show.fingerprint	index("SysAdmin",$lo.ucapex)>0
contacts	true

The Condition expression for the **show.fingerprint** form restricts viewing privileges to users with System Administrator capabilities only.


ServiceCenter - [Format Control contacts]
File Edit View Format Options List Options Window Help
OK Back Previous Next Add Save Delete
Views Queries Calculations Validations Subroutines Addl Options Privileges
Format Control Maint. - Views
Name: contacts View:
QBE Formats
QBE Format Name Comment Condition
Alternate Views
Display Format Name Condition
Ready
Response 0.110 draw 0.50 insert formatctrl.maint.qbe.b.g(fc.view) [US]

Display Format Name	Condition
show.fingerprint	index("SysAdmin","$lo.ucapex)>0
contacts	true

Figure 13-13: Using the Views Process for Fingerprinting

8 Click **OK.**

Set Up a Subroutines Call

Set up a Subroutines call to the **fingerprint** application in which Format Control is applied to the *contacts* file *before* the record is added to or updated in the database.

To set the subroutine:

1 Select **Subroutines** from the Options menu or by clicking the Subroutines button.

2 Select **Show Expanded Form** from the Options menu.

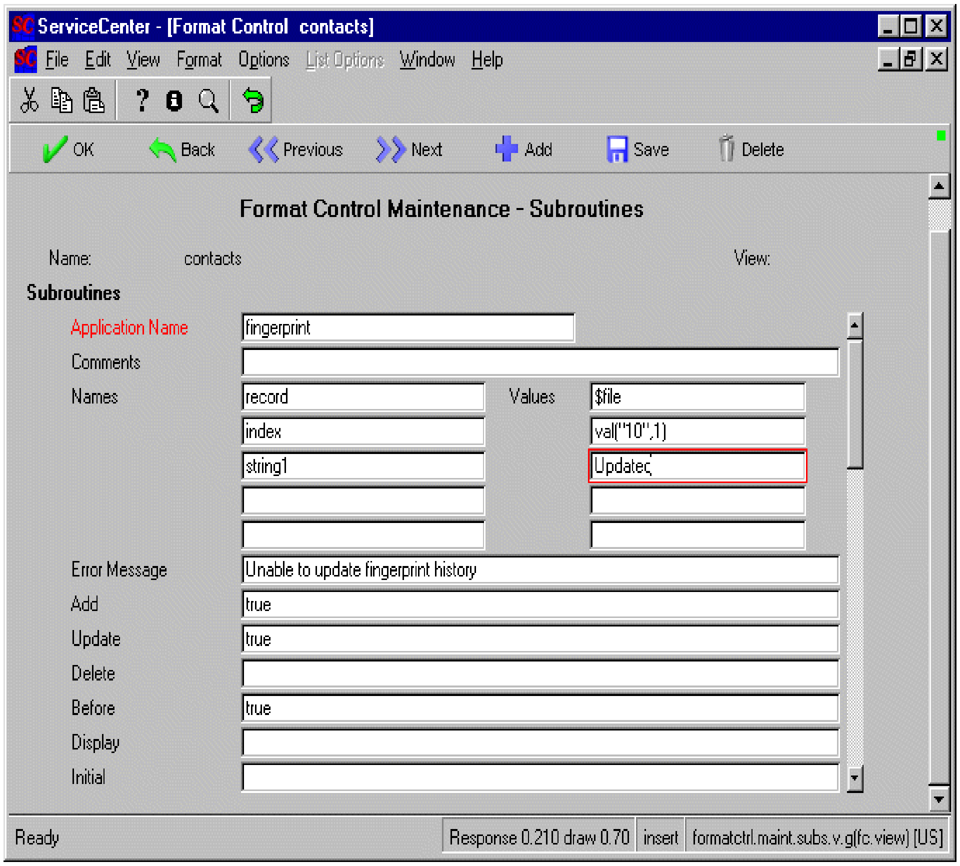
3 Enter the following data in the Subroutines form:

Name	fingerprint
Add	true
Update	true
Before	true

Parameters

Name	Value
record	$file
index	val("10", 1)
string1	Updated

4 Your Subroutines form will look like this:


ServiceCenter - [Format Control contacts]
File Edit View Format Options List Options Window Help
OK
Back
Previous
Next
Add
Save
Delete

Format Control Maintenance - Subroutines

Name: contacts View:

Subroutines

Application Name	fingerprint		
Comments			
Names	record	Values	$file
	index		val("10",1)
	string1		Updated
Error Message	Unable to update fingerprint history		
Add	true		
Update	true		
Delete			
Before	true		
Display			
Initial			

Ready | Response 0.210 draw 0.70 | insert | formatctrl.maint.subs.v.g(fc.view) [US]

Figure 13-14: Fingerprint Call from Subroutines

5 Click **OK**.

Update the *contacts* File and Check Fingerprinting

The parameters you have defined will display the names and dates of the last ten updates to the current **contacts** record, as made and viewed through the **contacts** form. Appended to the update information is a one word description of the action recorded, in this case **Updated**. Every time the **Save** button is clicked while viewing a **contacts** record via the **contacts** form, an additional entry to the **show.fingerprint** record array is added.

Important: You must be logged in as System Administrator in order to display the **show.fingerprint** form, and you must be viewing **contacts** records using the **contacts** form.

1. Select the **Toolkit** tab in the Home menu.
2. Click the **Database Manager** button.
3. Enter contacts in the Form field of the Database Manager dialogue box.
4. Press **Enter** or click **Search**.
5. Select **contacts** from the QBE list.
6. A blank **contacts** record is displayed.
7. Click **Search**.
8. A QBE list of contacts in your database is displayed.
9. Click on any contact name to display the contacts record.
10. Update any field.
11. Click **Save**.
12. Click the **Views** button to display a QBE list of the forms you have made available in the Alternate Views structure of the Views process.
13. Select **show.fingerprint** to display the fingerprinting form you created.
14. Scroll down the list to view earlier updates to the current Contact record.
15. Click **Views** to display the Views QBE list.
16. Select **contacts** to return to the Contact record.

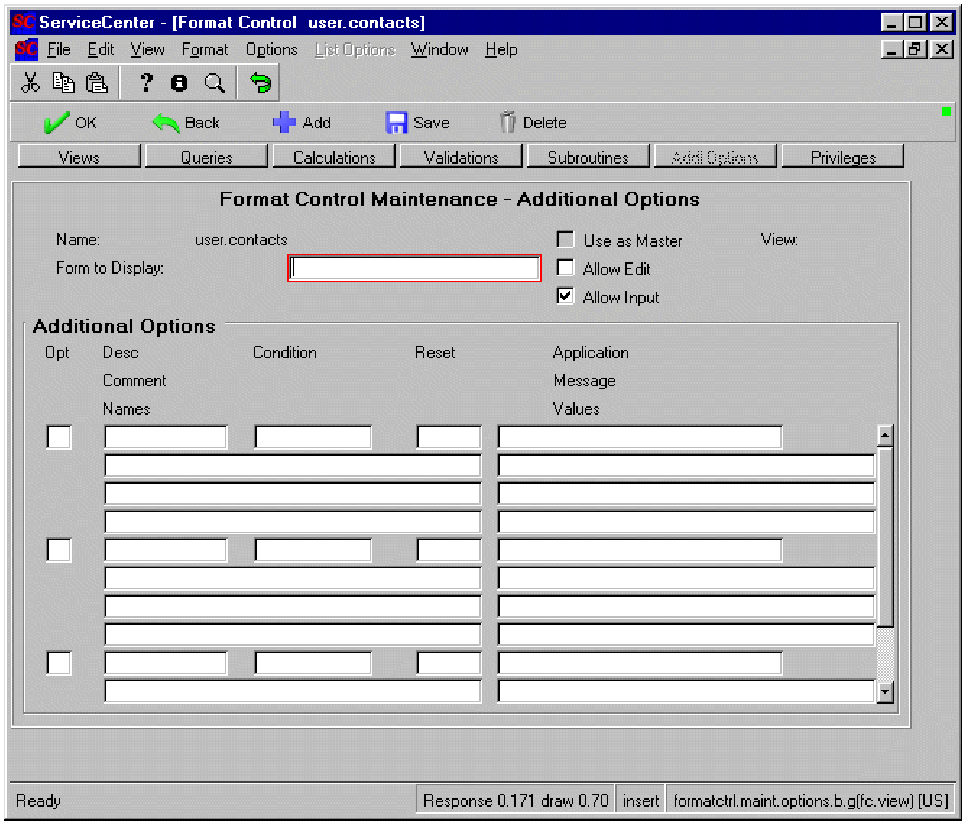
Additional Options

Select **Additional Options** from the Options menu, or click the **Addl Options** button, to display the Additional Options form.

Additional Options designates unique Options menu options (and corresponding F–keys) to perform additional RAD functions without actual RAD modifications. The Additional Options process layout is similar to that of the Menu file (where options on a menu are defined) and performs in the same fashion. Buttons are defined to run RAD applications (like **database**) from within another application. Once the application is finished running, the user is returned to the original application and form.

Settings within the Additional Options process can restrict access to a browse only mode or allow data input, depending upon a user's system rights.

Note: Additional Options are designed for use with the Database Manager only. Also, the ServiceCenter Document Engine and Display application are newer tools that provide similar functionality to Format Control Additional Options. They are generally preferable to Format Control Additional Options. However, Additional Options are still available, primarily because of carryover usage from previous ServiceCenter versions.


ServiceCenter - [Format Control user.contacts]
File Edit View Format Options List Options Window Help
OK Back Add Save Delete
Views Queries Calculations Validations Subroutines Add Options Privileges

Format Control Maintenance - Additional Options

Name: user.contacts

Form to Display:

☐ Use as Master

☐ Allow Edit

☑ Allow Input

View:

Additional Options

Opt Desc Condition Reset Application

Comment Message

Names Values

Ready | Response 0.171 draw 0.70 | insert | formatctrl.maint.options.b.g(fc.view) [US]

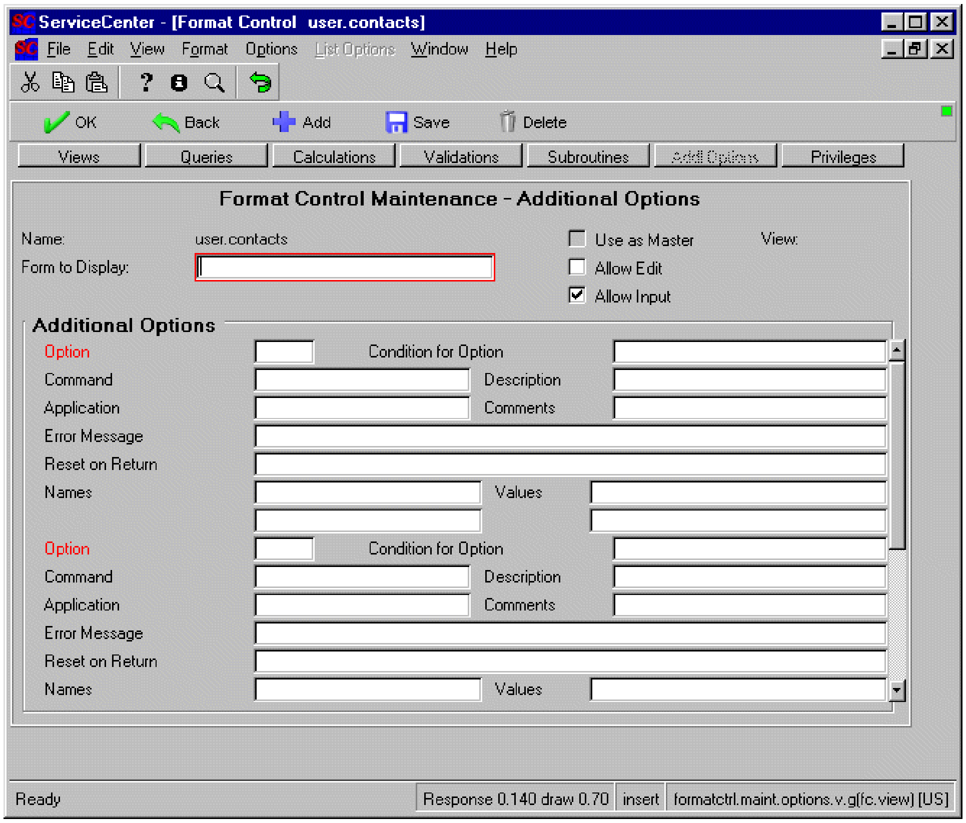
Figure 13-15: Additional Options—Short Form

Database files with attached Format Control records in which Additional Options have been designated, display extra Options menu selections corresponding to each defined Additional Option.

When defining Additional Options, you can:

- name another form to appear when the option is selected
- continue to view the form with the attached Format Control record

In either case, the System Tray buttons are replaced with buttons appropriate to the application launched via the Additional Option and the corresponding F–keys are activated. Users move between applications or files with a single click. After activating an Additional Option, the related application or file appears with its normal Service Tray buttons, e.g., allowing users to query for records and save updates.


ServiceCenter - [Format Control user.contacts]
File Edit View Format Options List Options Window Help
OK Back Add Save Delete
Views Queries Calculations Validations Subroutines Add Options Privileges

Format Control Maintenance - Additional Options

Name: user.contacts
Form to Display:
☐ Use as Master
☐ Allow Edit
☑ Allow Input
View:

Additional Options

Option / Condition for Option
Command / Description
Application / Comments
Error Message
Reset on Return
Names / Values

Option / Condition for Option
Command / Description
Application / Comments
Error Message
Reset on Return
Names / Values

Ready | Response 0.140 draw 0.70 | insert | formatctrl.maint.options.v.g(fc.view) [US]

Figure 13-16: Additional Options—Long Form

Fields

Name

The name of the current Format Control record appears here.

Form To Display

Name an additional form you want displayed when **User Options** is selected. This form must exist in the form file. If this field is left blank, the original form continues to display.

Allow Input

Select this feature to allow the user to edit the records called by the Additional Options process. Deselect this checkbox if you want the user to have browse–only capabilities when **User Options** is selected from the Options menu. The fields of a record open only for browsing appear gray, and cannot be modified.

Use As Master

Select this feature to set the Additional Options buttons as the default buttons for this file. Whenever a user opens a record from this form, the control buttons and F–keys defined in the Format Control Additional Options appear in the System Tray.

Allow Edit

This field determines whether the user will be given the **edit record function** (**F2**), which allows an edit of the record.

Records are not initially displayed in the edit mode. To modify the record, the user must click on the **Edit** button (or press **F2**). This is similar to the **Edit Report** option in the Report Writer Utility.

Option

Assign ID numbers to the option buttons in this field. Valid numbers are from 1–12, excluding 3 and 6. (Three is reserved for **Logout** and 6 identifies the **more** function in Text mode.)

Condition For Option

The value in this field determines whether or not this button/F–key will be available to the user. This field can evaluate to *true*, *false*, or a qualifying expression. If the value is *true*, the button will be displayed to all users.

Sample expression:

```
index("OCMO", $lo.ucapex)>0
```

In this example, the capability word of *OCMO* (OCM Order Applications) would have to appear in the Execute Capabilities array of the user's Operator Record and be a valid entry in the Capability File. Users with this capability would have access to the attached option button.

Command

Enter the description of the button's function in the Command field. This label will appear on the button in the Service Tray.

For example:

Option#	Command
1	Location

This button/F–key accesses the location file.

Description

Enter a plain text description of the option's function in this field. The user will not see this description; it is for administrative reference only.

Application

Name the application to be executed if the operator selects this option.

Comments

Brief comment about the option being defined. This field is for reference only.

Error Message

Enter the message to be displayed in the Status Tray if an error occurs when executing this application.

Reset On Return

If the `$file` variable is modified in the called application, do you wish to keep the changes or reset it back to its original state before calling the other application?

Since the passed variables are on the parameter panels (and thus are local variables), the variable will contain the value it had in the called application if this function evaluates to *true*.

For example: If the value of a field in $file (the current record) was *true* before being passed to the application, and upon return to the Format Control record it has a value of *false*, the field will retain the value of *false* if the reset on return function were set to false.

Options set to *true* will exit to the Home menu rather than returning the user to the form specified in the **Format to Display** field.

Name

Enter the name of the parameter used to pass data to the application.

For example:

name

name

Value

Enter the name of the field, value or form to be passed to the application.

For example:

value

location

The **database** application would be called using *location* (value) for the name (parameter) of the form to be used to display.

Example

A user adding or updating a Contacts file might want the option of viewing the Location or Clients file without leaving the current record. The normal procedure is to access each file separately through the Database Manager QBE list. To save time, the System Administrator can attach a Format Control record to the **contacts** file establishing Additional Options functions. The user can now go directly to secondary files in any sequence and return immediately to the **contacts** file or to any other form designated in the **Form to Display** field.

We will create an Additional Options Format Control record for the **user.contacts** form that will provide button access to the Location file and the Clients file.

Opening Prompt Form

If the user has no need to return to the **contacts** file record after initiating the Additional Options process, a custom prompt form containing instructions and information can be used. This form is not interactive and serves as a center point for moving through the files defined in Additional Options.

Create the form using Forms Designer, and enter the name of the form in the **Form to Display** field in the Additional Options form. Remember: If you leave this field blank, all option files will exit to the original file. (Refer to the **Forms Designer** documentation for details on creating forms.)

Note: Any ServiceCenter form can be designated in the **Form to Display** field of Additional Options. Updating display objects such as charts or marquee can be included on your form.

Designate Files to Be Called

Use the following values in the Additional Options form to designate the files to be called:

Option #	Condition	Command	Application	Reset on Return	Names	Values
1	true	Location	database	false	name	location
2	true	Clients	database	false	name	company.g

Error Message

Cannot display requested form. Form must already exist.

Form to Display

Leave this field blank.

Note: The name of the parameter is name because the type of parameter to be passed is *character*. The value to be passed in this case is the name of the form used to view the file.

User Interaction

1 Start the Database Manager and click the Administration Mode checkbox. Note: You must be in Administration mode to use the user.contacts form, as required for this example.

2 Open the **contacts** file in Database Manager by entering user.contacts in the **Form** field.

3 A blank **user.contacts** form is displayed.

4 Click the **Search** button

5 Select a **Contact Name** from the QBE list.

6 Press **Enter.**

7 A complete data record is displayed.

8 Select **User Options** from the Options menu.

Note: This option only appears in the Options menu of forms if there is an associated Format Control record in which Additional Options have been specified.

9 If you have named a form in the **Format to Display** field of the Additional Options process, that form will appear. If you have left the field blank, the **Contacts** record appears.

10 A new set of System Tray buttons appear.

11 Click on the **Location** button to display a blank Location record.

12 Click **Back** to return to the primary (**contacts**) file.

13 Click **Clients** to display a blank Clients record.

14 Click **Back** to return to the primary (**contacts**) file.

15 Click on the directional buttons to move forward and backward through the primary file records.

This will take you out of the Additional Options mode. To return, select **User Options** from the Options menu again.

16 Click **more** (or press **F6**) to restore the normal tray buttons.

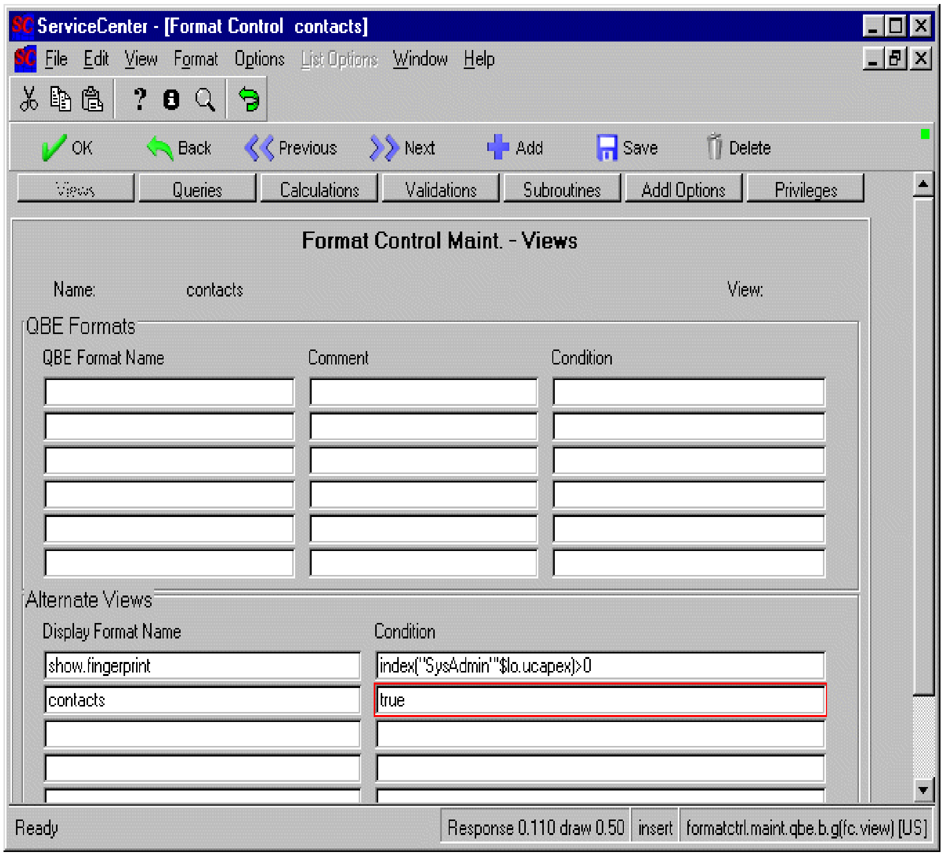
17 Click **menu** (or press **F12**) to return to the Home menu.

18 Click **Exit** from the primary file to exit the Additional Options settings and return to the QBE list

Views

The Views process allows a System Administrator to create and present alternate views of the database to different users based on their system capabilities. You can create several QBE and display forms that present different information from the same file. These forms can be selected whenever you view a QBE list or record by selecting the **Views** button. To enable the Views process, the appropriate array in the Format Control record must contain one or more form names, and the corresponding condition must evaluate to *true*. When a QBE list is displayed in Database Manager, the Format Control record for the *file* is used. Multiple QBE forms can be used to create quick data listings, without the requirement of generating a report.

To access the Views process, select **Views** from the Options menu of the Format Control Main Information form.


ServiceCenter - [Format Control contacts]
File Edit View Format Options List Options Window Help
OK Back Previous Next Add Save Delete
Views Queries Calculations Validations Subroutines Addl Options Privileges

Format Control Maint. - Views

Name: contacts

View:

QBE Formats

QBE Format Name	Comment	Condition

Alternate Views

Display Format Name	Condition
show.fingerprint	index("SysAdmin",$lo.ucapex)>0
contacts	true

Ready | Response 0.110 draw 0.50 | insert | formatctrl.maint.qbe.b.g(fc.view) [US]

Figure 13-17: Views

Fields

QBE Format Name

The name of a QBE form.

Note: It is advisable to enter the name of the default QBE form, <*filename*>.qbe, with a Condition of *true* to allow the user to return to the default list.

Comment

A brief comment about the use of this form. This field is for reference only.

Condition

A boolean statement which, if evaluated as *true* at run time, makes the named form available for viewing.

Note: If a form defined in the **QBE Format Name** field does not exist when it is requested for display, the default <*filename*>*.qbe* format will be used.

Display Format Name

The name of an alternate form.

Condition (Display)

A boolean statement which, if evaluated as *true* at run time, makes the named form available for viewing.

14 Posting

CHAPTER

Posting is the process of copying data from a source record to a target record. The purpose of Posting is to update similar fields in other records without having to open those records to modify each field. The relationship between the two records is normally based on a value defined in one (or more) of the source record input fields. Posting is the opposite of the Fill function, which copies data from the target record to the source record.

The posting function should not be used to post to problem tickets. The ticket being updated may be locked by another user and the post will not occur. In addition, the pm.eval.alerts application will create its own $file variable which conflicts with the $file variable that post.fc is using.

