

Peregrine

ServiceCenter

System Tailoring, Volume 2

Release 5.1

Copyright © 2003 Peregrine Systems, Inc. or its subsidiaries. All rights reserved.

Information contained in this document is proprietary to Peregrine Systems, Incorporated, and may be used or disclosed only with written permission from Peregrine Systems, Inc. This book, or any part thereof, may not be reproduced without the prior written permission of Peregrine Systems, Inc. This document refers to numerous products by their trade names. In most, if not all, cases these designations are claimed as Trademarks or Registered Trademarks by their respective companies.

Peregrine Systems® and ServiceCenter® are registered trademarks of Peregrine Systems, Inc. or its subsidiaries.

This document and the related software described in this manual are supplied under license or nondisclosure agreement and may be used or copied only in accordance with the terms of the agreement. The information in this document is subject to change without notice and does not represent a commitment on the part of Peregrine Systems, Inc. Contact Peregrine Systems, Inc., Customer Support to verify the date of the latest version of this document.

The names of companies and individuals used in the sample database and in examples in the manuals are fictitious and are intended to illustrate the use of the software. Any resemblance to actual companies or individuals, whether past or present, is purely coincidental.

If you need technical support for this product, or would like to request documentation for a product for which you are licensed, contact Peregrine Systems, Inc. Customer Support by email at support@peregrine.com.

If you have comments or suggestions about this documentation, contact Peregrine Systems, Inc. Technical Publications by email at doc_comments@peregrine.com.

This edition applies to version 5.1 of the licensed program.

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	The Document Engine	17
	Main RAD Applications and Flow	18
	The Doc Engine Flow	20
	Accessing the Document Engine	21
	Objects	22
	Fields on the Object Definition panel	23
	The Object Info Tab.	24
	The Locking/Revisions Tab.	26
	Variable/Global Lists Tab	27
	The Alerts Tab	28
	The Approvals Tab	30
	The SC Manage Tab.	32
	The Inboxes Tab	34

States	35
Creating States	35
Fields on the State Definition panel	37
Processes	37
Creating Processes	37
The Process Definition Panel	39
The Initial Expressions tab	40
The RAD tab	40
The Final Expressions tab	41
The Next Process tab	42
Standard Variables	42
Example: Add a Menu Option to the Options Menu	44
Chapter 2 Validity Table Processing	49
Accessing the validity File.	50
Menu Access	51
Database Manager Access	53
Command Line Access.	54
Field Definitions	56
Standard Info Tab	56
Secondary File Query Info tab	62
Alternate Application tab	66
Value Summary Detail	67
Field Definitions	69
Range Summary Detail.	70
Field Definitions	73
Creating Validity Table Definitions	75
Task 1: Add Field Level Definitions	75
Task 2: Add Value Definitions	78
Task 2: Add Range Definitions	79
Task 4: Validate the Validity Definitions	82
Deleting Validity Record Components	84
Printing Validity Definitions	86
Invoking Validity Table Processing.	88
Validating Fields During Record Processing.	88
Look-up Processing	93

	Examples	98
	Validity Table Specifications	100
	Invoking Validity with Format Control.	114
	Invoking Validity with a displayoption	115
	Example 1: Run-Time Validation	115
	Example 2: Assisted Entry	121
	Example 3: Validity Lookup Option	124
	Example 4—Special Processing	127
	Example 5—Basic Array Validation	136
	Additional tips	142
Chapter 3	Notification Engine	143
	The Notification File	144
	Editing a Notification Record.	148
	Adding new fields to a Notification Record	150
	Calling Notification from Format Control	153
	Messages	154
	Message Type	156
	Fields on the Message Type Panel	157
	On Call Scheduling	157
	Fields on the Oncall Schedule Panel	158
	Creating an oncall schedule.	159
	Modifying an oncall schedule.	160
	Oncall Flow	162
	Daily On Call Records	163
	Example: On Call Schedule.	164
	Exceptions.	167
	Default Notification Definitions.	167
Chapter 4	Global Lists / Global Initer	171
	Starting the Server Side Component of the Global Initer:	172
	Moving a Global List to a client:.	172
	Global Initer Considerations:	172
	Lister	173
	List records	173
	Field Definitions	175

	Advanced fields	176
	Global Lists	177
	Binding a List to a Form	177
	Returning a List to Your Client	178
	Building lists on startup	178
	Configuring with Format Control	180
	Troubleshooting Lister	181
	Verify Lister Status and Configuration	182
	Regenerate Obsolete Lists	185
	Regenerate All Lists	186
Chapter 5	Online Help	191
	Viewing Online Help	192
	Help Record Fields	193
	User views	197
	Accessing the <i>help</i> File	198
	Database Manager	199
	Command Line	199
	Toolbar Button	200
	Help Menu	200
	Types of Help	201
	Field—File Specific	202
	Field—Form Specific	203
	Field Contents	203
	Form	205
	Term	205
	Creating Online Help	206
	Related Information	208
	Creating Keyword Records	209
	Displaying Related Information	211
Chapter 6	The Cascade Update Utility	213
	Cascade Updates	213
	Creating a Cascade Update	214
	Creating a Trigger Record	214
	Create a Cascade Update Configuration Record	216