Fill

- Starts on a source record.
- Accesses a record in a secondary (target) file based on data in the source record (usually the contents of the input field selected by the cursor.
- Copies (fills) data from that target record back to the source record.

Posting

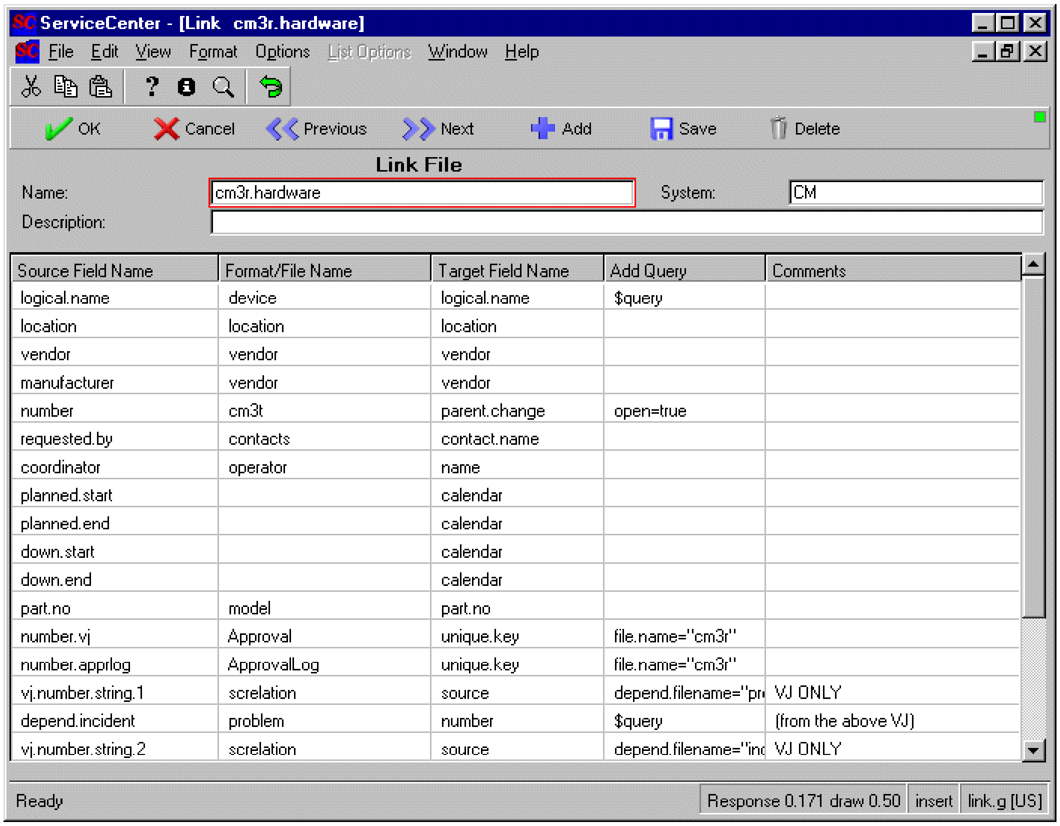
- Starts on a source record
- Accesses a record in a secondary (target) file, based on data in that record.
- Copies (posts) data from the source to the target and then performs some action (add, update, open, etc.) on the target record.

The posting process allows you to control the process in several ways:

- Define the target file.
- Select the fields to copy.
- Determine whether or not to prompt the user for add/update confirmation.
- Decide which application to invoke for the posting process.
- Define manual or automatic posting.

Link Records

The Posting process requires you to create a standard ServiceCenter Link record. The Link record establishes the connection between similar fields in the source and target files. When Posting, it is best to create a new Link record dedicated to the Posting process so as to avoid conflicts with Link records defining common **fill** relationships.


ServiceCenter - [Link cm3r.hardware]
File Edit View Format Options List Options Window Help
OK Cancel Previous Next Add Save Delete

Link File

Name:	cm3r.hardware	System:	CM
Description:			

Source Field Name	Format/File Name	Target Field Name	Add Query	Comments
logical.name	device	logical.name	$query	
location	location	location		
vendor	vendor	vendor		
manufacturer	vendor	vendor		
number	cm3t	parent.change	open=true	
requested.by	contacts	contact.name		
coordinator	operator	name		
planned.start		calendar		
planned.end		calendar		
down.start		calendar		
down.end		calendar		
part.no	model	part.no		
number.vj	Approval	unique.key	file.name="cm3r"	
number.apprlog	ApprovalLog	unique.key	file.name="cm3r"	
vj.number.string.1	screlation	source	depend.filename="pr	VJ ONLY
depend.incident	problem	number	$query	(from the above VJ)
vj.number.string.2	screlation	source	depend.filename="inc	VJ ONLY

Ready | Response 0.171 draw 0.50 | insert | link.g [US]

Figure 14-1: Link Record

Fields

When referring to fields in the source record, use the **$File** file variable. To refer to fields in the target record, use the **$Filet** file variable.

Important: The case must match that of the sample (upper case **F**– the rest lower case).

Name

This is the name of the form from which the Posting process will be initiated. The default process is for Posting to find a Link record that matches the name of the currently displayed form.

Note: When calling the Posting routine, you can override the default and define a specific Link record.

Description

Brief description of the function of the source form.

Add Query

This field defines the query that is executed against the target file in order to locate the target record. This query should be built to retrieve a unique record in the target file. If, however, more than one record is retrieved, the Posting process is completed against *all records in the list*. This same value is displayed on the Link line format.

Field Name (SOURCE)

This is the name of the input field on the form that allows the user to begin the Posting process.

Note: You can define the same input field multiple times on the Link record. If more than one field in the list matches the source input field, the Posting process is completed for all fields in the list.

Use existing input fields or define special input fields to invoke Posting to a particular file. For example, add the input fields called **post.to.problem** and **post.to.change**. Each of these input fields would then be defined to the Posting link record. This allows the user to place the cursor on either field and invoke the Posting process defined for that input field.

Consider using special input fields for the Posting process if you define a Posting link on the same format used for **find/fill** and if the same source input field defines the unique target record used for **find**, **fill** and Posting. This is because the fields that are *filled* from a target file will probably not be the same fields that are Posted to a target file.

Format/File Name (TARGET)

This is the name of the *target form* or *target file*. If this is the name of a form, data is posted to the file associated with that form. If it is not a form name, Posting attempts to post to a file with that name.

Field Name (TARGET)

This is the name of the input field in the target record that will be used to query for a unique record in the target file based on the contents of the Source Field input field. The **Add Query** field overrides this value.

Comments

This field is normally used for entering general comments about a particular Link line. This same value is displayed on the Link line form.

Link Line Definition File

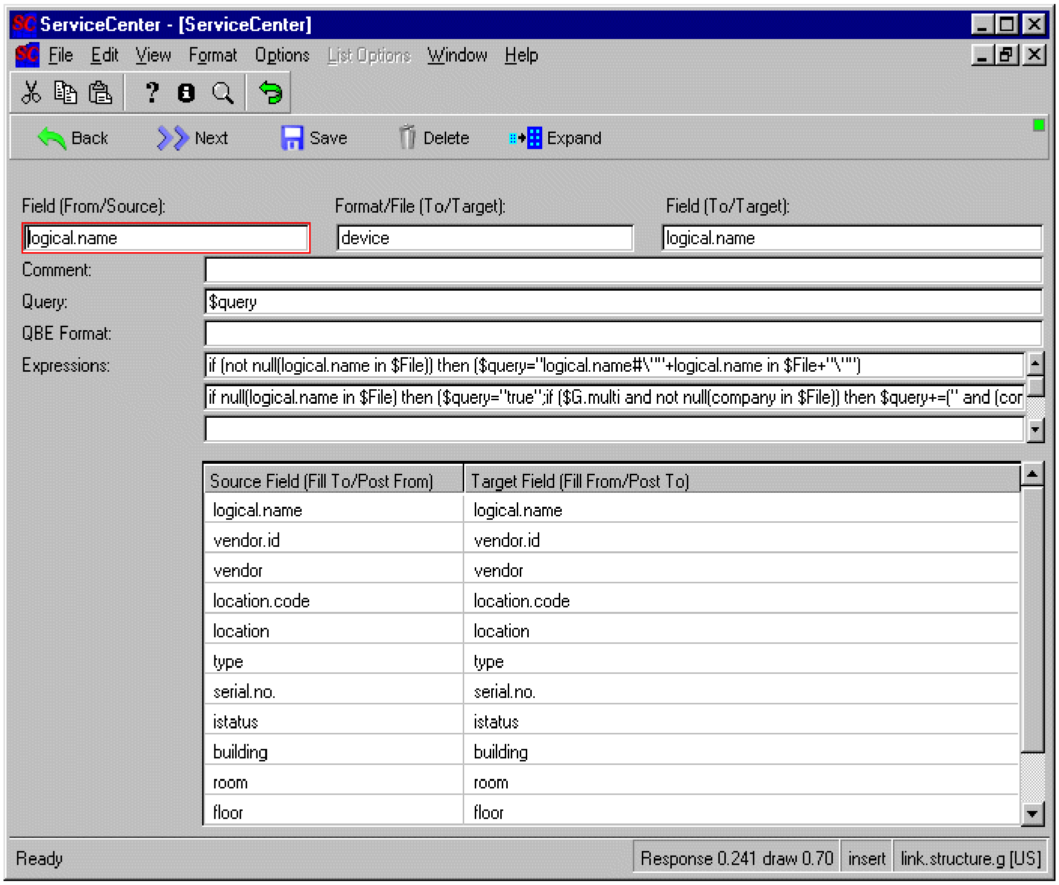
The *Link line definition file* defines a single field as the reference field for the Posting routine and lists the names of other fields linked for the Posting process. When Posting is invoked, the system will compare the value for the reference field in the source file with the value for the linked field in the target file. If the values are the same, the system compares the listed fields for changes and *posts* the new data to the target file.

The fields used for the Link line definition have the same definition/use for Posting as they do for **find** and **fill** with the following exceptions:

- The **Source Field** array lists fields in the source record and the **Target Field** array lists fields in the target record.
- You can define parameters in the **Expressions** field to control the Posting process flow. Refer to *Posting Variables* for a complete list of the variables used in this field to control Posting process.

Note: Reference or manipulate input fields with the **$File** file variable. Any modifications made to this record variable are passed back to the calling application.

Warning: Processing statements may modify the contents of the original record.


ServiceCenter - [ServiceCenter]
File Edit View Format Options List Options Window Help
Back
Next
Save
Delete
Expand

Field (From/Source): logical.name

Format/File (To/Target): device

Field (To/Target): logical.name

Comment:

Query: $query

QBE Format:

Expressions:

```
if (not null(logical.name in $File)) then ($query="logical.name#\""+logical.name in $File+"\"")
if null(logical.name in $File) then ($query="true";if ($G.multi and not null(company in $File)) then $query+=(" and (cor
```

Source Field (Fill To/Post From)	Target Field (Fill From/Post To)
logical.name	logical.name
vendor.id	vendor.id
vendor	vendor
location.code	location.code
location	location
type	type
serial.no.	serial.no.
istatus	istatus
building	building
room	room
floor	floor

Ready | Response 0.241 draw 0.70 | insert | link.structure.g [US]

Figure 14-2: Linking Individual Fields for Posting

Fields

Comment

When Posting, this field must contain the word **POST** (all upper case).

Query

(Optional). This field contains a specific Link query that overrides the standard link query.

The general rule is:

> **target field=source field in $File**

For example:

> **target1=source1 in $File**

The file variable **$File** is used for all references to the source file in **fill** and **find** operations.

QBE Format

(Optional) This field contains the name of the QBE form to be used if more than one record is selected in **find** or **fill**.

Expressions

This field contains logical or arithmetic expressions used within Link queries.

Example:

> **$post.confirm=true**

This expression instructs the system to prompt the user to confirm the Posting process.

Source Field (Fill To/Post From)

List the fields from the source record you wish to Post to the target record. Although field names do not have to match, the position of the source fields in the array must correspond to the position of the target fields receiving the data.

Target Field (Fill from/Post To)

List the fields from the target record to which the data from the corresponding source fields is to be Posted. Although field names do not have to match, the position of the target fields in the array must correspond to the position of the source fields from which data is Posted.

Posting Variables

The following variables are available during the posting process and can be used to control the Posting process flow:

$File

The source record file variable

$Filet

The target record file variable

$post.msg

The message issued to the operator when the posting routine successfully completes. The default message is: *The posting routine is complete.*

$post.confirm

A logical field that controls whether on not the user is prompted to confirm the Posting action. The default is *false*.

$post.confirm.fmt

The name of the form used to display the target record for Posting confirmation.

$post.appl

The name of the RAD routine used for posting. If an application is not defined to this variable, and the **$post.confirm** option is *true*, the confirmation is a simple **add** or **update** of the target record. No other processing is performed against the record.

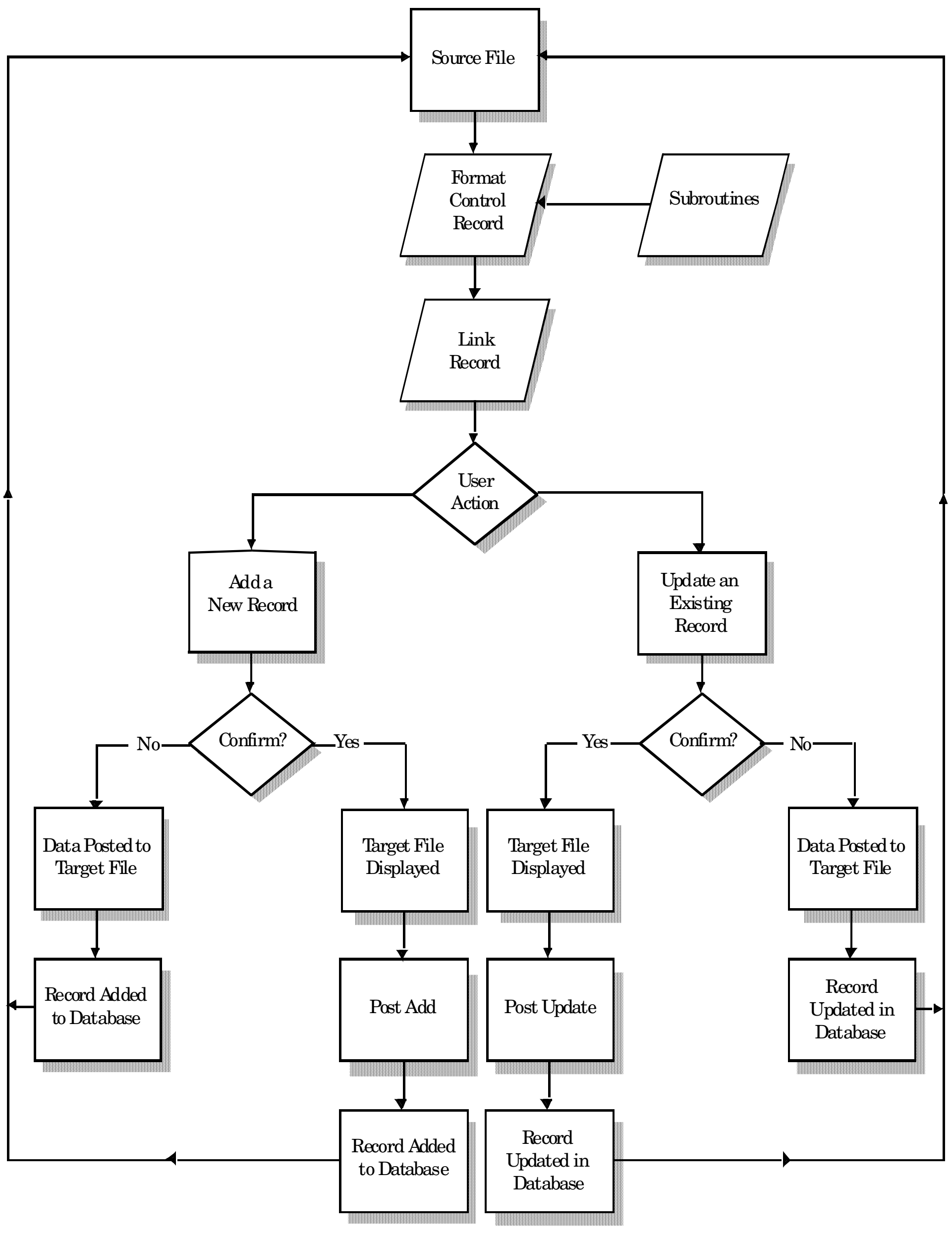
$post.names

An array that defines the names of the parameter panel input fields to which data will be passed. This field is needed only when an application is defined to the **$post.appl** parameter.

$post.values

An array that defines the values passed to the called application. There is a one-to-one relationship between this array and the **$post.names** array.

Posting Process Flow – Automatic



Setting up Posting via Format Control Subroutines or Additional Options

This section defines the field values required to invoke Posting through either Format Control Subroutines or Format Control Additional Option.

To access Format Control:

1 From System Administrator’s home menu, click the Utilities tab.

2 Click the Tools button.

3 Click Format Control.

Fields

The following fields on the Format Control record must be filled in.

Application

This is always **post.fc**.

Allow Input

(Additional Options only) Check the box to allow the user to modify fields and update records while Format Control is in effect.

Parameters

The following are the definitions for the parameters passed to **post.fc** through the **name** and **value** fields.

file

The file variable containing the source record for link processing. This will always be *$file* when calling **post.fc** through the Additional Options process. In most cases, the variable will be *$file* when calling **post.fc** through the Subroutine process; however, it could be *$file1* (or some other file variable established by Format Control or your own RAD processing).

name

(Optional) The name of the link record that defines the posting rules. This defaults to the name of the currently displayed form.

prompt

(Optional) The source field within the posting link record. This defaults to the input field name where the cursor was located when the option was selected.

Examples

There are two types of Posting available in Format Control:

- Automatic posting, setup via Format Control Subroutines
- User-controlled posting, setup via Format Control Additional Options for use with Database Manager only.

The following examples will demonstrate both types of Posting by demonstrating how to setup posting from the Change Management application and file to the *contacts* file.

Automatic Updates via Format Control Subroutines

Posting can be used to update any ServiceCenter file from another part of ServiceCenter. For example, new contact information from a change request can post directly to related fields in the *contacts* file without any user interaction. If you are opening a change request for a new contact, Posting can create a new Contact record and populate the common fields defined in the Link record. Posting from Change Management can occur when new requests are opened and when active requests are updated.

In this example, **cm3r.hardware** is our Source Format (accessing the cm3r file) and **contacts** is our Target form (accessing the contacts file). Both files contain such contact information as name, phone, and department. The relationship between the two files is based on a single common field—the *name* field in this example. When Posting is initiated from a change request, the system attempts to find a Contact record in which the value of the **contact.name** field matches the value of the **requested.by** field in the change request. If a match is found, the system updates the Contact record. If *no* match is found (as in the case of a request from a new person), the system creates a new Contact record based on the values in the change request.

Identify Field Input Values

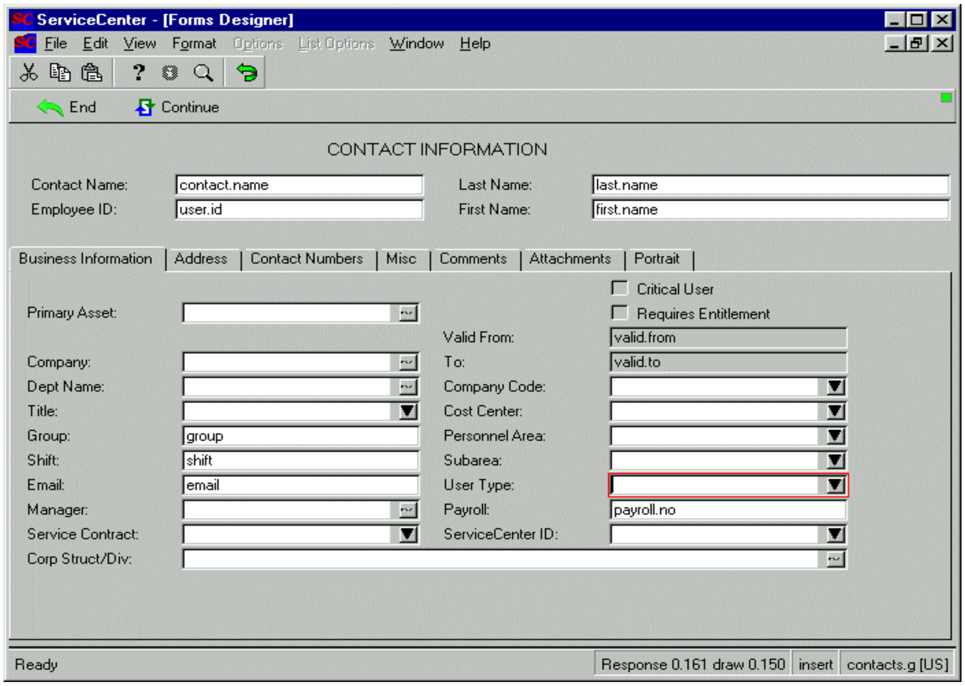
In order to create a Link record for Posting, you must identify the common fields in the Source and Target files. This can be accomplished from within Forms Designer by two methods:

- The Validate Format Option.
- Select individual fields in the Design mode.

Validate Format

1. Open form **contacts.g** in Forms Designer.
2. Select **Validate Format** from the Options menu. The input values of all the fields are displayed in the form.
3. Note the input values of the fields you wish to name in the Posting routine.

Field	**Value**
Contact Name:	contact.name
Email:	email
Work Phone:	contact.phone


ServiceCenter - [Forms Designer]
File Edit View Format Options List Options Window Help
End
Continue

CONTACT INFORMATION

Contact Name:	contact.name	Last Name:	last.name
Employee ID:	user.id	First Name:	first.name

Business Information | Address | Contact Numbers | Misc | Comments | Attachments | Portrait

☐ Critical User
☐ Requires Entitlement

Primary Asset:		Valid From:	valid.from
Company:		To:	valid.to
Dept Name:		Company Code:	
Title:		Cost Center:	
Group:	group	Personnel Area:	
Shift:	shift	Subarea:	
Email:	email	User Type:	
Manager:		Payroll:	payroll.no
Service Contract:		ServiceCenter ID:	
Corp Struct/Div:			

Ready | Response 0.161 draw 0.150 | insert | contacts.g [US]

Figure 14-3: Displaying Field Input Values in the Target File

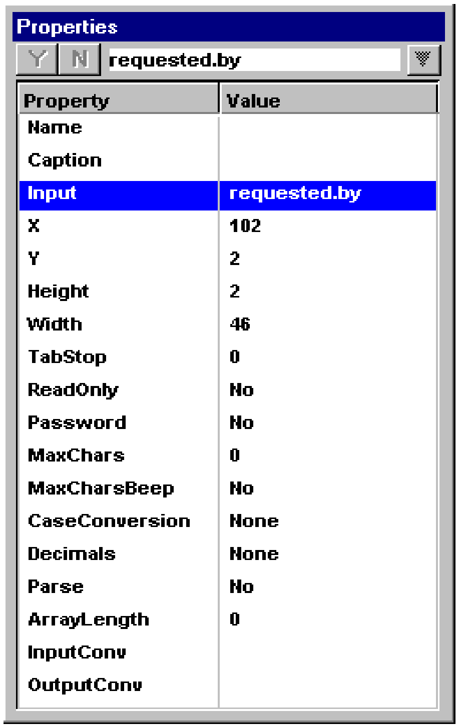
Note: The input value for a Fill box is not displayed by the Validate Format option. To see values not displayed, use the method described below.

Individual Fields

1 Open **cm3r.hardware** in Forms Designer.

2 Click on the **Design** button.

3 Select a field and check the Input value in the Properties window.



Property	Value
Name	
Caption	
Input	requested.by
X	102
Y	2
Height	2
Width	46
TabStop	0
ReadOnly	No
Password	No
MaxChars	0
MaxCharsBeep	No
CaseConversion	None
Decimals	None
Parse	No
ArrayLength	0
InputConv	
OutputConv	

Note: This method is slower than using the **Validate Format** option, but will display *all* input values, including those not shown in Fill boxes.

Important: Be certain not to alter the form when checking field input values. Click on the **Cancel** button to exit the Design mode without changing the form.

Create the Link Record

Create a new Link record for cm3r.hardware. for the Posting process. This will avoid any conflicts involving links established in a current record for the Fill function.

1 Click the **Tools** button in the Utilities tab in the administrator's home menu.

The Tools menu is displayed.

2 Click on the **Links** button.

3 Enter the name of the Link record you wish to create in the **Form** field of the Link Manager dialogue box—in this example, **cm3r.hardware.post**

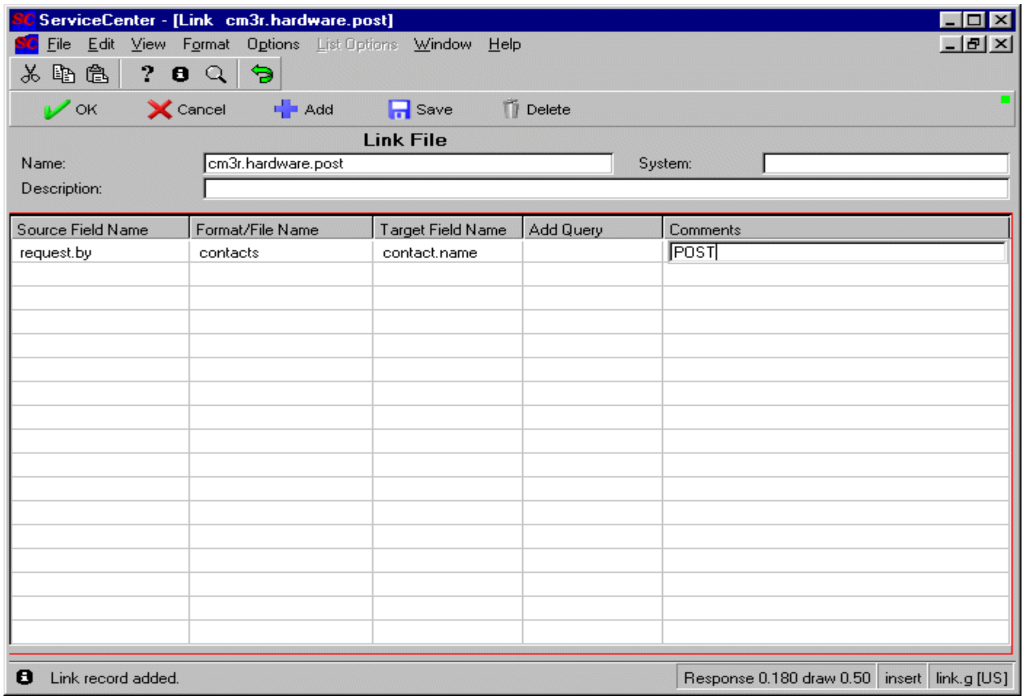
4 Click on the **New** button.

5 Enter the following values in the first line of the new Link record:

Field	Value
Field Name (SOURCE)	requested.by
Format Name (TARGET)	contacts
Field Name (TARGET)	contact.name
Comment	POST

Important: You must enter **POST** in upper case letters in the **Comment** field for the Posting process to work.

6 The new Link record will appear as follows:



Link File

Name: cm3r.hardware.post
System:
Description:

Source Field Name	Format/File Name	Target Field Name	Add Query	Comments
request.by	contacts	contact.name		POST

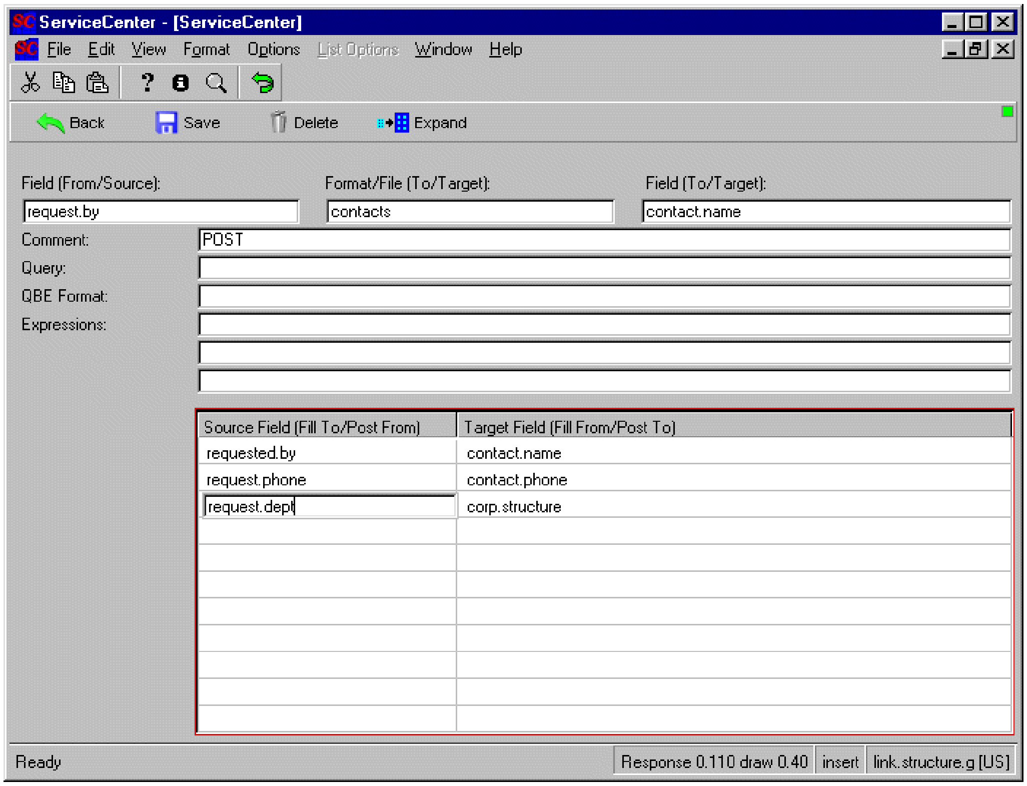
Link record added.
Response 0.180 draw 0.50 | insert | link.g [US]

Figure 14-4: Source File Link Record

7 Place the cursor in the **requested.by** line and select **Select Line** from the Options menu.

8 The Link line definition form is displayed. Enter the following values for the **Source** and **Target** fields:

Source Field	Target Field
requested.by	contact.name
request.phone	contact.phone
request.dept	corp.structure


ServiceCenter - [ServiceCenter]
File Edit View Format Options List Options Window Help
Back Save Delete Expand

Field (From/Source): request.by

Format/File (To/Target): contacts

Field (To/Target): contact.name

Comment: POST

Query:

QBE Format:

Expressions:

Source Field (Fill To/Post From)	Target Field (Fill From/Post To)
requested.by	contact.name
request.phone	contact.phone
request.dept	corp.structure

Ready | Response 0.110 draw 0.40 | insert | link.structure.g [US]

Figure 14-5: Link Line Definition Form

9 Click the **Back** button.

10 A prompt will ask if you want to save the Link record you have created.

11 Click the **Yes** button to save the record.

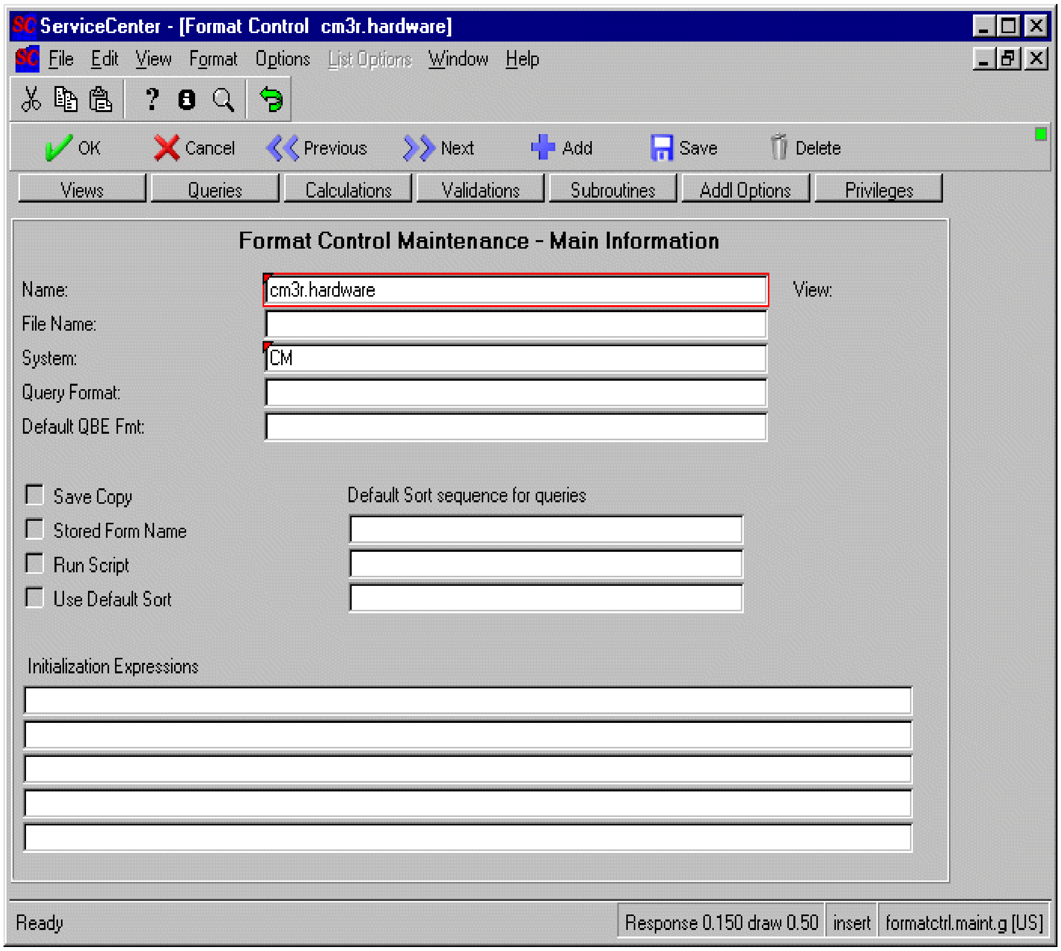
Create the Format Control Record

Automatic Posting is defined in the Subroutines process of Format Control. The application called is **post.fc**. Posting takes place before the change request is added or updated in the database.

Important: As posting may add a new record to the target file, all fields that are part of a unique or no nulls key must be filled in using the Source Field / Target Field table, otherwise posting will fail.

To create the Format Control Record:

1. Click the **Tools** button in the Utilities tab.
2. Click on the **Format Control** button.
3. Type `cm3r.hardware` in the **Name** field of the Format Control Initializations form.
4. Click on the **Search** button.
5. If no record exists, click on the **New** button.
6. The Format Control record for **cm3r.hardware** is displayed.


ServiceCenter - [Format Control cm3r.hardware]
File Edit View Format Options List Options Window Help
OK Cancel Previous Next Add Save Delete
Views Queries Calculations Validations Subroutines Addl Options Privileges
Format Control Maintenance - Main Information

Name: cm3r.hardware
View:
File Name:
System: CM
Query Format:
Default QBE Fmt:

☐ Save Copy
☐ Stored Form Name
☐ Run Script
☐ Use Default Sort

Default Sort sequence for queries

Initialization Expressions

Ready | Response 0.150 draw 0.50 | insert | formatctrl.maint.g [US]

Figure 14-6: Format Control Initializations Form

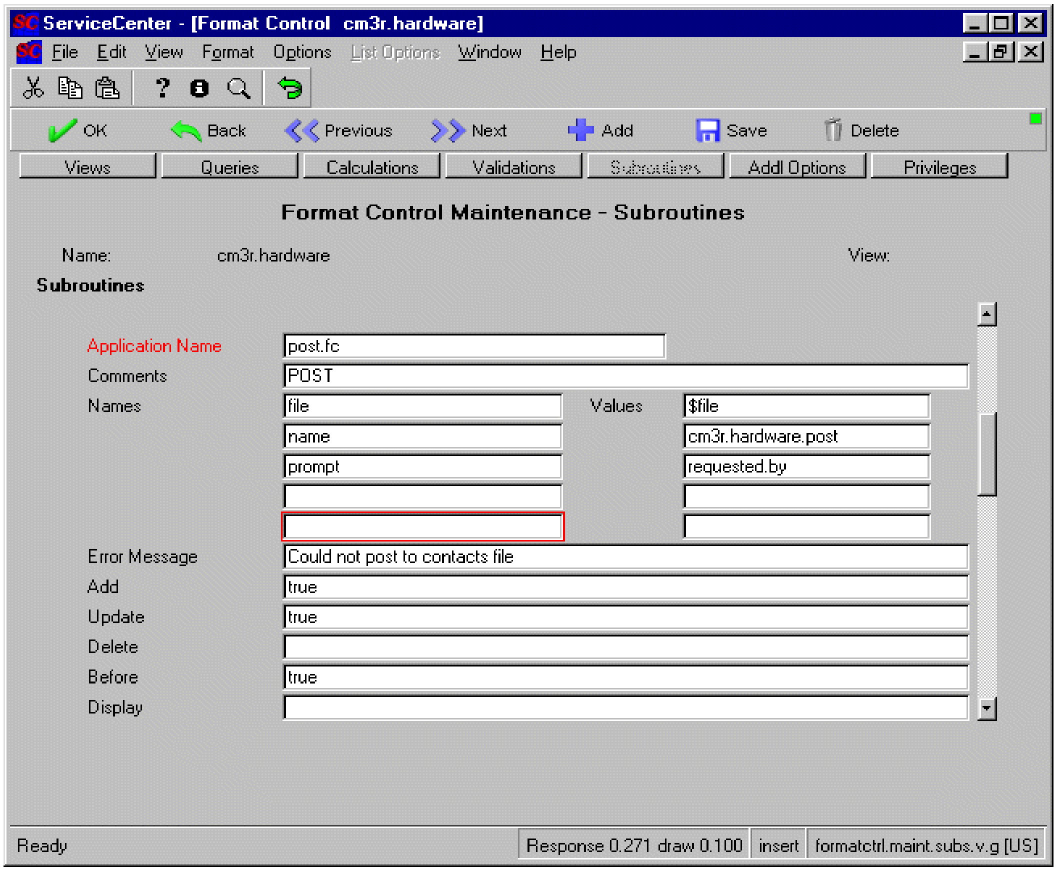
7. Select **Subroutines** from the Options menu or click the Subroutines button.

8 Select **Show Expanded Form** from the Options menu to display the long version of the Subroutines form.

9 Enter the following values in the first available slot:

Names	Values
file	$file
name	cm3r.hardware.post
prompt	requested.by
Error Message	Could not post to contacts file.

Field	Value
Application Name	post.fc
Comments	POST
Add	true
Update	true
Before	true


ServiceCenter - [Format Control cm3r.hardware]
File Edit View Format Options List Options Window Help
OK
Back
Previous
Next
Add
Save
Delete
Views
Queries
Calculations
Validations
Subroutines
Addl Options
Privileges

Format Control Maintenance - Subroutines

Name: cm3r.hardware View:

Subroutines

Application Name	post.fc		
Comments	POST		
Names	file	Values	$file
	name		cm3r.hardware.post
	prompt		requested.by
Error Message	Could not post to contacts file		
Add	true		
Update	true		
Delete			
Before	true		
Display			

Ready | Response 0.271 draw 0.100 | insert | formatctrl.maint.subs.v.g [US]

Figure 14-7: Format Control Subroutines Setup to Call Posting

Note: The name parameter passes the name of the Link record you created, and the prompt parameter passes the source field for the Link relationship.

10 Click the **Back** button.

11 A prompt will ask if you want to save the changes to the Format Control record.

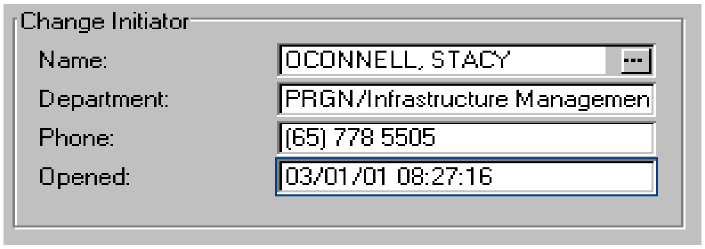
12 Click **OK** to save the record.

Open a Change Request

In order to test the Posting routine you have set up, you must open an existing hardware change request or create a new one. We will first test updating the *contacts* file by changing the values in the **Department** and **Phone** fields in the General tab of the change request. Then we will test creating a new Contact record by changing the **Name** field in the change request.

To begin:

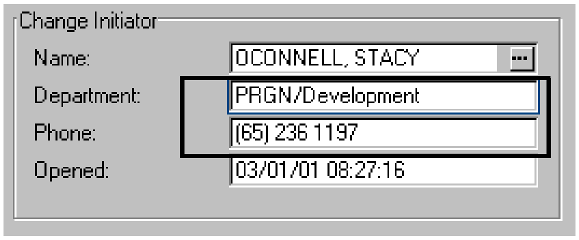
1. Click on the **Change Management** button in the Services tab in Home menu.
2. Click Open New Change in the Main tab of the Change Management menu.
3. Select the *Hardware* category for your request.
4. An initial hardware change request form is displayed.
5. Place the cursor in the **Name** field in the Change Initiator group and click on the **Fill** button.
6. Select a name from the QBE list of contacts. (The contact information displayed in the QBE list comes from the *contacts* file.)
7. The system populates the Change Initiator structure with information from the *contacts* file.


Change Initiator

Name:	OCONNELL, STACY
Department:	PRGN/Infrastructure Managemen
Phone:	(65) 778 5505
Opened:	03/01/01 08:27:16

Figure 14-8: Contact Information from a Change Request

8 Change the department designation and phone number in the change request form.

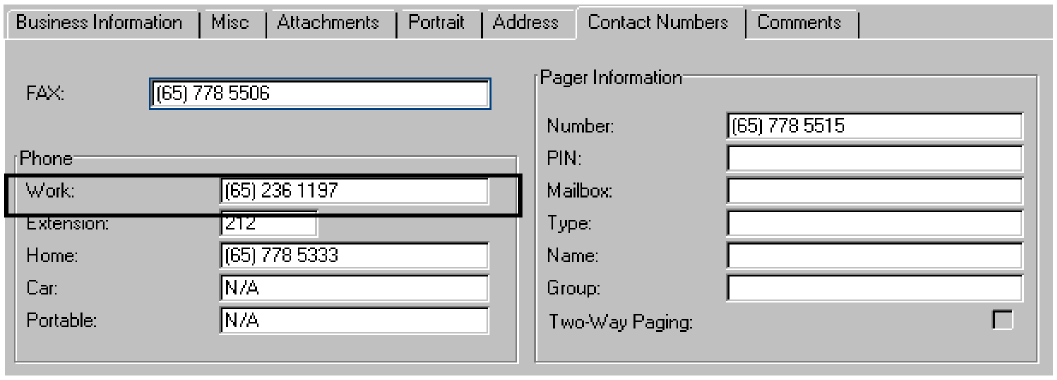


Change Initiator	
Name:	OCONNELL, STACY
Department:	PRGN/Development
Phone:	(65) 236 1197
Opened:	03/01/01 08:27:16

9 Click on the **Save** button to save the request and post the data to the *contacts* file.

10 Place the cursor in the **Name** field and click the **Find** button.

11 The Contact record displays, showing your changes. The departmental information is at the bottom of the Business Information tab and the phone number is in the Contact Numbers tab of the Contact record.



FAX:	(65) 778 5506
Phone	
Work:	(65) 236 1197
Extension:	212
Home:	(65) 778 5333
Car:	N/A
Portable:	N/A
Pager Information	
Number:	(65) 778 5515
PIN:	
Mailbox:	
Type:	
Name:	
Group:	
Two-Way Paging:	☐

Figure 14-9: Updated Contact Phone Information

12 If there are problems, click on the **Message** button to view posting messages.

Manual Posting via Format Control Additional Options

In some cases, you may want to give your users the option to manually control posting. *User Options* for Posting data are defined in the Additional Options process of Format Control and require user interaction to complete the routine.

Manual Posting is desirable whenever you do not want changes to post automatically. This is true whenever the changes may only be temporary; typically you do not want temporary changes to update permanent records. For example, if the contact is calling from a different phone, her temporary number can be added to the change request without being posted to her **contacts** record.

Additional Options are only available for Database Manager. To setup manual posting within other ServiceCenter modules, such as Incident Management or Change Management, setup a new Display option definition to call the post.fc application.

In this example, we will re-use our posting link record from Database Manager and will define Additional Options for a hardware change request form. The user will be able to click on a button to Post updates to contact information. As in the previous example, the Source File is **cm3r**, and the Target File is **contacts**.

Note: Peregrine does not recommend accessing Change Management forms and records via Database Manager. We do so in this example ONLY to demonstrate Additional Options capabilities for manual posting.

Create the Link Record

Manual posting requires a link record to control the posting rules. For our example, we can use the same link record we created for automated posting from **cm3r.hardware** to **contacts**.

Create the Format Control Record

Manual posting is defined in the Additional Options process of Format Control. User Options for Posting appear as Service Tray buttons after the post.fc (Posting) application is launched.

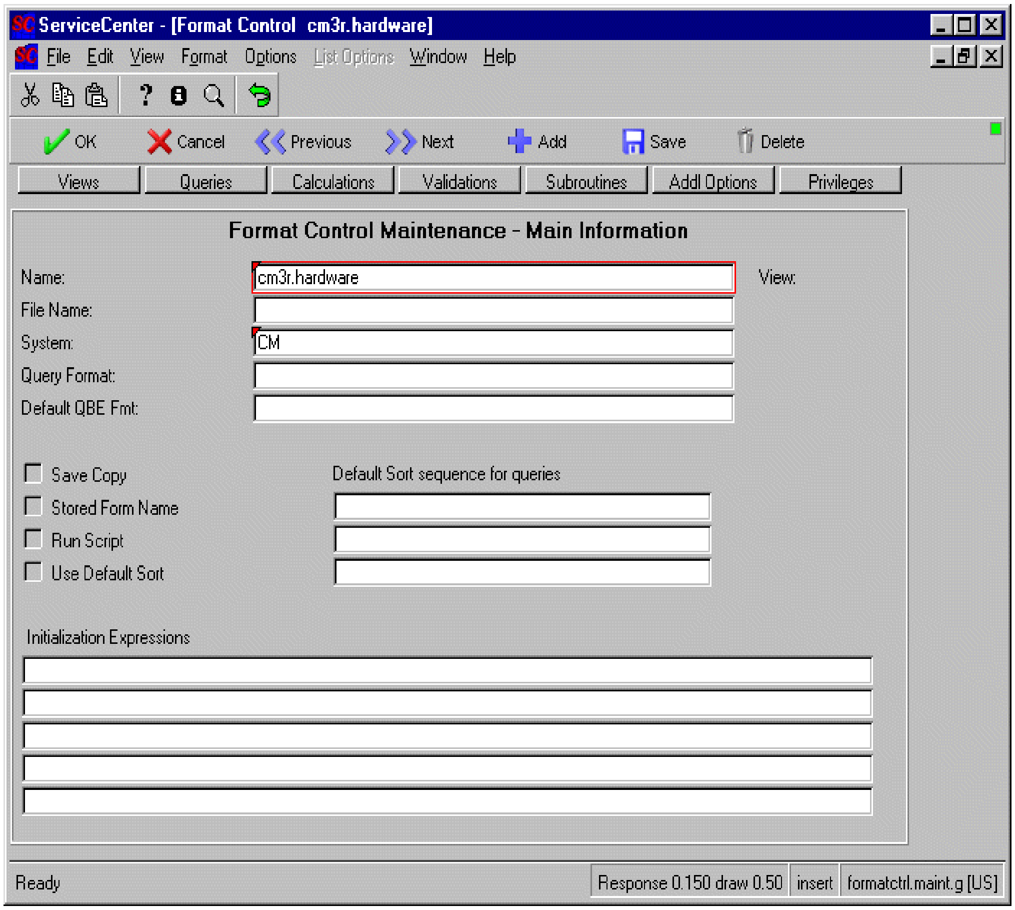
To create the Format Control Record:

1 Click the **Tools** button in the Utilities tab.

2 Click on the **Format Control** button in the Tools menu.

3 Enter the name of the form in the **Name** field of the blank Format Control record. In this example, enter `cm3r.hardware`.