	Example	219
Chapter 7	Display Options	223
	Creating Options	223
	Sorting a column	224
	Opening an external application	231
	Balloon Help	236
Chapter 8	The Display Application	237
	Accessing Display Records	239
	Displaymaster File.	240
	Displayscreen File	241
	Showing/Hiding Display Screen Names	241
	Main structure	242
	Initialization tab	243
	Options tab	244
	Events tab	246
	Displayoption File.	247
	Fields	248
	RAD tab.	249
	Comments tab	250
	Database Dictionary keys.	250
	Displayevent File	251
	Fields	251
	Database Dictionary keys.	253
	Creating a displayscreen record	254
	Procedures for creating displayoptions	254
	Restricting Access	255
	Selecting the option	256
	Defining a capability word	257
	Display conditions	258
	Adding operator capability	260
	Testing the display option	262
	Calling an Application	264
	Threading	264
	Scripts	266

	Creating Displayscreen Records: Advanced Topics	268
	Custom RAD	269
	Script forms	270
Chapter 9	Advanced Operations	273
	Copying a Database Dictionary	273
	Copying Database Dictionary record only	274
	Copying Database Dictionary record and data records	275
	Renaming a Database Dictionary Record	277
	Deleting a Database Dictionary Record	278
	Resetting the Database and All Records.	278
	Information Pooling.	281
	Modifying pool settings	282
	Searching for a Field	283
	Creating Subtables from an Array of Structures	284
	Organization in the Database Dictionary	285
	How to create subtables from an Array of Structures	285
	Actions	288
Chapter 10	Adding and Modifying Fields in the Database Dictionary	291
	Adding a New Field	291
	Forming field names	292
	Scalar fields	292
	Arrays.	295
	Structures	298
	Fields within structures	300
	Alias Fields.	308
	Adding an alias field.	309
	Adding a Key.	315
	Adding a key as the first key	315
	Modifying a Field	324
	Modifying field types: character (scalar) <-> array	325
	Changing data types: character <-> number	328
	Modifying a Key	331
	Modifying keys: nulls & duplicates <-> unique	331
	Modifying keys: single <-> concatenated	333

	Deleting a Field	335
	Deleting a Key	337
Chapter 11	Building the Calendar	341
	Preparing the Files	344
	Building the Calendar	345
	Processing rules	346
	Viewing the Calendar	348
Chapter 12	Calendar Forms	349
	Accessing the Calendar Menu.	349
	Overview of Calendar forms	350
	Daily Hours	350
	The Duty Hours Form	354
	Creating a new Duty Hours record	357
	Updating an existing entry	359
	Deleting an existing entry	359
	Holidays.	360
	The calholidays database.	362
	Creating a new holiday entry	362
	Updating an existing holiday entry	364
	Deleting an existing holiday entry	365
	Creating a new Holiday table	366
Chapter 13	Calculating Dates and Times	367
	The Alert Date Calculation Process	367
	RAD subroutine example	368
	Format Control example.	369
	Alert Date calculation examples.	370
	The Date Interval Calculation Process	373
	RAD subroutine example	373
	Format Control example.	374
	Date interval calculation examples.	376
	Index	379

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

For more information and help with this new release or with ServiceCenter in general, contact Peregrine Systems' Customer Support.

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) (800) 960-9998 (US and Canada only, toll free)
+(1) (858) 794-7428 (Mexico, Central America, 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 The Document Engine

CHAPTER

The Document Engine provides:

- Consistent behavior across modules for standard actions, such as list, view and search.
- A central point for setting privileges and behavior across modules.
- A means to customize ServiceCenter without touching the RAD layer of the product.

The Document Engine comprises Objects, States, and Processes to centralize tasks. The Document Engine controls behaviors with Objects. An Object is referenced whenever a form is opened and determines behavior for the state of the form. (open, list, search, etc.) Implementers can change behavior easily for standard actions across ServiceCenter in one place with one Object.

The relationship between Objects, States, and Processes is hierarchical:

- Objects handle States and Processes, which are called from the module. Objects are overall file behavior definitions.
- States define how a record is displayed and what options are available at specific times and circumstances. For instance, States can determine an action, such as Save, given a user's access privileges. Processes are called by States.
- Processes use RAD expressions to perform work. Processes modify information or perform an action on records.

Implementers can use the Processes shipped with ServiceCenter to build their own States and Objects.