4 If the Format Control record does not already exist, click on the **New** button to create a new Format Control record.



Format Control Maintenance - Main Information

Name:	cm3r.hardware
File Name:	
System:	CM
Query Format:	
Default QBE Fmt:	

View:

- [ ] Save Copy
- [ ] Stored Form Name
- [ ] Run Script
- [ ] Use Default Sort

Default Sort sequence for queries

Initialization Expressions

Ready | Response 0.150 draw 0.50 | insert | formatctrl.maint.g [US]

Figure 14-10: New Format Control Record

5 Select **Additional Options** from the Options menu or click the Additional Options button.

6 Select **Show Expanded Form** from the Options menu to display the long version of the Additional Options form.

7 Enter the following field values in the first available slot:

Field	Value
Option	1
Condition for Option	true
Command	Post to Contacts
Description	Post to Contacts
Comments	POST
Application	post.fc
Error Message	Could not post to contacts file.
Reset on Return	true

Names	Values
file	$file
name	cm3r.hardware.post
prompt	requested.by

Note: After entering the second parameter name and value (name/cm3r.hardware.post), click **Save** to save the updated Format Control record. A scroll bar appears next to the Names array so you can scroll down and enter the third Names value (prompt). To enter the third Values value (requested.by), position the cursor on the first Value entry($file) and press the PgDn key.

8 Click the **OK** button to save your changes.

Modify a Hardware Change Request

In order to test the manual Posting routine you have set up, you must access a hardware change request and modify one or more fields in the Change Initiator section that were defined in the Link record. Then, you will invoke the Additional Options to verify that the contacts file data changes to reflect the changes in your change request.

We will do this via the Database Manager Administrator mode because Format Control Additional Options are not available in Change Management.

1 Access the **Database Manager.**

2 Select the cm3r.hardware form.

3 Click Administration Mode.

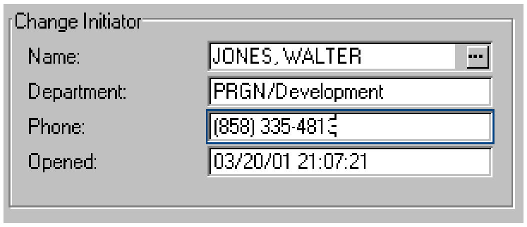
Note: You must click Administration Mode or you will be using Change Management instead of Database Manager. Format Control Additional Options are not available in Change Management.

4 Retrieve any existing Hardware change request, for example the change request you created with a new contact in the last section.

5 Verify that the existing Change Initiator matches an existing contacts file record. To do so, position the cursor on the Change Initiator Name field and click Find. You should see the related Contact Information record displayed on form contacts.g.

6 If no contacts record match exists, clear the existing Change Initiator Name field, then use the Fill button to fill a valid contact value into the change request record.

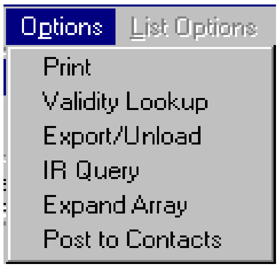
7 Update the **Department** and **Phone** fields.


Change Initiator

Name:	JONES, WALTER
Department:	PRGN/Development
Phone:	(858) 335-481
Opened:	03/20/01 21:07:21

8 Select **Post to Contacts** from the Options menu to initiate posting to the contacts file.

Note: The option name depends on the value entered in the Command field of the Additional Options definition. Therefore, you may see something other than Post to Contacts in your Options menu



9 Place the cursor in the Change Initiator **Name** field and click on the **Find** button.

10 The updated Contact record is displayed.

Note: The updated departmental data from the change request displays in the Corp. Structure field at the bottom of the Business Information tab, and the updated phone data displays in the Work phone field in the Contact Numbers tab.

Confirming the Posting Routine

A confirmation function is available in Posting that posts the new contact information to the target file, but waits for user approval before updating the record. The target file is displayed to the user, who can edit any of the information before confirming the update. For example, when a new Contact record is created by the Posting routine from Change Management, only the fields specified in the Link record are populated. At some point, the record must be completed. When the confirmation feature is active, the new Contact record is displayed to the user when Posting occurs. The user can complete the contact information form, save the record, and return to the request without leaving Change Management.

Note: This feature is available in both the automatic and the manual Posting modes.

Confirmation is activated by setting a flag called $post.confirm to *true* in the Expressions field of the source file Link line definition record.

1 Click the **Tools** button in the Utilities tab.

2 Click the **Links** button.

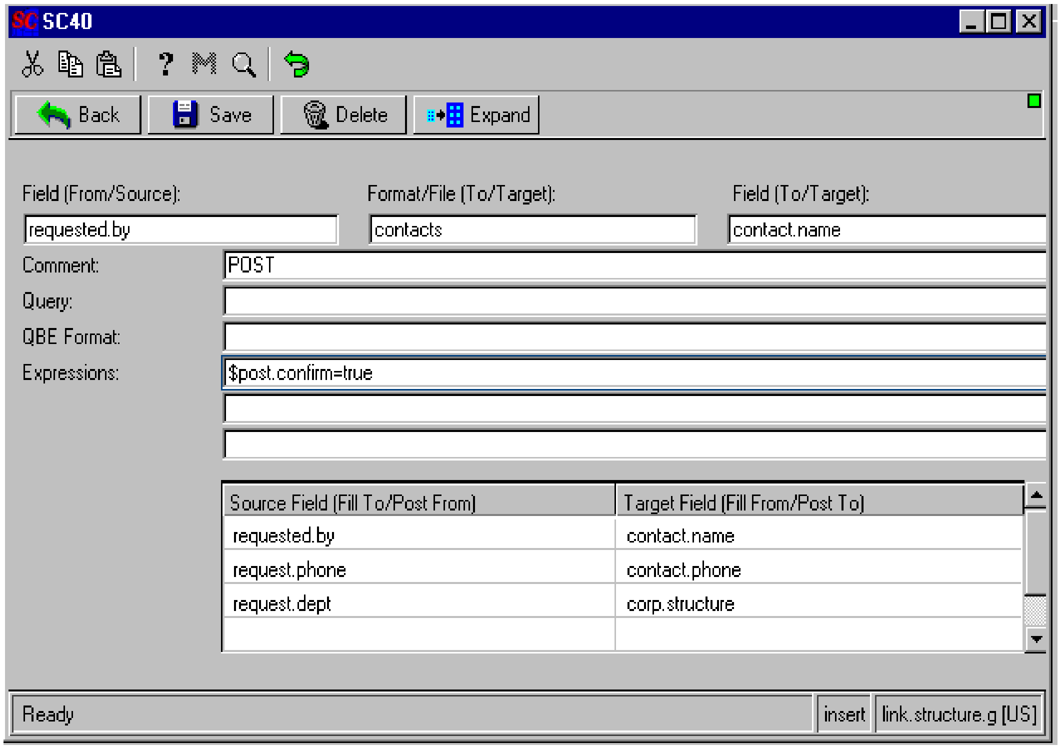
3 Enter the name of the source file Link record in the **Form** field of the Link Manager dialogue box. In this example, use the name of the Link record you created for these exercises, cm3r.hardware.post.

4 Click the **Search** button. The source file Link record is displayed.

5 Place the cursor in the **requested.by** field of the Link record and select **Select Line** from the Options menu.

6 The Link line definition form displays.

7 Enter **$post.confirm=true** in the **Expressions** field.



Field (From/Source): requested.by
Format/File (To/Target): contacts
Field (To/Target): contact.name
Comment: POST
Query:
QBE Format:
Expressions: $post.confirm=true

Source Field (Fill To/Post From)	Target Field (Fill From/Post To)
requested.by	contact.name
request.phone	contact.phone
request.dept	corp.structure

Ready | insert | link.structure.g [US]

Figure 14-11: Posting Link Line with posting confirmation

8 Click the **Back** button.

9 A prompt box appears, asking if you want to save the changes to the Link record.

10 Click **Yes**.

Manual Posting with Confirmation

Use the same procedures for evoking the confirmation process as for establishing User Options in manual posting. When you activate the Posting routine, the confirmation process displays the target form and file (contacts) and provides a new set of System Tray buttons.

1 Access the **Database Manager.**

2 Select the cm3r.hardware form.

3 Click Administration Mode.

Note: You must click Administration Mode or you will be using Change Management instead of Database Manager. Format Control Additional Options are not available in Change Management.

4 Retrieve any existing Hardware change request.

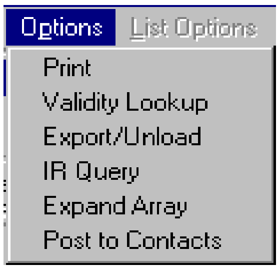
5 Verify that the existing Change Initiator matches an existing contacts file record. To do so, position the cursor on the Change Initiator Name field and click Find. You should see the related Contact Information record displayed on form contacts.g.

6 If no contacts record match exists, clear the existing Change Initiator Name field, then use the Fill button to fill a valid contact value into the change request record.

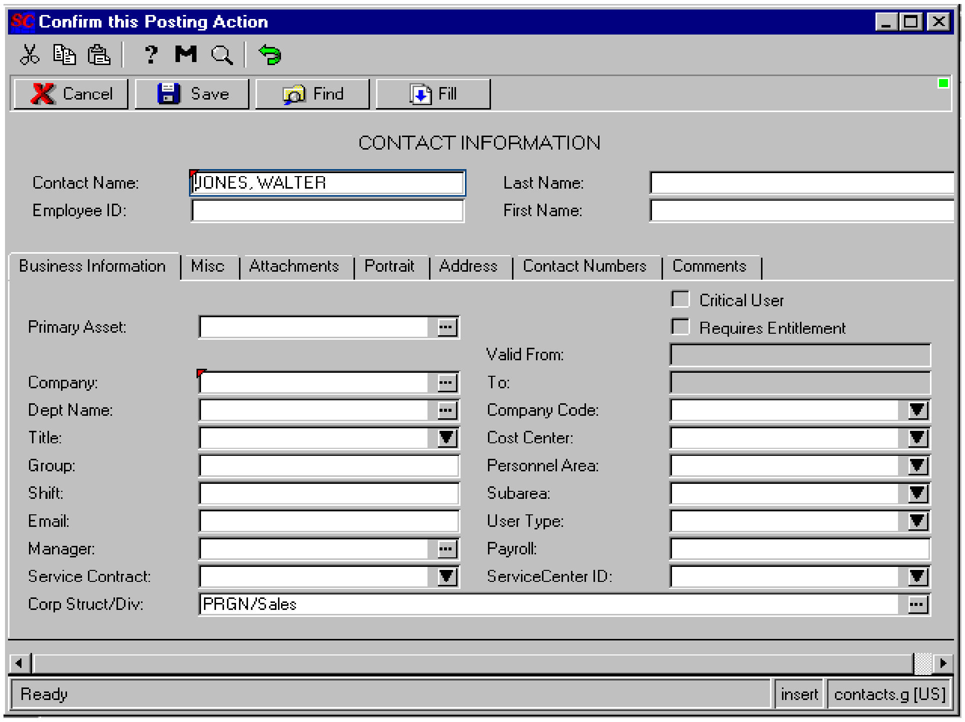
7 Update the **Department** and **Phone** fields.

8 Select **Post to Contacts** from the Options menu to initiate posting to the contacts file.

Note: The option name depends on the value entered in the Command field of the Additional Options definition. Therefore, you may see something other than Post to Contacts in your Options menu.



9 The Contact record (target file) for the contact specified in the change request form is displayed, with a confirmation request.



CONTACT INFORMATION

Contact Name:	JONES, WALTER	Last Name:	
Employee ID:		First Name:	

Critical User
Requires Entitlement

Primary Asset:		Valid From:	
Company:		To:	
Dept Name:		Company Code:	
Title:		Cost Center:	
Group:		Personnel Area:	
Shift:		Subarea:	
Email:		User Type:	
Manager:		Payroll:	
Service Contract:		ServiceCenter ID:	
Corp Struct/Div:	PRGN/Sales		

Ready | insert | contacts.g [US]

Figure 14-12: Target File with Confirmation Request

10 You may edit other fields than those selected for the Posting routine.

11 Click the **Save** button to save ALL changes, from both the change request and also changes entered directly in the contact record, to the database.

12 Click the Cancel button to ignore any changes you have made in the record. The Posting process is NOT completed and you are returned to the change request form.

15 CHAPTER

Error Messages

There are several error messages you will receive if you enter an incorrect expression into a Format Control record. These errors can take several forms:

- If any syntax errors are present when the system attempts to parse (evaluate) an expression field (e.g., Calculation, Initialization) in a Format Control record, the following message will appear in the Status Bar at the bottom of your screen when you click on the **Back** button or attempt to select another function from the Options menu:

  Field contains an invalid expression.

In the case of incorrect syntax in a Calculation expression, the system will not let you exit the Format Control record until the problem is resolved. If you cannot resolve it, delete the line and re-enter the desired input when the correct syntax is determined.

In the case of incorrect syntax in an Initialization Expression, the system will not let you access additional Format Control functions (Subroutines, Additional Options, etc.) from the Options menu.

- The system will accept a properly constructed Initialization Expression in a Format Control record that may not be accepted by the called application at run time. When an application cannot recognize an Initialization Expression, the Format Control function fails, and the following message is displayed in the Status Bar at the bottom of your screen

  Cannot evaluate initialization expression #:1

- Format Control will accept properly constructed but conflicting expressions within a record.

For example:

Initialization expression:

$x="ibm"

Calculation expression:

$x+=1

When Format Control attempts to perform this calculation on a non-numeric field at run time, the system displays the following error message:

Wrong or mismatched type in increment or decrement

16 CHAPTER Common Applications Called from Format Control

Users typically want to know which ServiceCenter applications they can call from Format Control. Since there are so many applications that can be called, listing each one is not practical; however, certain applications are called frequently or have a specific function within Format Control that should be documented. These applications and their parameters are listed below.

Also included in this chapter are the steps for determining the parameters for any application you may decide to use.

Applications

fill.fc

This application is called to copy data from a target file to the current file, using predefined links under user control.

Parameters

record

This will always be **$file** when calling **fill.fc** through the Additional Options process. This is a required parameter.

text

The name of the field in the target file to be filled. The default is the *current* field (the field in which the cursor is located). If the field is an array, the fill operation will attempt to copy data back to that item in the selected array.

string1

The form name that determines which Link record to use. The default is the Link record for the *current* form.

Variables Supported

- $fill.replace
- $project.first
- $fill.exact

post.fc

This application is called to execute the Posting routine from the Subroutines process (automatic Posting) and from the Additional Options process (manual Posting). For further information, refer to *Posting* on page 261.

Do not use post.fc to post to problem tickets. The pm.eval.alerts application will create its own $file variable which can conflict with the $file variable that post.fc is using.

Parameters

file

This will always be *$file* when calling **post.fc** through the Additional Options process. In most cases, the variable will be *$file* when calling **post.fc** through the Subroutine process; however, it could be *$file1* (or some other file variable established by Format Control or your own RAD processing).

name

This is an optional field that is defaulted to the name of the currently displayed form. You can override this default by passing a string of the specific Link record you want to use.

prompt

This is an optional field that is defaulted to the name of the input field in which the cursor was located when the option was selected. You can override this default by passing a string of a specific field name you want to use.

getnumb.fc

Call this application to establish the numbering sequence in Sequential Numbering. You may create simple identification numbers for database records or complex numbers composed of prefixes and suffixes. For further information, refer to *Sequential Numbering for Format Control* on page 187.

Parameters

record

The data record in which the number will be placed. The value of this parameter must be *$file*.

name

The sequential number Class (category) of the record to which the Format Control record is attached. This value appears in the **Class** field of the Sequential Numbering File.

prompt

Names the field in the data record in which the sequential number will appear.

text

Defines the data type of the sequential number (**number** or **string**).

number1

Defines the length of the sequential number. The subroutine will append as many zeroes to the number as necessary to match the required number of digits. Pass the RAD function **val**() to the number1 parameter. For example, a four digit number would have the value **val("4", 1)** where **4** is the length of the number and **1** is the data type (number).

string1

Defines the prefix to use. This is the identifying character string that appears before the sequential number.

query

Defines the suffix to use. This is the identifying character string that appears after the sequential number.

numbers,1

Determines the reset point for the sequential number. When the value passed to the application is reached, the sequential number is reset to its starting value.

numbers,2

Establishes a new starting value for the sequential number when the reset point has been reached.

numbers,3

Sets the increment/decrement value. For example, you may want to increment by 2, displaying odd or even numbers only.

numbers,4

Allows the user to increment or decrement the sequential number independent of the *number* file. Use this parameter to attach a special number to your records without updating the *number* file.

message.fc

This application sends messages under user control.

Parameters

index

Select one of three *message levels*:

- 1 for information (default)
- 2 for action
- 3 for error

In Text mode, message levels may appear in different colors.

prompt

Enter a *message class* that matches one of the records in the **msgclass** table (e.g., **problemclose**). To send email, there must be a **msgclass** record with a type of *email* for the message class name specified. The default is **msg**.

text

The text of the message can be either a string or an array. You can generate an array of the screen contents using the **genout**() function, for example, and insert lines of text at the top of the array.

name

This value can contain either a list of operator names of a single name. For internal ServiceCenter messages, the names must be operator ID's defined in the operator table. For email, the names can either be operator ID's or contact names (**contact.name**) defined in the contacts table; an email address must be specified in the relevant table. The default is **operator**().

string1

The *message name* parameter is used to identify the message. Use the name of the ServiceCenter application or application *area* that generates the message.

number1

The *message number* parameter is used to identify a message within the area specified by the **string1** parameter.

query

The *mail class* parameter is used within Incident Management applications to identify the ticket number so that mail already sent can be selected and updated. It must contain the string **pm.main** and a Mail Target must be defined.

names,1

The *mail target* parameter must contain the problem ticket number.

validate.fields

Call this application in the Subroutines process to validate fields in a form.

Parameters

name

Names the field in the current form to validate. For example, **logical.name**. To name the field in which the cursor was last located when an add or update were performed, use the RAD function **cursor.field.name**().

names

Name of the array in the current form to validate. The line of the array to be validated must be defined in the Initializations process.

second.file

The data record in which the field is being validated. The value of this parameter must be *$file*.

cond.input

The required value of the field defined by the parameter **val**(). For example, to validate a boolean field to true, use the value **val**("**true**", 4).

Example

ValidateNameValue

all fieldssecond.file$file

a specific fieldsecond.file$file

name<*field name*>

a range of fieldssecond.file$file

names$array (defined in Initializations

database

Call this application to access other tables from a particular record. For example, you might want an option button to appear on your users' screens allowing them to call the *location* file from the *contacts* file.

Important: This application is called *only* from Additional Options in Format Control.

Parameters

name

The name of the form you want to display to a user.

query.stored

This application is called to access all queries for a specific file or to execute a specific stored query.

Important: This application is called *only* from Additional Options in Format Control.

Parameters

name

Name of the file you want the user to be able to query. For example, **device.workstation**.

text

Name of the stored query you want to execute. For example, **name=operator()**.

axces.page

This application builds the eventout record used by the Telemon **axces** interface. The pager group, type, phone number, and PIN are retrieved from the **contacts** file.

Parameters

name

Required field containing the Contact Name from the **contacts** file.

prompt

The *numeric message*. This is a required field if the text parameter is *not* used.

text

The *alphanumeric message*. This is a required field if the prompt parameter is *not* used.

string1

This parameter defines the *separation character*. If you define your own character, be sure it does not occur naturally in fields in the event. The default is ^.

query

This parameter defines the *page response code*, used by the **pageresp** input event to identify the type of event processing that should occur. For example, to update a particular problem with the response from a page, pass **pm** and the problem ticket number (e.g., **pm9700123**). The registration record determines the application to call by examining the data in the first position of the **evfields** field.

values

This parameter identifies a list of addressees from the contacts file or operator file.

names,1

This parameter lists the pager phone number from the *contacts* file.

names,2

This parameter lists the pager PIN number from the *contacts* file.

names,3

Use this parameter to pass the name of a group defined in the *distgroup* file.

axces.write

This application is called to build an **eventout** record used by the SCAutomate interface

Parameters

record

The parameter names the record to be written. In Format Control this will be **$file**. This is a required parameter.

name

This parameter names the *registration type* as it appears in the **eventregister** file. For example, to write an **eventout** record when an incident is opened, use the value **pmo** for this parameter.

string1

This parameter defines the *separation character*. If you define your own character, be sure it does not occur naturally in fields in the event. The default is ^.

text

This parameter defines a *system sequence ID* with a maximum length of 16 characters. ServiceCenter will generate the system sequence ID unless you supply one.

prompt

This parameter defines a *user sequence ID* with a maximum length of 16 characters.

query

This parameter defines the *user name*. The default is **operator**().

axces.fax

This application is called from Format Control to build an **eventout** record used by the Replix FAX axces interface.

Parameters

names,1

This parameter passes the name of the sender. Use your login ID.

name

This parameter passes the name of the recipient. Use the recipient's login ID.

prompt

This parameter defines the recipient's FAX phone number. For example, **fax.phone** from the *contacts* file.

string1

This parameter defines the *separation character*. If you define your own character, be sure it does not occur naturally in fields in the event. The default is ^.

text

This parameter names the form or text string. If a record variable is passed in the record parameter, pass the form name in the text parameter. If you pass a string in the text parameter, use the pipe symbol (|) to separate line of text.

names,2

This parameter defines the FAX title.

names,3

The destination FAX phone number.

record

This parameter passes the record variable. Use **$file** in Format Control.

fingerprint

This application automatically records the names of users adding or updating records of *fingerprinted* files and the time each action is performed.

Parameter

record

Refers to the current file to which the Format Control record is attached. In Format Control this parameter will always be **$file**.

name

The name of the array.

index

Defines the maximum length of the array used to display the fingerprinting data expressed by the function **val**(). For example, a value of **val**(**"10," 1**) returns an array listing the last ten updates.

string1

Attaches a one word description of the action taken to the end of the fingerprint data string. For example, a **string**1 value of **Updated**, adds the word *Updated* to the user name and date appearing in the fingerprinting array.

sort.array

This application is called from Subroutines to sort simple arrays (number, character, date/time) in either ascending or descending order.

Parameters

file

When calling **sort.array** from Format Control, always pass **$file** to this parameter.

name

The name of the input field (array) in **$file** you wish to sort.

boolean1

Controls the sort order. A value of true sorts in ascending order, and a value of false sorts in descending order. The value is expressed as **val**(). For example **val**(**"false"**, 4).

Determining Parameters

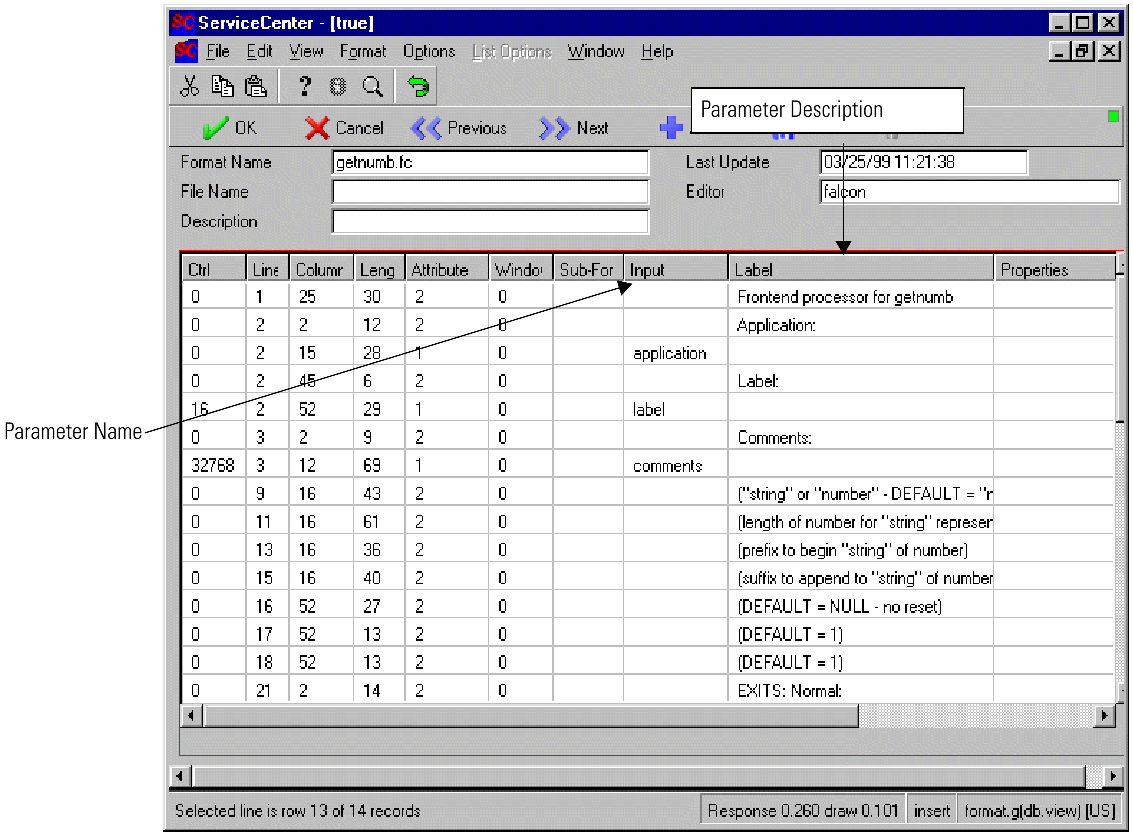
The parameters for any application in ServiceCenter can be determined without using the RAD Editor. Use Database Manager to display the input value of each field in a particular application parameter panel. This value is the parameter name used in the Subroutines process of Format Control.

1 Select the **Toolkit** tab in the Home menu.
2 Click on the **Database Manager** button.
3 Type format in the **Form** field of the Database Manager dialogue box.
4 Press **Enter** or click on the **Search** button.
5 Select **Format** from the QBE list displayed.

6 Enter the name of the application whose parameter values you want to view (getnumb.fc in this example) in the **Format Name** field of the blank **Format** form.

7 Press **Enter** or click on the **Search** button.

The parameter Name appears in the **Input** field of the form and the function appears in the **Label** field.



Format Name	getnumb.fc	Last Update	03/25/99 11:21:38
File Name		Editor	falcon
Description			

Ctrl	Line	Column	Leng	Attribute	Window	Sub-For	Input	Label	Properties
0	1	25	30	2	0			Frontend processor for getnumb	
0	2	2	12	2	0			Application:	
0	2	15	28	1	0		application		
0	2	45	6	2	0			Label:	
16	2	52	29	1	0		label		
0	3	2	9	2	0			Comments:	
32768	3	12	69	1	0		comments		
0	9	16	43	2	0			("string" or "number" - DEFAULT = "r	
0	11	16	61	2	0			(length of number for "string" represer	
0	13	16	36	2	0			(prefix to begin "string" of number)	
0	15	16	40	2	0			(suffix to append to "string" of number	
0	16	52	27	2	0			(DEFAULT = NULL - no reset)	
0	17	52	13	2	0			(DEFAULT = 1)	
0	18	52	13	2	0			(DEFAULT = 1)	
0	21	2	14	2	0			EXITS: Normal:	

Selected line is row 13 of 14 records | Response 0.260 draw 0.101 | insert | format.g(db.view) [US]

8 Scroll down to view all the parameters listed.

9 Click on the **Back** button and exit this record without saving it.

Warning: Be sure to exit the record without altering any information.

17 CHAPTER Publish and Subscribe

Virtually any data generated by ServiceCenter can be displayed to users in a one-way presentation. End users, from first level support personnel to administrators, need current, updated information about changing conditions across the system. Publish and Subscribe allows a system administrator to select pertinent data and display it to user groups in a visual format.

Automatic Updates

Display objects subscribing to ServiceCenter data are automatically updated with current information in intervals defined by the system administrator. Rapidly changing features and slowly developing trends can be expressed with the same process.

No User Interaction Necessary

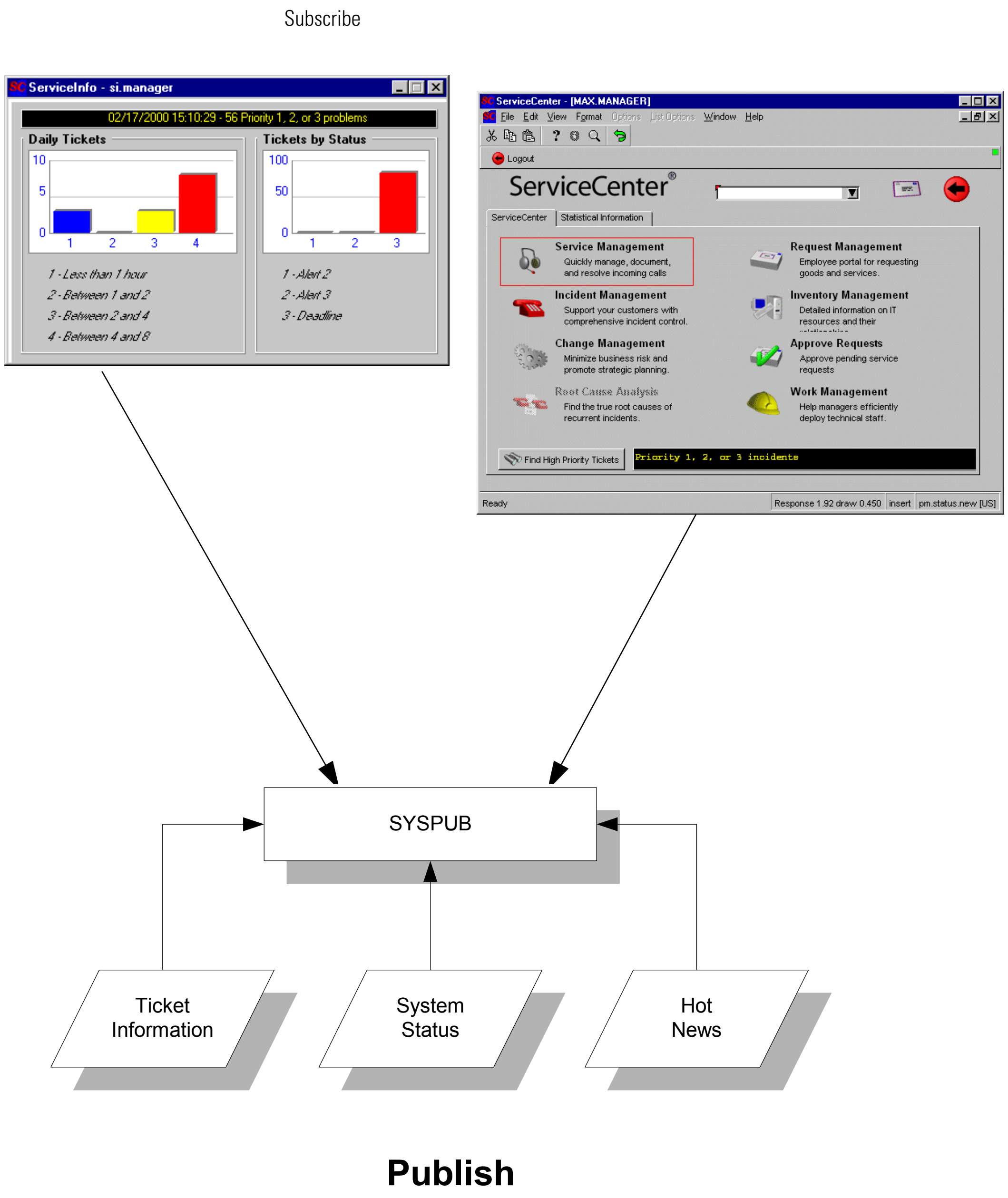
Forms created for specific user groups (e.g., Change Management personnel or Incident Management administrators) provide the end user with dynamic system data that requires no user interaction to extract. The system administrator chooses the data to be published and defines display attributes such as chart values, marquee messages, and color.

Reduced System Load

Publish and Subscribe saves system resources by eliminating the need for individual users to query the system every time they want an update. Desired system data is gathered by automatic background processes at prescribed intervals and published to subscriber groups.

Workflow

- Information is *published* in ServiceCenter when system data is associated with selected variables and stored in **SYSPUB** (ServiceCenter's *publishing house*). Information stored here is available to subscribers designated by the system administrator.
- Forms designed by the system administrator for user groups *subscribe* to the data by using published variables to populate *display objects* such as charts and marquees.
- Charts display data sets in bar graph formats. Charts show such things as ticket activity for different categories, assignment groups, time periods, and priorities. Charts can be added to a form and updated automatically at scheduled intervals. Colors can be set to reflect priorities, and button bars can be added, providing drill–down capabilities for viewing underlying data such as individual tickets within categories.
- Marquees display messages in horizontal, scrolling banners on a user's screen. This is an effective method of keeping users informed of the status of the system. Marquees are added to a form in Forms Designer and set by variables to subscribe to a particular message. They may be updated automatically at scheduled intervals.



02/17/2000 15:10:29 - 56 Priority 1, 2, or 3 problems

Daily Tickets: 1 - Less than 1 hour; 2 - Between 1 and 2; 3 - Between 2 and 4; 4 - Between 4 and 8

Tickets by Status: 1 - Alert 2; 2 - Alert 3; 3 - Deadline

Service Management: Quickly manage, document, and resolve incoming calls

Request Management: Employee portal for requesting goods and services.

Incident Management: Support your customers with comprehensive incident control.

Inventory Management: Detailed information on IT resources and their

Change Management: Minimize business risk and promote strategic planning.

Approve Requests: Approve pending service requests

Root Cause Analysis: Find the true root causes of recurrent incidents.

Work Management: Help managers efficiently deploy technical staff.

Priority 1, 2, or 3 incidents

Ready | Response 1.92 draw 0.450 | insert | pm.status.new [US]

18

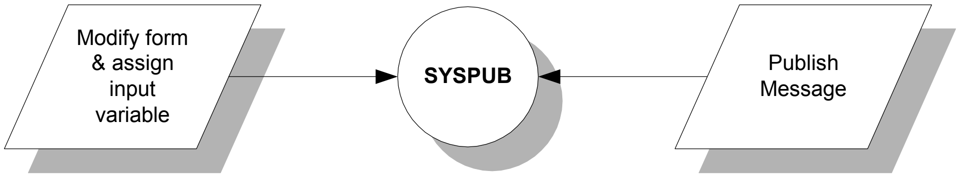
CHAPTER

Static Messages

Simple, static messages and marquees can be published to any form in ServiceCenter. For example, you may want to attach a static message to a menu advising a certain user group of the average time for resolution of an open ticket. The text of a static message does not change until it is republished in the Service Center publishing house. This is the simplest method of publishing and subscribing.

Simple Messages

This process publishes a non–updating, text message to a user's form. The message remains unchanged until edited manually.



Modifying the Form

1 Select the **Toolkit** tab in the system administrator's home menu.

2 Click **Forms Designer**.

The Forms Designer dialog box is displayed

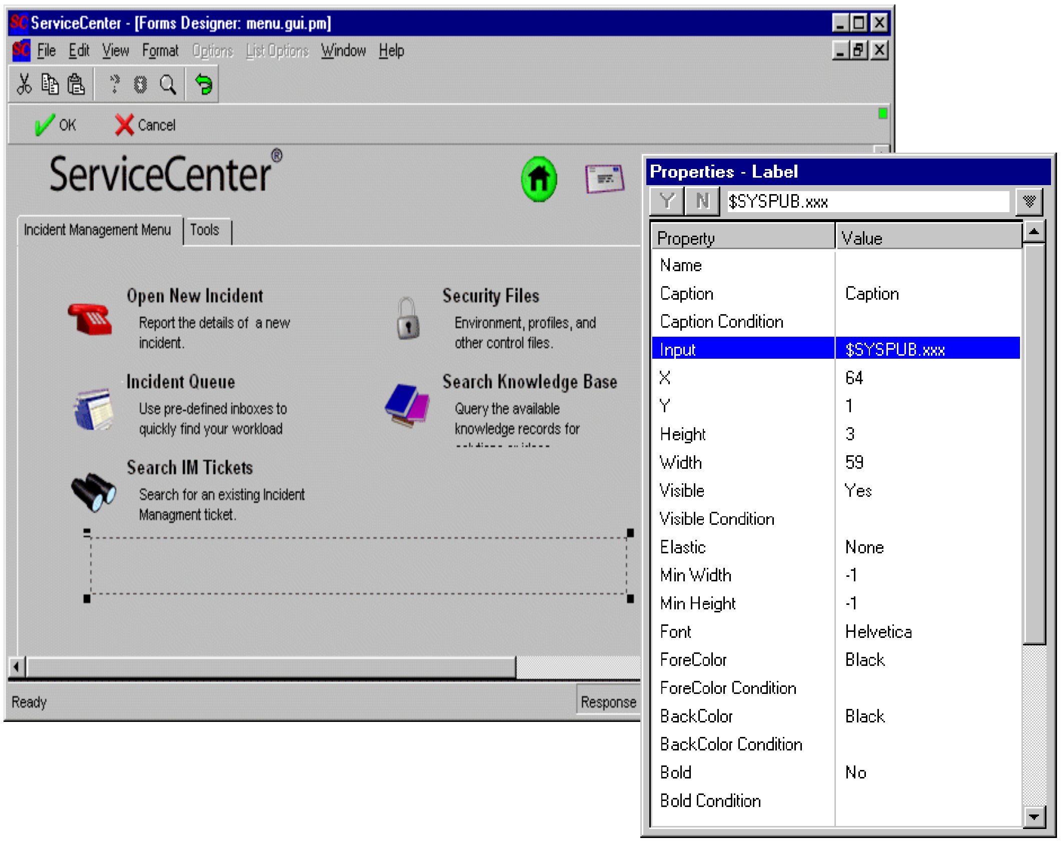
3 Type the name of the form (menu.gui.pm) in the **Form** field of the Forms Designer dialogue box.

4 Press **Enter** or click **Search**.

The Incident Management menu is displayed.

5 Click **Design** to display the Properties Box and the Tool Palette.

6 Use the **Label** tool to add a new label to the form. (Be sure the length of the label field is long enough to accommodate your message.)


ServiceCenter - [Forms Designer: menu.gui.pm]
File Edit View Format Options List Options Window Help
OK
Cancel
ServiceCenter®
Incident Management Menu
Tools
Open New Incident
Report the details of a new incident.
Security Files
Environment, profiles, and other control files.
Incident Queue
Use pre-defined inboxes to quickly find your workload
Search Knowledge Base
Query the available knowledge records for
Search IM Tickets
Search for an existing Incident Managment ticket.
Ready
Response
Properties - Label
Y N $SYSPUB.xxx

Property	Value
Name	
Caption	Caption
Caption Condition	
Input	$SYSPUB.xxx
X	64
Y	1
Height	3
Width	59
Visible	Yes
Visible Condition	
Elastic	None
Min Width	-1
Min Height	-1
Font	Helvetica
ForeColor	Black
ForeColor Condition	
BackColor	Black
BackColor Condition	
Bold	No
Bold Condition	

7 In the Properties Box, erase **Caption**. (Be sure to leave this field blank.)

8 Set **Input** to $SYSPUB.xxx (where xxx are your initials). The **Label** field appears as an empty box outlined with a dotted line in the Design mode.

9 Click **OK** to save your changes. The label field is invisible in this mode.

Publishing the Message

You must now publish the message to the ServiceCenter publishing house (SYSPUB) by setting the proper variable.

1 Select the **Utilities** tab in the system administrator's home menu.

2 Click **Tools.**

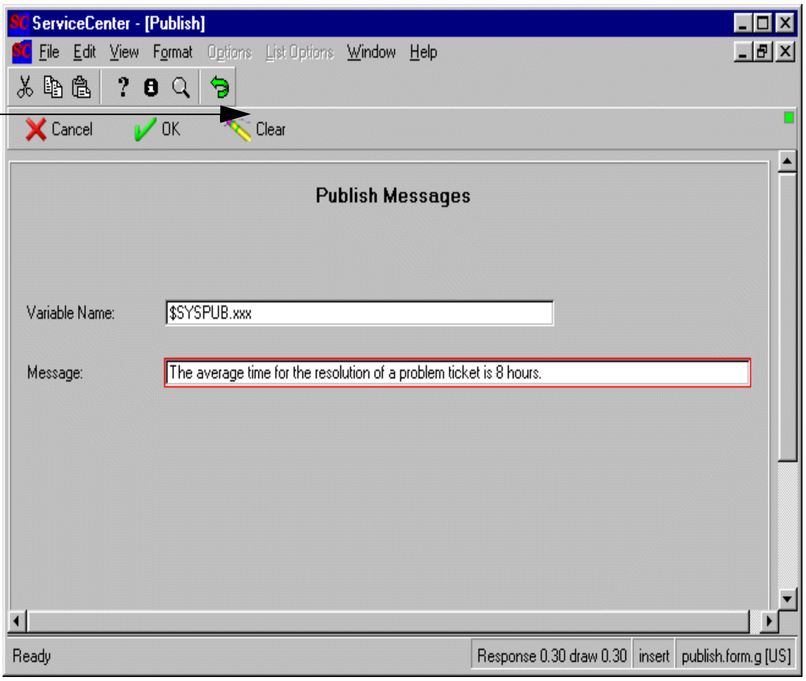
3 Select the **Publishing Utilities** tab.

4 The Publishing Utilities menu is displayed.

5 Click **Publish Messages.**

6 Enter the name of the variable from the **Input** field of the Forms Designer Properties Box (**$SYSPUB.*xxx***) and the text of your message in the field provided. (In this case, enter: The average time for the resolution of an incident ticket is 8 hours.)

The Clear button clears the fields on the publish form. It does NOT delete the message itself. See "Deleting the Message" for more information.



Variable Name: $SYSPUB.xxx

Message: The average time for the resolution of a problem ticket is 8 hours.

Figure 18-1: Publishing your message

7 Click **OK** to publish your message to the ServiceCenter publishing house.

The following message is displayed in the status bar: *Your message has been published.*

8 Click **Cancel** to return to the Publishing Utility menu.

9 Click **Back** to return to the system administrator's home menu.

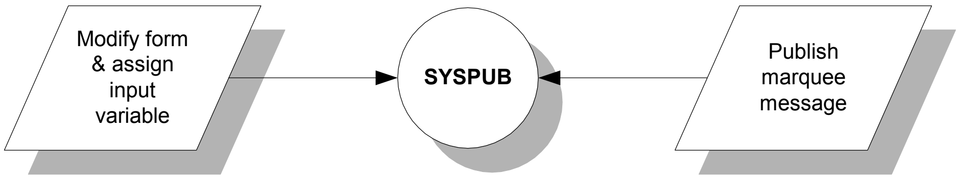
10 Select the Services tab and click **Incident Management** to check your message in the Incident Management menu.

Deleting the Message

SYSPUB messages cannot be deleted. You must edit the label property on the format you modified to show the SYSPUB message in order to remove the message.

Simple Marquees

This process publishes a non–updating marquee to a user's form. The marquee message remains unchanged until edited manually.



Modifying the Form

1 Select the Toolkit tab in the system administrator's home menu.

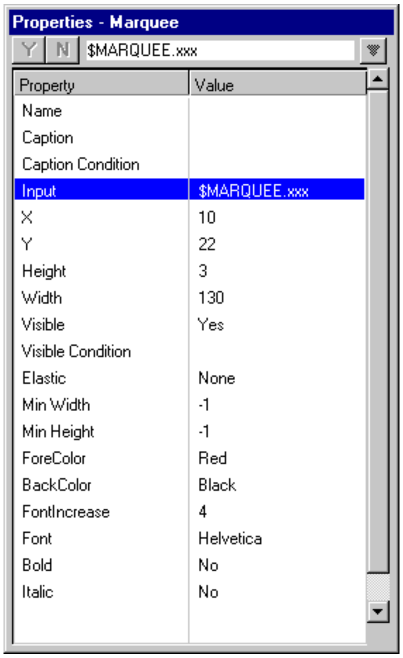
2 Click **Forms Designer**.

3 Enter `menu.gui.pm` in the **Form** field of the Forms Designer dialogue box.

4 Click the **Design** button to display the Properties Box and the Tool Palette.

5 Add a Marquee to the form with the Marquee Tool.

6 Set the **Input** value to $MARQUEE.xxx (where xxx are your initials).



Property	Value
Name	
Caption	
Caption Condition	
Input	$MARQUEE.xxx
X	10
Y	22
Height	3
Width	130
Visible	Yes
Visible Condition	
Elastic	None
Min Width	-1
Min Height	-1
ForeColor	Red
BackColor	Black
FontIncrease	4
Font	Helvetica
Bold	No
Italic	No

7 Click **OK** to save your changes.

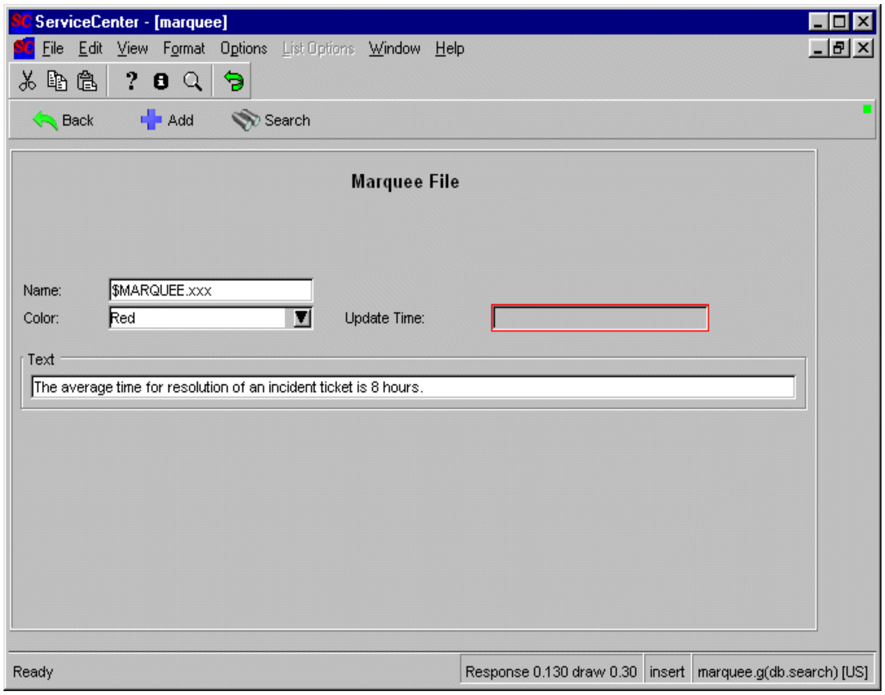
Publishing the Marquee

1 Access the Publishing Utilities menu.

2 Click **Modify Marquees.**

The marquee publishing form is displayed.

3 Enter the variable from the **Input** field of the Forms Designer Properties Box ($MARQUEE.xxx) in the **Name** field.


ServiceCenter - [marquee]
File Edit View Format Options List Options Window Help
Back Add Search

Marquee File

Name: $MARQUEE.xxx
Color: Red
Update Time:
Text
The average time for resolution of an incident ticket is 8 hours.

Ready | Response 0.130 draw 0.30 | insert | marquee.g(db.search) [US]

Figure 18-2: Publishing your marquee message

4 Select a display color from the drop-down list in the **Color** field.

5 Type in the text of your message in the **Text** field. For example, you might enter, The average time for the resolution of an incident ticket is 8 hours.

6 Click **Add** to add your marquee to the Marquee Table. (The marquee will be published by the marquee background agent.)

ServiceCenter automatically adds the date and time of day to the **Update Time** field.

The following message is displayed in the status bar: *Record added to the marquee file.*

7 Return to the system administrator's home menu.

8 Click **Incident Management** to check the marquee you have created.

You may have to wait a minute until the background agent wakes up and publishes the data.

19 CHAPTER Management View of a System

Publish and Subscribe can be used to display dynamic data that reflect changing conditions across your network. You can publish color coded marquees advising subscribers of escalating tickets, or create charts that display ticket status by category, assignment, priority, etc. This data is an extremely powerful tool for providing a good overview of your system and the status of all the pieces.

Since each group of end users will have different system capabilities, forms must be designed to provide users with appropriate system access and an overview of incidents that occur. Startup screens will communicate different data to different users as determined by their operator records.

To demonstrate the dynamic displays of a management view, we will publish messages regarding incident ticket status and ticket categories, then create a simple form with updatable marquees subscribing to the status messages and a chart showing open tickets by category. Automatic status updates will be sent to these objects in regularly scheduled intervals.

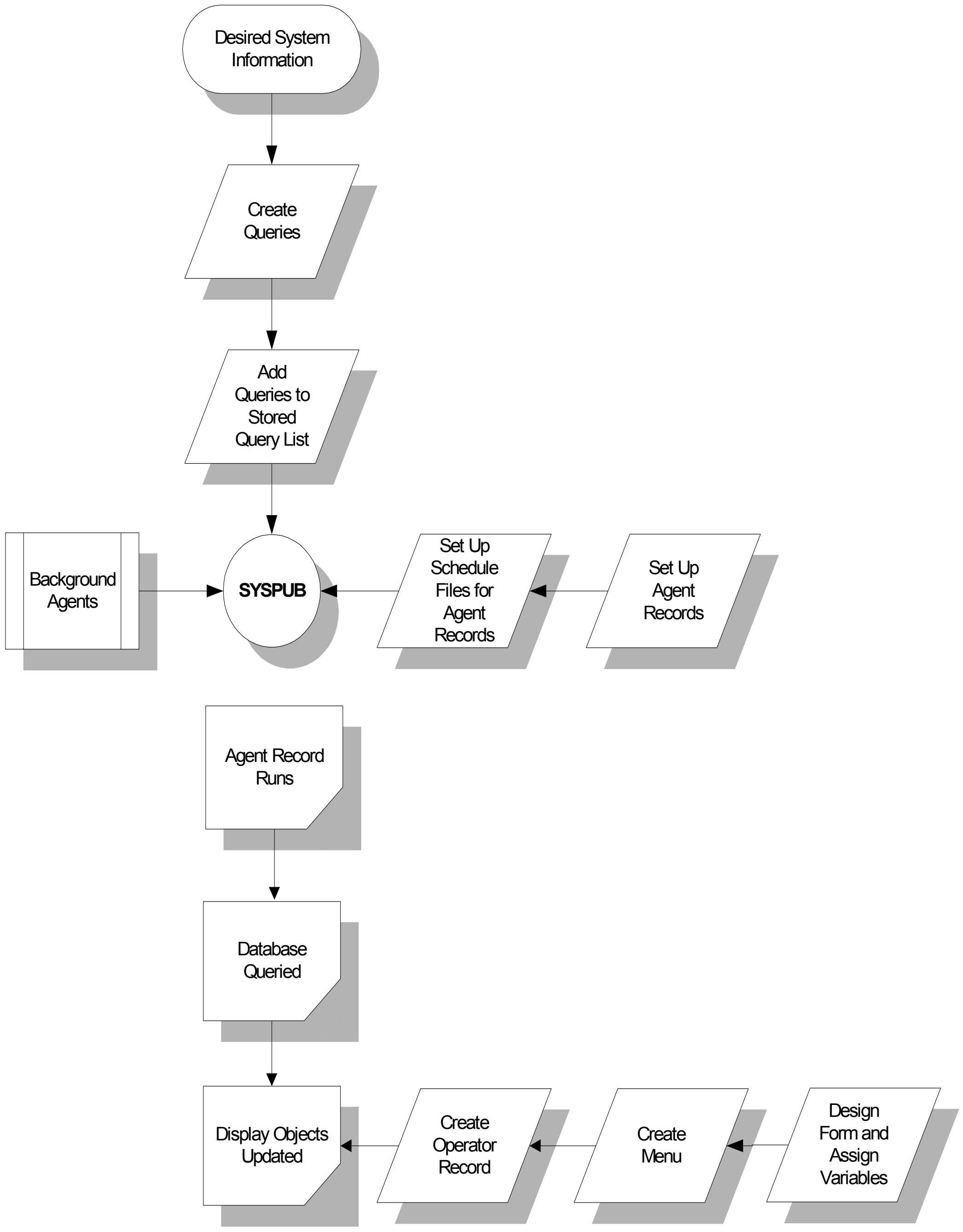
Dynamic Display Options

Management Views in ServiceCenter are used to:

- *Enhance* existing forms
- Create an *option form* accessible by certain users from within ServiceCenter
- Create a *startup screen* for a group of users
- Create *ServiceInfo* forms.

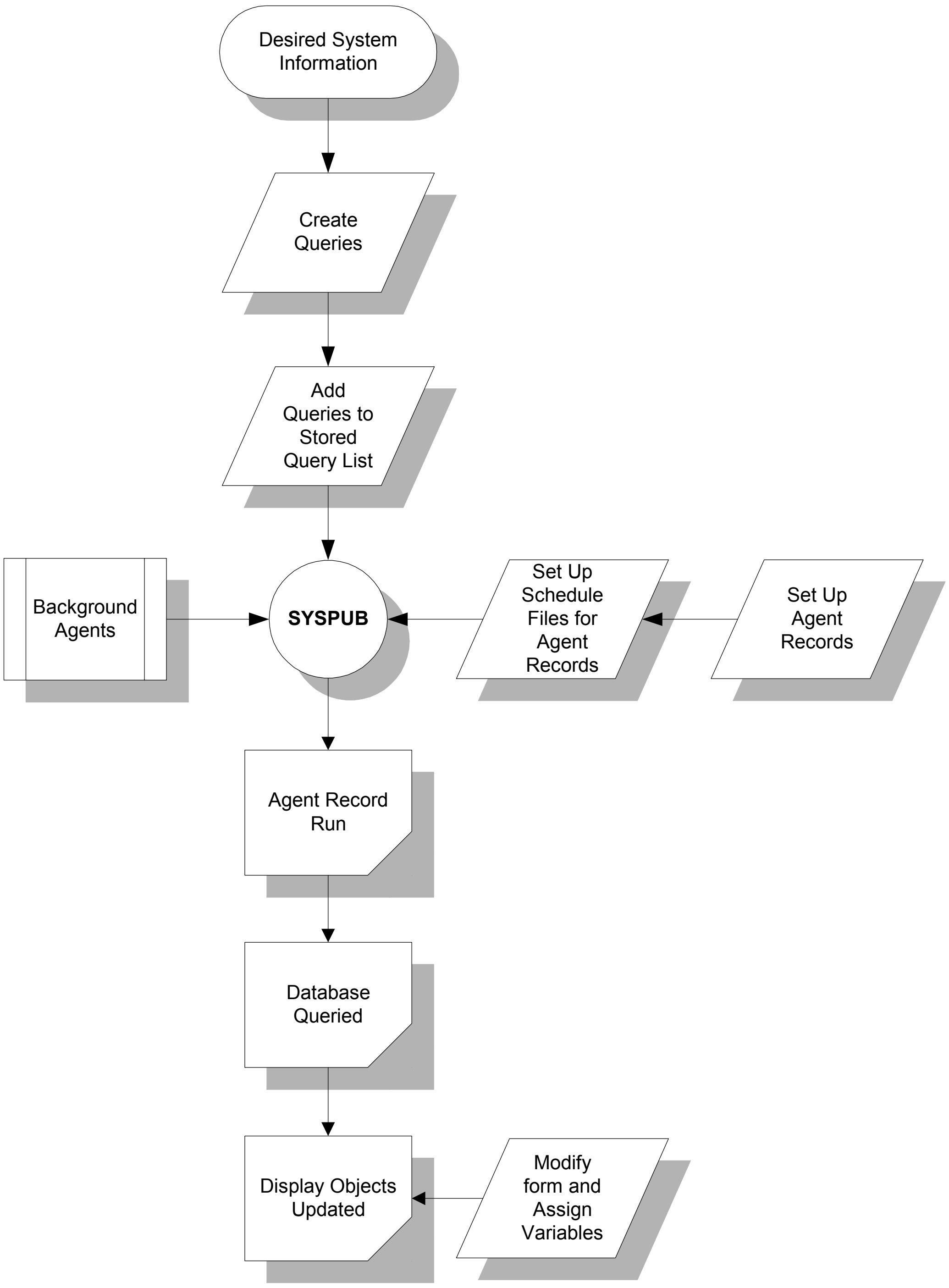
Enhanced forms

This process adds an updating display object to an existing form and creates a dynamic view of some aspect of your system for user groups.


Desired System Information
Create Queries
Add Queries to Stored Query List
Background Agents
SYSPUB
Set Up Schedule Files for Agent Records
Set Up Agent Records
Agent Record Runs
Database Queried
Display Objects Updated
Create Operator Record
Create Menu
Design Form and Assign Variables

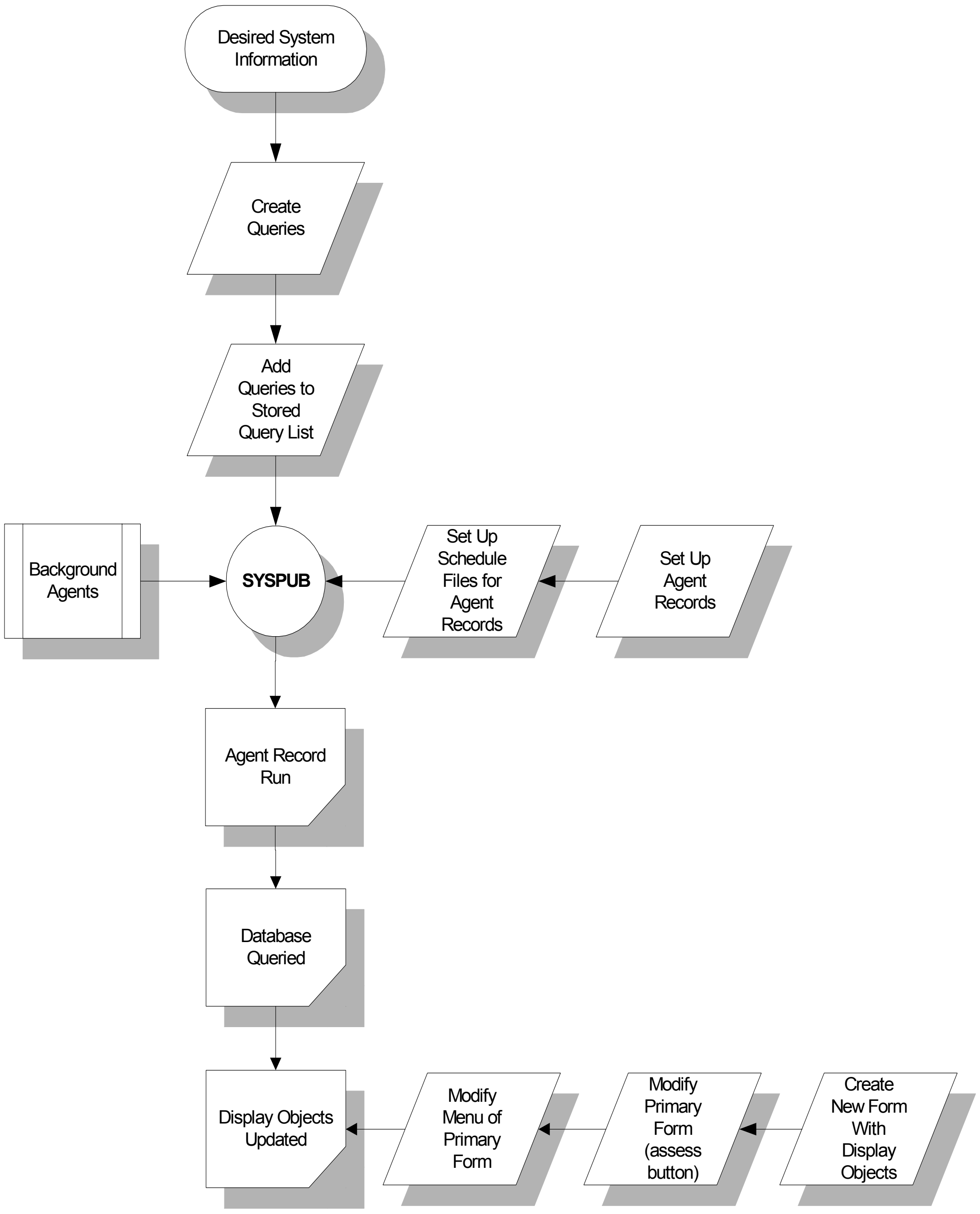
Option forms

This process creates a new form with updating display objects that is accessed from an existing form by an option button. This form may be available only to a certain group of users.


Desired System Information
Create Queries
Add Queries to Stored Query List
Background Agents
SYSPUB
Set Up Schedule Files for Agent Records
Set Up Agent Records
Agent Record Run
Database Queried
Display Objects Updated
Modify form and Assign Variables

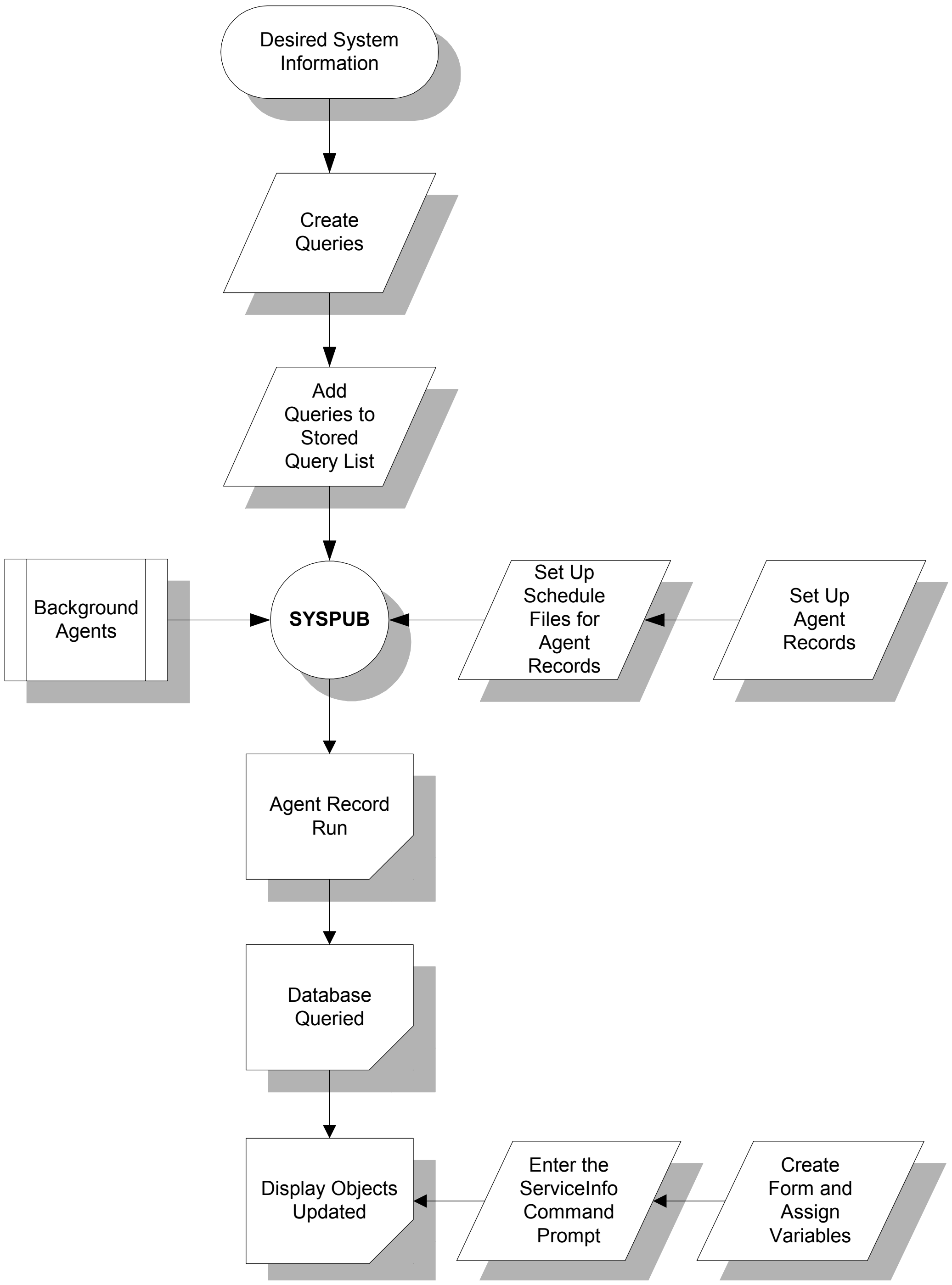
Startup screen

This process creates a startup screen for a user or group of users containing updating display objects and access buttons for other ServiceCenter features.


Desired System Information
Create Queries
Add Queries to Stored Query List
Background Agents
SYSPUB
Set Up Schedule Files for Agent Records
Set Up Agent Records
Agent Record Run
Database Queried
Display Objects Updated
Modify Menu of Primary Form
Modify Primary Form (assess button)
Create New Form With Display Objects

ServiceInfo

This process creates a new form with automatically updating system information for display purposes. ServiceInfo is a marketing tool and provides no access to the system. Subscribers need not be logged on to ServiceCenter to display a ServiceInfo screen.


Desired System Information
Create Queries
Add Queries to Stored Query List
Background Agents
SYSPUB
Set Up Schedule Files for Agent Records
Set Up Agent Records
Agent Record Run
Database Queried
Display Objects Updated
Enter the ServiceInfo Command Prompt
Create Form and Assign Variables

Publishing

Management View publishing is a three step process:

1. Set up queries or access existing stored queries.
2. Create an Agent Record.
3. Schedule the agent.

Stored queries

Stored queries define the search parameters for data acquisition. The agents responsible for updating messages must query the database and report back with the current information. In order to publish a message that will monitor alert status or count tickets of a certain priority, queries must be created and stored.

The easiest method to generate a stored query is to select the information you want using normal search methods, open the **Advanced Search** window and store your query directly. By following these steps you can ensure that the query you build returns only the information you want it to return; moreover, the query is optimized for best performance. Refer to Chapter 58, *Stored Queries* for additional information about creating and maintaining stored queries.

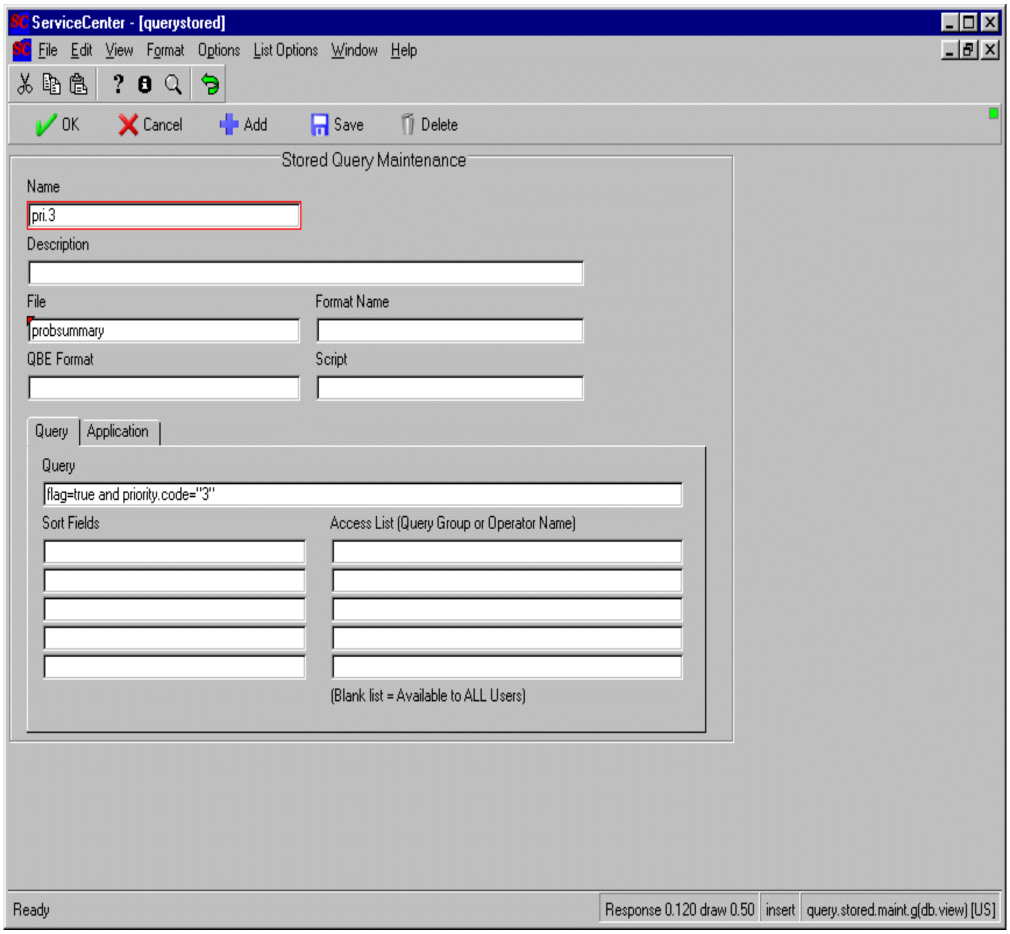
Select the queries you want to use for your published message from the stored query list.

To view a stored query:

1. Select the **Utilities** tab in the system administrator's home menu.
2. Click **Tools**.
3. Click **Stored Queries**.
4. Enter `pri.3` in the **Name** field.

   This query returns an URGENT priority message to the subscriber.
5. Click **Search**.

The **pri.3** stored query record is displayed.


ServiceCenter - [querystored]
File Edit View Format Options List Options Window Help
OK
Cancel
Add
Save
Delete
Stored Query Maintenance
Query
Application

Name: pri.3
Description:
File: probsummary
Format Name:
QBE Format:
Script:
Query: flag=true and priority.code="3"
Sort Fields:
Access List (Query Group or Operator Name):
(Blank list = Available to ALL Users)
Ready | Response 0.120 draw 0.50 | insert | query.stored.maint.g(db.view) [US]

Figure 19-1: Stored Query Maintenance form displaying a record

6 Click Cancel to return to the blank form.

7 Type dept.problems in the **Name** field.

This query returns a list of all your department's incident tickets.

8 Click **Search** to display the record.

Note: There are a number of predefined queries in the standard ServiceCenter system. You may use these or create your own.

Agent records

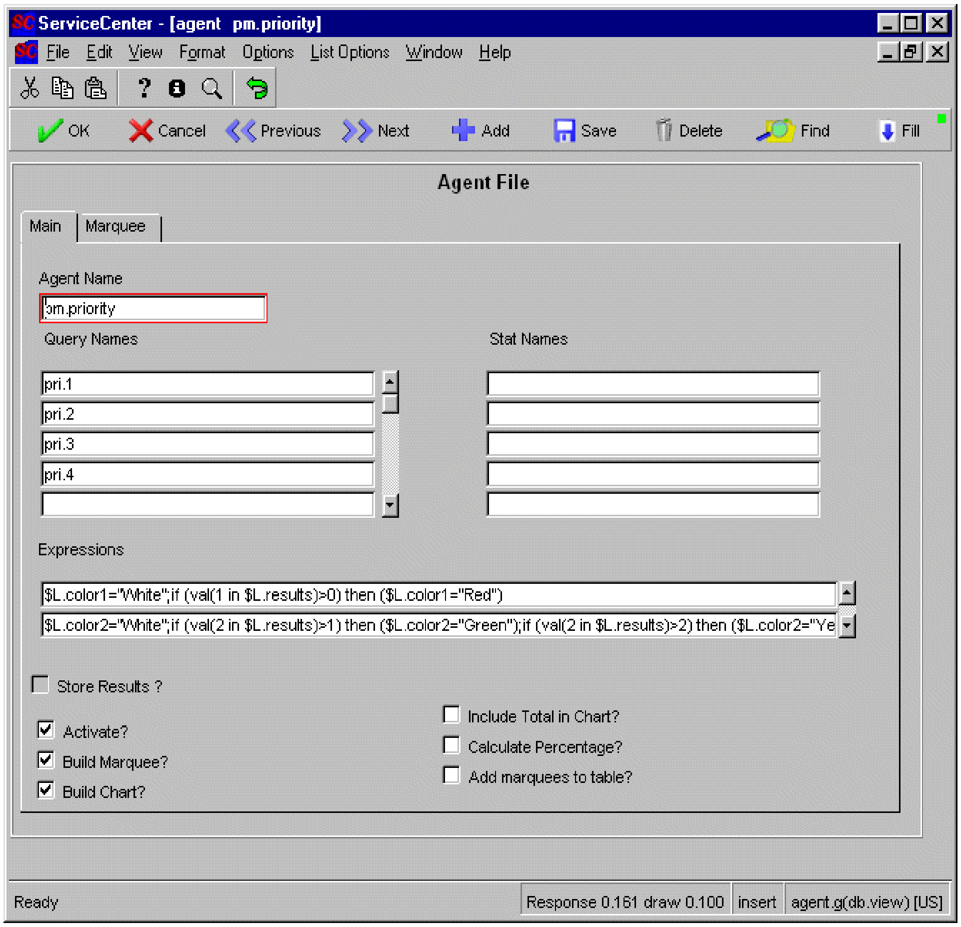
Agents are special background processes used to perform certain tasks for the user. In the case of Publish and Subscribe, they count information in your database, such as the number of open incident tickets in a specified alert status, or the number of tickets assigned to a certain group. *Agent Records* are the resources used by these background processes to reference stored queries, define messages and colors, and select display options. These records are published to the ServiceCenter publishing house and used by agents to update a subscriber's charts and marquees.

Creating an agent record

To create an agent record:

1. Select the Utilities tab in the system administrator's home menu.
2. Click **Tools.**
3. Select the Publishing Utilities tab
4. Click **Define Agents.**
5. Click **Search** to display the Agent Records in your database.
6. Select **pm.category** from the record list.

This agent publishes tickets by category.


ServiceCenter - [agent pm.priority]
File Edit View Format Options List Options Window Help
OK Cancel Previous Next Add Save Delete Find Fill

Agent File

Main | Marquee

Agent Name: pm.priority

Query Names	Stat Names
pri.1	
pri.2	
pri.3	
pri.4	

Expressions

```
$L.color1="White";if (val(1 in $L.results)>0) then ($L.color1="Red")
$L.color2="White";if (val(2 in $L.results)>1) then ($L.color2="Green");if (val(2 in $L.results)>2) then ($L.color2="Ye
```

- ☐ Store Results ?
- ☑ Activate?
- ☑ Build Marquee?
- ☑ Build Chart?
- ☐ Include Total in Chart?
- ☐ Calculate Percentage?
- ☐ Add marquees to table?

Ready | Response 0.161 draw 0.100 | insert | agent.g(db.view) [US]

Figure 19-2: An agent record

7 Select **pm.priority** to display the agent record for publishing priority status reports.

Note: The agent records in this list are included in the standard ServiceCenter system.

Main tab fields

Field	Description
Agent Name	Unique name for an agent. These names may be descriptive of the agent's function.
Query Names	Array field of stored queries in the record used by the agent to update charts and marquees on your users' forms. Each query returns a string containing the number of records selected. This string is stored as an element in a list, called $L.results, that corresponds to the position on the list of the query requesting the count. To return the second stored query value (pri.1) in this example, use the following expression: str(tod())+": There are "+2 in $L.results+" CRITICAL priority incidents."
Stat Names	Identifies the value being stored in the stathistory file when the Store Results field evaluates to *true*. When Stat Names are specified and the Store Results is checked or otherwise set to true, the queries' counts post to a stathistory file. Each time an agent executes, one stathistory record is created for each query with a corresponding Stat Name. Since agents typically execute their queries repeatedly, based on repeat intervals specified in their corresponding agent schedule records, this feature produces recurring data points suitable for trend analysis. **Note:** If you use this feature in agents with frequent repeat intervals, the stathistory file grows quickly. Peregrine recommends coding separate agent(s) specifically for collecting trending statistics. Schedule the trending agents for longer intervals: hourly, daily, weekly, or monthly.
Expressions	Sets the colors for the marquees based on values returned from the stored queries. These expressions are evaluated once—after the stored queries are processed and before any of the marquees are evaluated.
Store Results?	Values returned by the stored query are stored in the stathistory file if this box is checked.
Activate	Tells background processing agent to process this record. Only active agent records are used to count data.
Build Marquee?	Evaluate and publish the marquee expressions.
Build Chart?	Publish the chart variable. Stored queries are still processed.

Field	Description
Include Total in Chart	Publish (and append $L.results) an extra value containing the total of all the stored query counts.
Calculate Percentage	Calculate the percentages of the total query counts. This option is exclusive of the Include Total in Chart option.
Add marquees to table	■ Add the marquees to the marquee table so that the marquee background agent can publish them. ■ Deselect to publish the marquees immediately without putting them into the marquee table. (Recommended)

Marquee tab fields

Agent	Description
Text	Defines the marquee text for each query and appends a date/timestamp prefix to the result. The variable, $L.results, is the record count generated by the stored query. Use variables and function calls to make up the message. The text of each marquee is evaluated at run time.
Color	Defines the marquee color for each query. The color of each marquee is evaluated at run time. The color is either a valid system color or its numeric equivalent

Scheduling background processes

A schedule record is required for every agent that runs in ServiceCenter. In order for a subscriber to receive dynamic data, an agent background processor must be running to process agent records. Publish and Subscribe requires two background agents, **marquee** and **agent**, to be running in order to display data in marquees or charts.

These background agents are set to read records at specified intervals.

There are two methods for running these agents:

- Register them at startup
- Start and stop them during a session from the **system.status.list** form.

System startup

To load these agents into the startup record for your system:

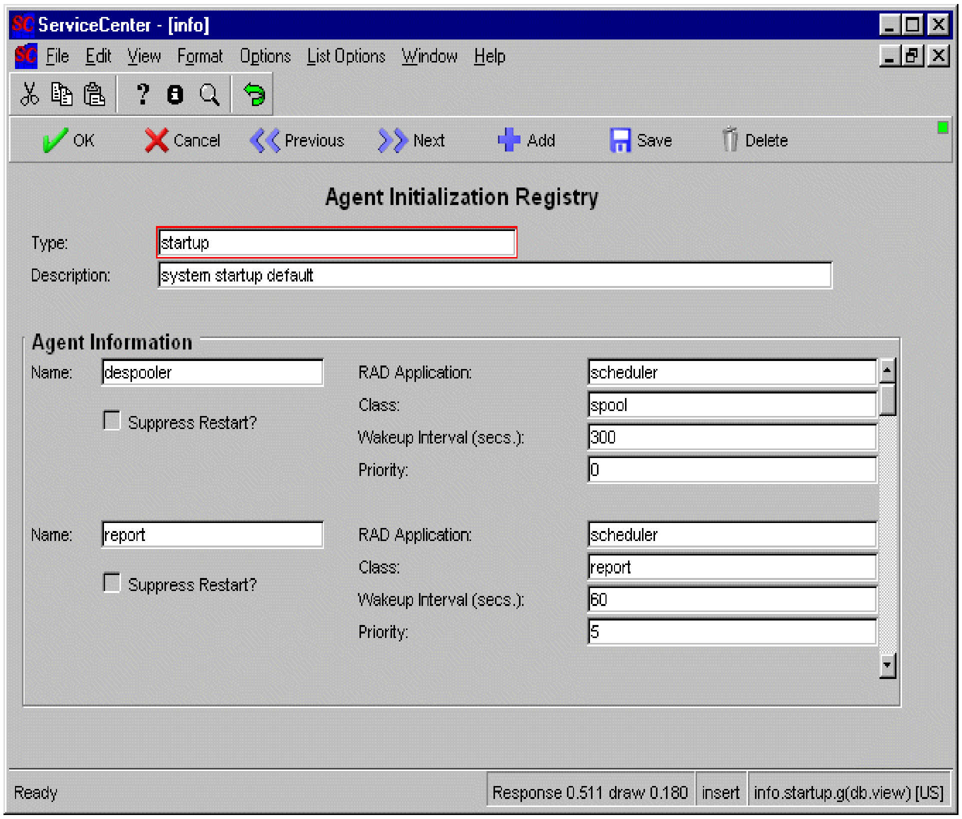
1 Select the **Utilities** tab in the system administrator's home menu.

2 Click on the **Maintenance** button.

3 Click **Startup Information** in the System tab.

4 Enter startup in the **Type** field.

5 Press **Enter** or click **Search**.



Agent Initialization Registry

Type: startup
Description: system startup default

Agent Information

Name: despooler
Suppress Restart?
RAD Application: scheduler
Class: spool
Wakeup Interval (secs.): 300
Priority: 0

Name: report
Suppress Restart?
RAD Application: scheduler
Class: report
Wakeup Interval (secs.): 60
Priority: 5

Ready | Response 0.511 draw 0.180 | insert | info.startup.g(db.view) [US]

Figure 19-3: Agent initialization registry record

6 Scroll through the names in the Agent Information structure. To run at startup, **marquee** and **agent** must be listed here.

7 Select a **Wakeup Interval** in which the agent background process will read the records.

Note: The **startup** registry functions separately from the registry for individual agents. To change the settings for an agent, be sure to change both records.

Important: Background processes must be started from the server.

Starting background processes

Agent background processes can be started from the System Status form (system.status.list.g)

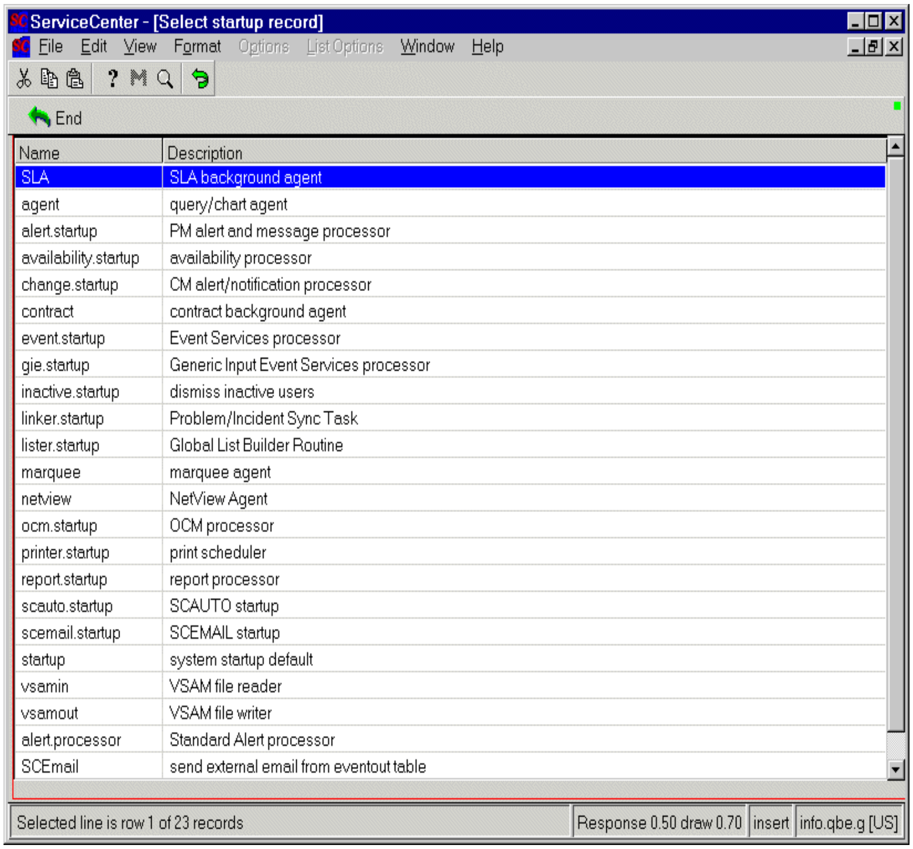
1 Click on the **System Status** button in the system administrator's home menu.

2 The System Status form is displayed, listing all current background processing agents.

3 Click **Start Scheduler** to display a list of background processes to start.

Important: You must connect to the server with an express client to access the Start Scheduler button. This button is grayed out for a full client connection.

A list of all the processes available is displayed


ServiceCenter - [Select startup record]
File Edit View Format Options List Options Window Help
End

Name	Description
SLA	SLA background agent
agent	query/chart agent
alert.startup	PM alert and message processor
availability.startup	availability processor
change.startup	CM alert/notification processor
contract	contract background agent
event.startup	Event Services processor
gie.startup	Generic Input Event Services processor
inactive.startup	dismiss inactive users
linker.startup	Problem/Incident Sync Task
lister.startup	Global List Builder Routine
marquee	marquee agent
netview	NetView Agent
ocm.startup	OCM processor
printer.startup	print scheduler
report.startup	report processor
scauto.startup	SCAUTO startup
scemail.startup	SCEMAIL startup
startup	system startup default
vsamin	VSAM file reader
vsamout	VSAM file writer
alert.processor	Standard Alert processor
SCEmail	send external email from eventout table

Selected line is row 1 of 23 records | Response 0.50 draw 0.70 | insert | info.qbe.g [US]

Figure 19-4: Scheduler processes

4 Double–click on the agent you wish to start.

Stopping background agents

Agent background processes can be stopped from the System Status form (system.status.list.g).

1 Click on the **Command List** button to review available commands.

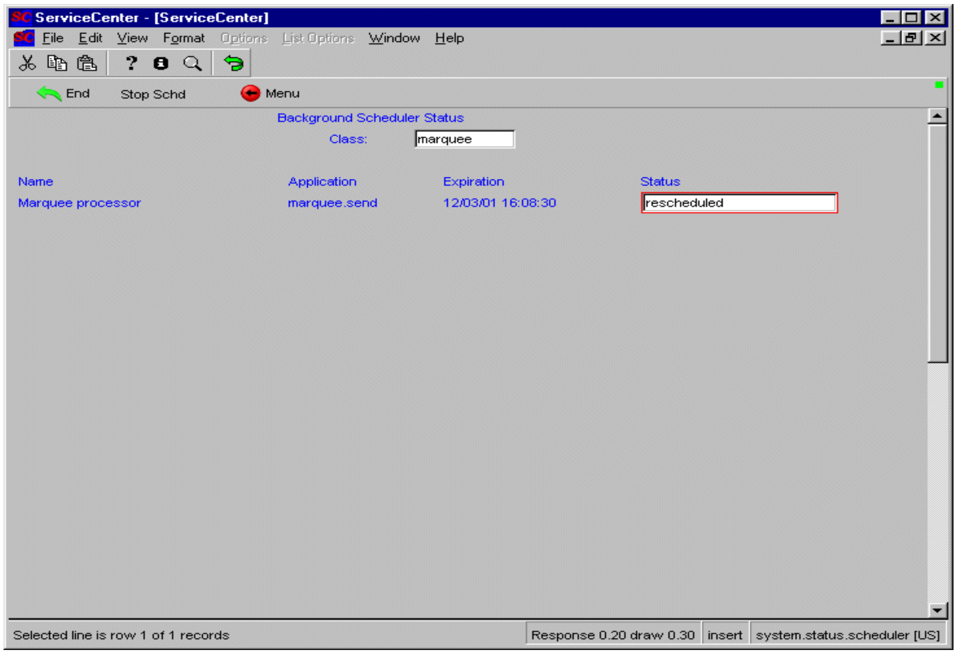
A list of available commands is displayed.

2 Click **End** to return to the system status form.

3 Type s in the command box beside the process.

4 Click **Execute Command.**

The Background Scheduler Status form is displayed.


ServiceCenter - [ServiceCenter]
File Edit View Format Options List Options Window Help
End
Stop Schd
Menu

Background Scheduler Status

Class: marquee

Name	Application	Expiration	Status
Marquee processor	marquee.send	12/03/01 16:08:30	rescheduled

Selected line is row 1 of 1 records | Response 0.20 draw 0.30 | insert | system.status.scheduler [US]

Figure 19-5: Background scheduler status

5 Click **Stop schd.**

The Schedule Stop window is displayed.

6 Enter the date and time you want the background process to stop.

7 Close the window.

8 Click **End** to return to the system status form.

9 Click **SC Menu** to return to the system administrator's home menu.

Note: For a detailed discussion of the System Status form, refer to the ServiceCenter *System Administrator's Guide.*

Schedule file

Once you have created your Agent Record, you must *schedule* it to be run by the background agent. The Schedule File allows you to establish the interval in which your background processing agents refresh your charts and marquees.

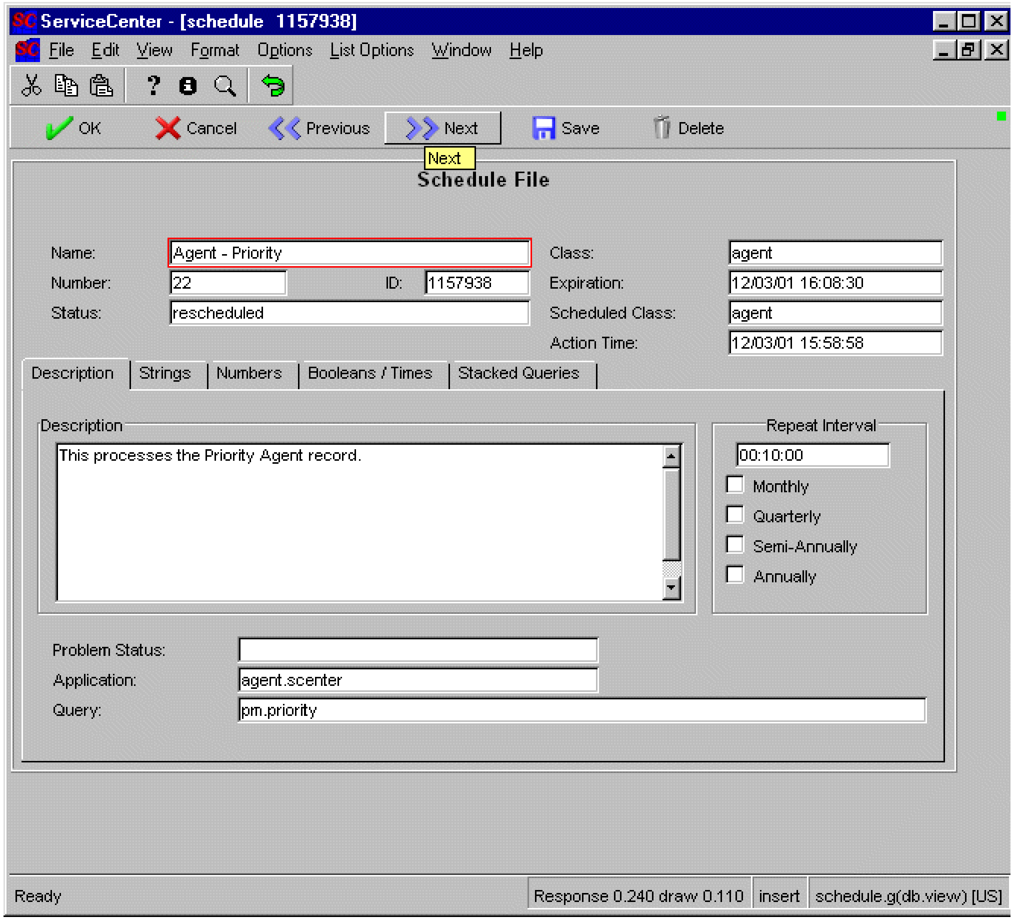
To access the schedule file:

1 Select the **Utilities** tab in the system administrator's home menu.

2 Click **Tools**.

3 Select the **Publishing Utilities** tab.

4 Click **Modify Schedule** to schedule your agent.


ServiceCenter - [schedule 1157938]
File Edit View Format Options List Options Window Help
OK
Cancel
Previous
Next
Save
Delete
Next
Schedule File

Name:	Agent - Priority	Class:	agent
Number:	22 ID: 1157938	Expiration:	12/03/01 16:08:30
Status:	rescheduled	Scheduled Class:	agent
		Action Time:	12/03/01 15:58:58

Description: This processes the Priority Agent record.

Repeat Interval: 00:10:00
☐ Monthly
☐ Quarterly
☐ Semi-Annually
☐ Annually

Problem Status:	
Application:	agent.scenter
Query:	pm.priority

Ready | Response 0.240 draw 0.110 | insert | schedule.g(db.view) [US]

Figure 19-6: Schedule file records

5 Select a record from the record list for the type of display object you want to create. For example, you might select **Agent–Priority**, which retrieves priority status data.

Important: The entry in the **Query** field must match the **Agent Name** of the agent record (Figure 19-2 on page 329).

6 Reset the counter in the Repeat Interval structure to the desired update interval.

7 Click **Save**.

Marquee processor

If you have selected the **Add marquee to table** function in the agent record (see page 331), your marquees are not published immediately, but are added to the marquee table. In order to publish them from the table, you must schedule a publishing interval with the *marquee processor*.

To access the schedule file:

1. Select the **Utilities** tab in the system administrator's home menu.
2. Click **Tools.**
3. Select the **Publishing Utilities** tab.
4. Click **Modify Schedule** to schedule your agent.
5. Select Marquee processor from the record list.
6. Schedule the time interval for publishing your marquee in the **Repeat Interval** structure.
7. Click on the **Save** button to save the record.

   The following message is displayed in the status bar: *Record updated in the schedule file.*

Subscribing

Subscribing to Management View data published in ServiceCenter requires the use of display objects whose fields are identified by selected variables. These variables, stored in ServiceCenter's publishing house (SYSPUB), return the data published by the Agent Record.

There are several ways in which to view published data:

- **Startup screens for user groups.**

Startup screens are created with each user's system rights in mind. Custom menus with automatically updating display objects provide end users with appropriate system tools.

- **Display objects added to another form.**

Existing forms and menus can be modified to display Basic or Management View data.

- **Button access to a custom form.**

Add a button in any ServiceCenter form to access a custom form containing updating display objects.

Note: For detailed information about designing forms, refer to ServiceCenter *Forms Designer*.

Designing the form

In this example, we will create a marquee that displays the alert level status of incident tickets in the ServiceCenter database and a chart that indicates open incident tickets by category.

All forms must have a Back button or a Logoff button. For this example, we will create a Logout button to exit ServiceCenter.

To design a form containing dynamic display objects:

1. Select the **Toolkit** tab in the system administrator's home menu.
2. Click **Forms Designer**.
3. Enter xxx.status (where *xxx* are your initials) in the **Form** field of the Forms Designer dialogue box.
4. Click **New**.

   A prompt appears asking if you want to use Form Wizard.
5. Click **No**.

   A blank drawing canvas is displayed.
6. Select the Marquee tool from the Tool Palette and draw a marquee with the following values:

Field	Value
Input:	$MARQUEE.pm.priority3
X:	20
Y:	2

Height:	3
Width:	120
FontIncrease:	3

The marquee variables are named **$MARQUEE.<*name*><*line*>**, where *name* is the unique name of the agent record, and *line* is the array position of the marquee message in the stored query list. (See page 330)

7 Select the Chart tool from the Tool Palette and draw a chart with the following values:

Field	**Value**
Input:	$CHART.pm.category
X:	52
Y:	9
Height:	13
Width:	57
BarWidth:	5
ScaleMax:	0
ButtonBase:	Yes
ButtonBaseID:	11
ColorList:	Green;Blue;Yellow;Red
ColorScale:	10;20;40

Chart variables are named **$CHART.<*name*>**, where *name* is the unique name of the Agent Record.

8 Using the Label tool, create a list of button ID's for your chart:

1= DEFAULT

2= Abends

3= Circuit

4= Data Network

5= Equipment

6= Example

7= Host Hardware

8= IP Network

9= Local Area Network

10= Software

Use the following values for each Label field:

<n> = <Category>

9 Select the Button tool from the Tool Palette and create a button with the following values:

Field	**Value**
Caption:	Logout
X:	68
Y:	35
Height:	3
Width:	21
ButtonID:	3

Note: ServiceCenter uses **3** as an ID number for **Logout** and **Back** buttons.

10 Place the new **Logout** button at the bottom of your form, beneath the category list.

11 Click **OK.**

12 Your form will look like this:

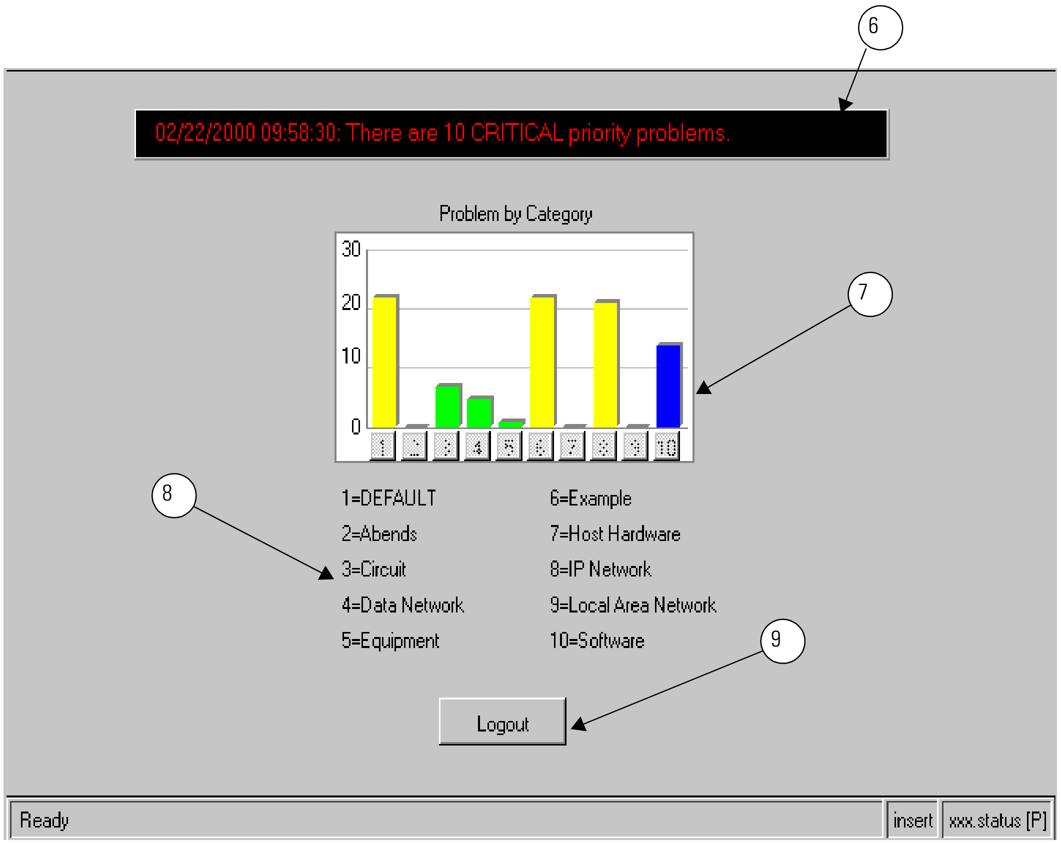


Figure 19-7: Form with updating display objects

Creating menus

Physically, your form is complete. The marquee will show updated ticket priorities, and the chart will display incident tickets by category; however, button features will not function until you assign applications to the button ID's. This is done by creating a *menu* for your form in which all button ID's are defined. This menu can then be used to reference the form in an Operator Record or to a button ID in another ServiceCenter form. For additional detail on menu setup, refer to the ServiceCenter *System Administrator's Guide*.

You must create a menu record for the form *xxx*.**status** and assign parameters to button ID's 3 and 11—20. (Numbers 11—20 were assigned to incident ticket categories as defined in the Stored Query list, and button 3 was assigned to the **Logout** button.)

Use the following procedures to create a menu:

1. Enter `menu` in a command line.
2. Press **Enter.**
3. A blank menu form is displayed.
4. Type **XXX STATUS** in the **Menu Name** field.
5. Type the name of your form (*xxx*.**status**) in the **Format** field.

Important: Use uppercase letters where indicated.

6. Enter the following values in the fields indicated:

Option #	Command	Application	Parameter Name	Parameter Value	Thread	Condition for Option
3	logoff	menu.manager	name	LOGOUT	true	true
11		query.stored	text	cat.default	true	true
12		query.stored	text	cat.abends	true	true
13		query.stored	text	cat.circuit	true	true
14		query.stored	text	cat.datanetwork	true	true
15		query.stored	text	cat.equipment	true	true
16		query.stored	text	cat.example	true	true
17		query.stored	text	cat.hosthardware	true	true
18		query.stored	text	cat.ipnetwork	true	true
19		query.stored	text	cat.lan	true	true
20		query.stored	text	cat.software	true	true

7. Click **Add.**

Note: **Option #** corresponds to the button ID numbers assigned on the form.

Displaying the data

The only step remaining in the *subscribing* process is to make your new form available to the end user. Display objects can be added to any existing form or used to create a startup screen for a group of users. For this example, create an operator record for *xxx***user** and assign our form as that user's startup screen. For more information about creating an operator record, refer to the ServiceCenter *System Administrator's Guide*.

To create an operator record

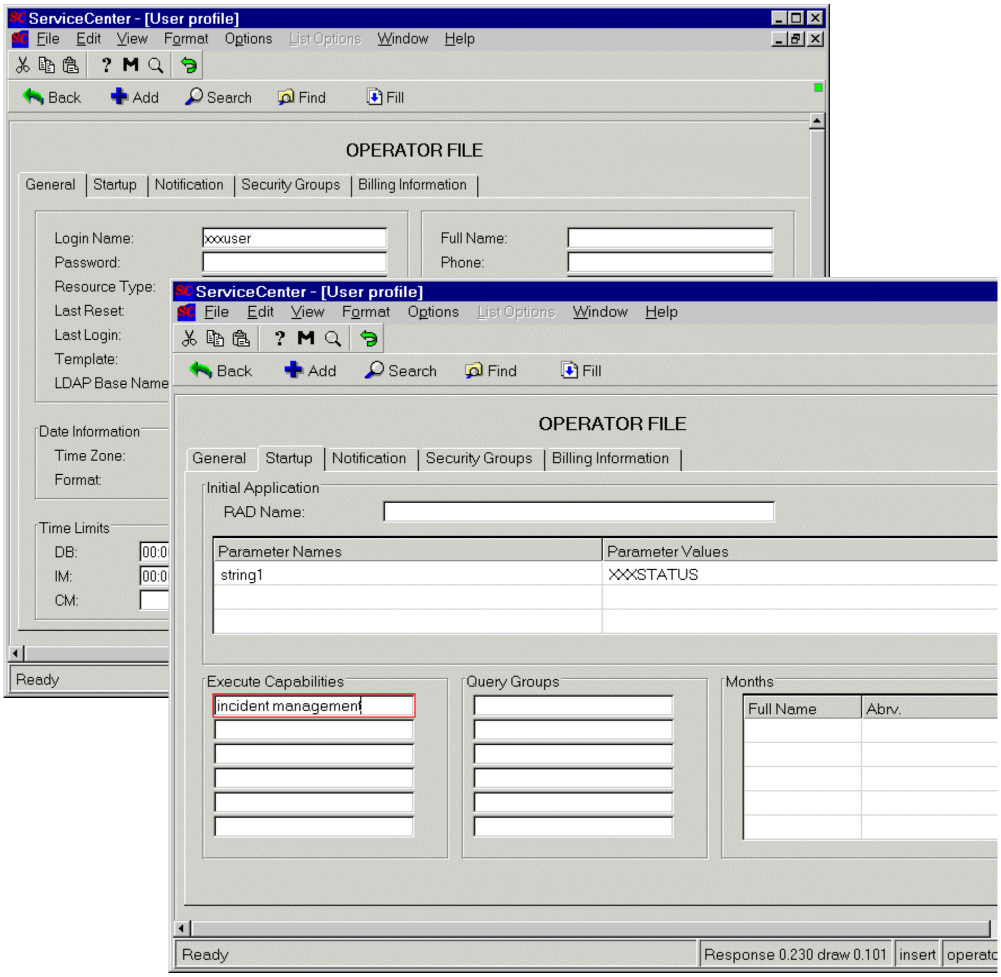
1 Create an operator record using the Centralized Administration Utility (CAU). Utilities>Administration>User Administration.

2 Create a new user using the User Quick Add Utility.

3 Enter these values in the following fields:

Field	**Value**
Login Name:	*xxx*user
Execute Capabilities:	Incident Management
RAD Name:	menu.manager
Parameter Names:	string1
Values [string1]:	*XXX* STATUS

4 Click **Add**.



OPERATOR FILE

Login Name: xxxuser
Password:
Resource Type:
Last Reset:
Last Login:
Template:
LDAP Base Name
Full Name:
Phone:

Date Information
Time Zone:
Format:

Time Limits
DB: 00:0
IM: 00:0
CM:

OPERATOR FILE

Initial Application
RAD Name:

Parameter Names	Parameter Values
string1	XXXSTATUS

Execute Capabilities
incident management

Query Groups

Months

Full Name	Abrv.

Figure 19-8: Operator Record

Note: XXX STATUS is the name of the menu you created for your custom form in the last section. It must appear in uppercase type.

Check new user

Log out and begin another session. This time, log on as *xxx***user** to test the startup screen you created. Your form should appear immediately after login. Open tickets in the various categories using the button bar beneath the chart. When you are through, click on the **Logout** button to end the session.

Editing the marquee

To change the message displayed in the marquee, open another System Administrator client and edit the marquee using the following procedure:

1 Open your form (*xxx*.**status**) in Forms Designer.

2 Click **Design**.

3 Select the marquee.

4 In the Properties Box, change the **Input** field to read: $MARQUEE.pm.priority4.

5 Press **Enter**.

6 Click **OK**.

The marquee now displays URGENT priority incidents in green rather than CRITICAL priority incidents in red.

Note: You may create as many marquees as you need to display your status messages. Assign a different priority number to each (e.g., pm.priority1, pm.priority2, pm.priority3, etc.).

Publishing in the RAD Environment

You can call the following RAD applications from Format Control to display Publish and Subscribe messages. The table below shows the proper parameter names and values to pass to the application called. For instruction on creating a Format Control subroutine call, refer to *Format Control* in this book.

Application	Parameters Names	Values
publish—publishes a variable into SYSPUB	name	The published variable used by the agent to search the database.
	text	Variable that identifies the message to be published.
marquee.publish—sends a marquee message. This application is always called from within **marquee.send**.	name	The published variable used by the agent to search the database. **Note:** The system automatically prepends a **$MARQUEE** to your variable name before publishing it.

Application	Parameters Names	Values
	text	Either the variable that identifies the message to be published or the actual text of your message in the **text** field.
	prompt	Display color for your marquee or the valid system number in the **Color** field. The default is Red.
	boolean1	Logical field. Set the flag to *true* to add this message to the marquee table. The marquee agent reads this record and publishes it. If the flag is set to *false*, the application immediately publishes your message without adding it to the marquee table. This is the recommended method.
	cond.input	Logical field. Set the flag to true to add the name to the database if that record does not exist.
marquee.send—routine that reads the schedule record and initiates the **marquee.publish** application	record	Schedule record for the marquee process
	boolean1	Logical field. Set the flag to *true* to send all the marquee messages defined in the schedule record. If the flag is set to *false*, or the field is NULL, the application publishes only the specific marquee message defined in the schedule record. This is the recommended method.

20 ServiceInfo

CHAPTER

ServiceInfo forms allow users to have the power of agents without actually having to be logged on to the system, and without having a count against the user license. ServiceInfo forms are created in Forms Designer and subscribe to the same published data as any management view of the system. They display updated information only and provide no access to the system.

ServiceInfo data is published to **SYSPUB** in the same manner as Management View data (see *Management View of a System* on page 317)

- Establish stored queries
- Define agents
- Schedule agent background processes

Subscribers need only to access the ServiceCenter executable file to display a ServiceInfo form.

Designing the Form

ServiceInfo forms can display any of the data contained in a typical Management View startup screen. This includes updated marquees and charts reflecting the changing status of your system. Build your forms just as you would for any user group, but without button access to the system. (For more information on building forms, refer to ServiceCenter *Forms Designer*).

You may give your ServiceInfo form any name you please. The example shown here uses an **si.** prefix to give it a name unique to ServiceInfo.

Use the *$CHART* and *$MARQUEE* variables to subscribe to the data defined by your stored query list.

In this example:

- *$CHART.pm.daily.aging*,
- *$CHART.pm.status*
- *$MARQUEE.pm.priority1*.

Important: ServiceInfo forms automatically size themselves to the available screen dimensions using the lower most object in the right corner of the screen as a guide. In order to properly center your display, put a bevel frame around the entire form with the bevel tool.

Use the following values for a bevel frame:

- Outer bevel = 3
- Inner bevel = 0

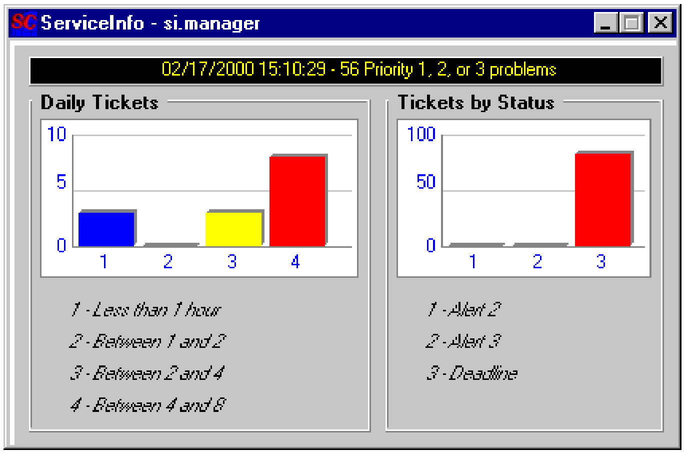


Figure 20-1: ServiceInfo Level of Urgency Form

Subscribing to the Data

To display a ServiceInfo form, type `-si:<form name>` after the command you normally use to start ServiceCenter. *Form name* is the name of the form you want to display. In this example, **si.manager**.

ServiceInfo command prompts:

From a UNIX text prompt, enter:

<*fully qualified path to scenter directory*> `scenter -si:`<*form name*>

- From Windows operating systems, the command line would read:

```
<my disk>:\<directory>\scguiwns.exe -si:<form name>
```

Index

A

accessing Forms Designer 29
adding
  records 156
additional file queries process (Format Control)
  fields 230
additional options 149, 157, 160, 167, 171, 250
  example 256
  fields 253
agent (background agent)
  loading at startup 332–333
  starting 333–334
  stopping 334–335
agent records
  creating 328–329
  data fields 330–331
alerts
  Incident Management 185
animation 138
applications
  called from Format Control
    axces.fax 303
    axces.page 301
    axces.write 302
    database 300
    determining parameters 305
    fill.fc 295
    fingerprint 304
    getnumb.fc 297
    message.fc 298
    post.fc 296
    query.stored 301
    sort.array 305
    validate.fields 299
arrayed structures 224
  as.copy 163
    parameters 163
  as.delete 164
    parameters 164
  as.get.name 169
  as.insert 165
    parameters 165
  as.move 166
    example 167
    parameters 166
  as.options 158
    example 159
    parameters 159
  as.sort 169
    parameters 169
  subformats 158, 165
  user options
    copy 162, 168
    delete 162, 168
    insert 162, 168
    move 163, 168
    replace 163, 168
arrays
  denulling 227
  editing 204
  editor 157

sorting 173–177
automatic updates 307

B
background agents
scheduling 335
background processes
scheduling 331–335
basic publish and subscribe 315–316
simple messages 311–312
Bevel tool 54
boolean fields 152, 231, 233
box properties 39
Button tool 50

C
calculations process (Format Control) 148, 210, 221, 222, 223
canvas invisible grid 19
CenterPoint Web site 14
Change Management
approvals
Format Control record 181
changes
opening 282, 286
detail record 181
master record 181
processing considerations 180
Chart tool 91
charts 308
creating 339
variables in agent record 330
Check Box tool 70
colors
setting 308, 330, 331
colors on charts 94
Combo Box tool 78
ComFill tool 108
common properties 37
conditional statements
dynamic functions 131
field comparisons 129
field value matching 130
Courier font 134
creating a new form 29

D
data fields
creating 46
default fonts in ServiceCenter 134
deleting an object 43
deselecting objects 43
design
guidelines 136
process 30
tips 133
drawing canvas
introduction to 18
dynamic functions 131

E
Edit Area list 35
education services 15
errors
messages 293
eventout records
Incident Management 182–183
expressions
boolean fields 152
definitions
primary file 150
semicolon 150
statements 151
variable 150
error messages 293

F
field comparisons 129
field value matching 130
Fill Box tool 63
font properties 38
fonts 134
form style guidelines 133
Form Wizard
accessing 141
creating a form 143
introduction 141
Format Control
accessing
from a menu 156
from Forms Designer 155

creating records 174, 198, 278, 284
editing records 211
eventout records 182–183
privileges 233
process views 148
processes
additional file queries 229
additional options 149, 250
calculations 148, 210, 221
examples 223
fields 222
initializations
fields 207
queries 148
security 234
selecting a view 149
subroutines 149, 240
validations 148, 213
views 258
QBE forms 258, 267
creating 208
default 207
scripting 209
Forms Designer
accessing 29
case sensitivity 36
design process 30
introduction 17
Java client 23
properties ??–39
sizing on the drawing canvas 20
timer tool 118
validating formats 273
Frame tool 56
functions
denull 151
index 151
lng 151
nullsub 151
operator() 151
str 151
substr 151
tod() 151
val 151, 159, 164

G
Gfile tool 114
Graphic tool 67
graphics and animation 138
grid units 134

I
Incident Management
alert records 185
eventout records 182–183
processing considerations 179
initialization expressions 171, 210
incorrect 293
initializations process (Format Control)
fields 207
invisible grid 19

L
Label tool 44
lassoing an object 43
link records
creating 275
fields 263
fields in link line definition file 266
link line definition file 265
posting 262
Live User Interface
dynamic displays
enhanced forms 318
option-accessed forms 320
ServiceInfo 324
startup screens 322
features
automatic updates 307
system load 308
user interaction 307
forms
charts 339
creating for dynamic display 338–340
designing 308, 317
marquees 338
modifying
marquees 314–315
messages 311–312

        ServiceInfo 347–348

M

marquee (background agent)
    loading at startup 332–333
    starting 333–334
    stopping 334–335
Marquee tool 83
marquees
    adding to a form 314–315
    color
        setting in agent records 331
    colors
        setting in agent records 330
    creating 338
    described 308
    editing 345
    expressions in agent records 330
    publishing 315–316
    scheduling 337
    simple 314–316
menus, creating 341–342
messages
    designing message field 311–312
    publishing 313–314
    simple 311–312
Multi-Line Text Box tool 60

N

Notebook tool 102

O

OLE container tool 122
operator record
    creating 343–344, ??–345

P

palette, tools 21
parameters
    application 305
Peregrine Systems
    Corporate headquarters 14
    Worldwide Contact Information 14
performance, graphics 139
Pointer tool 42
positioning an object 43
posting
    additional options set up 271
    additional options setup 271
    automatic
        confirmation function 288
        example 272
    compared with Fill 261
    confirmation function 288
    link records 262
        fields 263
        link line definition file 265
    post.fc 271
    subroutines set up 279
    user options
        confirmation function 288
        example 283
    variables 268
privileges 233
Properties Window 22, 33–39
    box properties 39
    common properties 37
    Edit Area list 35
    entering a value manually 35
    font properties 38
    selecting a property 35
    sizing 34
    using the window 34
property conditions 129
Publishing 315
publishing
    defined 308
    dynamic data 318–337
        agent records 328–331
        schedule file 335–337
        scheduling background processes 331–335
        stored queries 326–327
    marquees 315–316
    messages 313–314

Q

QBE list
    forms 258, 267
        creating 208

default 207
selecting views in Format Control 148
queries process (Format Control) 148

R

Radio Button tool 73
records
editing 211
resizing an object 43

S

SCAutomate
page messages 184
schedule file 335–337
background agents 335
scripting 209
searching for forms 29
security process (Format Control) 234
selecting objects 42
sequential numbering
data types 188
getnumb.fc 187, 188, 194
number file 194
fields 195
parameters 188, 297
prefixes 188, 196
simple 189
suffixes 188, 196
ServiceInfo 324
designing form 347
designing the form 347–348
subscribing to data 349
sizing forms 20
sizing the Properties Window 34
sort.array 173
example 173
startup screens 317, 322
stored queries
in agent records 330
predefined 327
style guidelines 135
Subformat tool 86
subroutines 183, 187, 198–199, 240
fields 241
parameters 242
process in Format Control 149
subscribing
dynamic data 337–??, 343–??
charts 339
designing the form 338–340
displaying 343–345
marquees 338
menus 341–342
SYSPUB 337, 347
defined 308
using 313
system load 308
system status 333

T

tab pages 104
Table tool 94
tabs
navigating 104
Text Box tool 47
timer tool 118
Tools Palette
Bevel tool 54
Button tool 50
Chart tool 91
Check Box tool 70
Combo Box tool 78
ComFill tool 108
Fill Box tool 63
Frame tool 56
Gfile tool 114
Graphic tool 67
introduction to 41
Label tool 44
Marquee tool 83
Multi-line Text Box tool 60
Notebook tool 102
Pointer tool 42
Radio Button tool 73
Subformat tool 86
Table tool 94
Text Box tool 47
Wrapping Label tool 104
tools palette 21
OLE container tool 122

timer tool 118
training services 15

U
user interaction 307

V
validating
formats 273
validations process (Format Control) 148, 213
examples 215
fields 214
views process (Format Control) 149, 258
fields 260

W
workflow 308–309
Wrapping Label tool 104

X
X coordinates 19

Y
Y coordinates 19

May 28, 2003