Peregrine ships default Objects, States, and Processes. The Document Engine is designed to meet the needs of most customers out of the box, yet still retain flexibility. Before making changes to any Objects, States, or Processes, Peregrine recommends making a backup of the files you modify.

Main RAD Applications and Flow

When working with a database driven application, there are three basic record sets that the user will be viewing at any one time:

- Zero records – when searching for information.
- Many records – when looking at a list of information.
- One record – when making changes to a single record.

The three main RAD applications used by the Document Engine mirror this idea.

se.search.engine

The search engine is used when there are no records being viewed. The main purpose of this routine is to formulate a query and select records from the correct database file. This routine may also be used for the initial entry of information into a blank record for the purpose of adding a new record to the database.

se.list.engine

The list engine is used to display multiple database records. Using the list engine a user may select a specific record from the list, or perform actions on the entire list of records.

se.view.engine

The view engine is used to display a single database record. This application is used to perform actions against a specific record, such as updates or deletes.

The RAD routine used is simply determined by the number of records being acted on at any time.

Note: When using ServiceCenter's record list functionality, the view engine is used for both the list and single record information.

The Doc Engine Flow

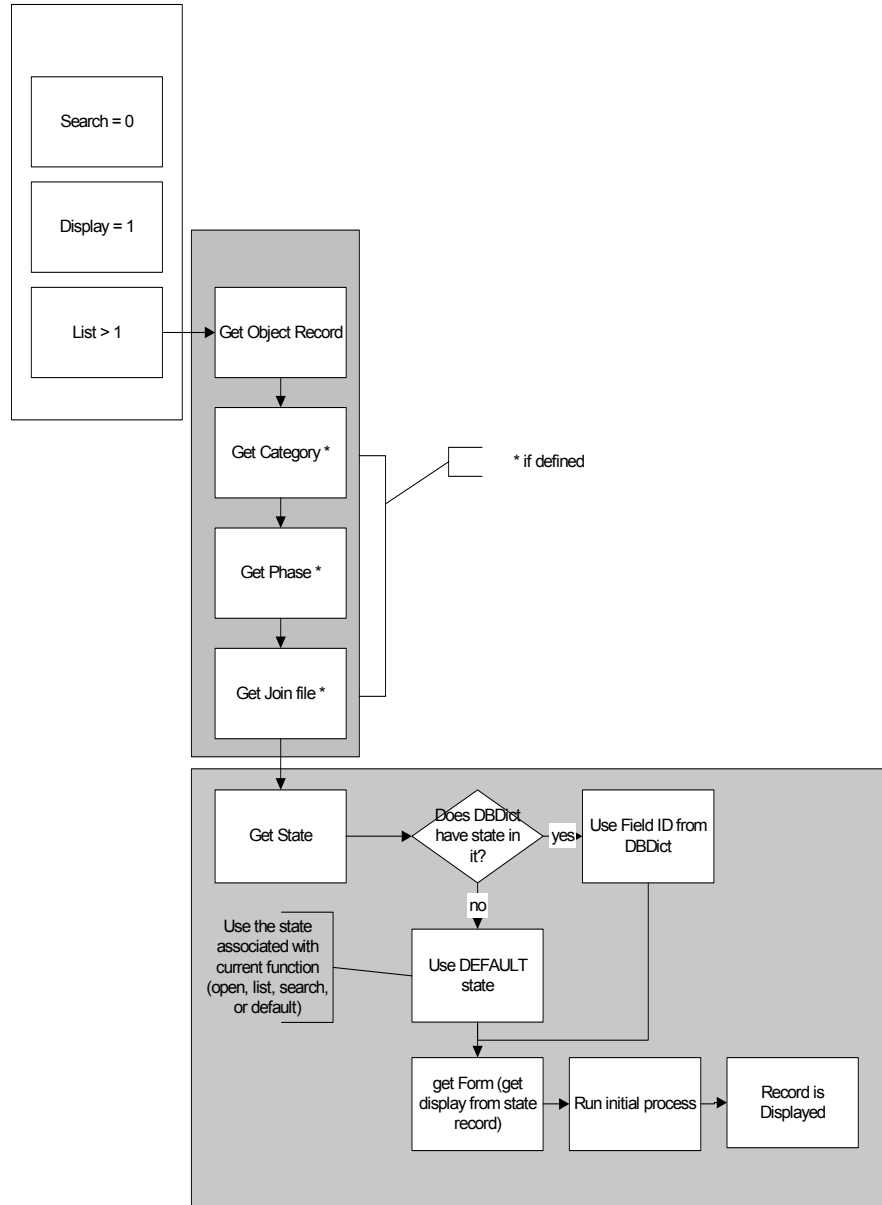


Figure 1-1: Doc Engine Flow Part 1 (Before the record is displayed)

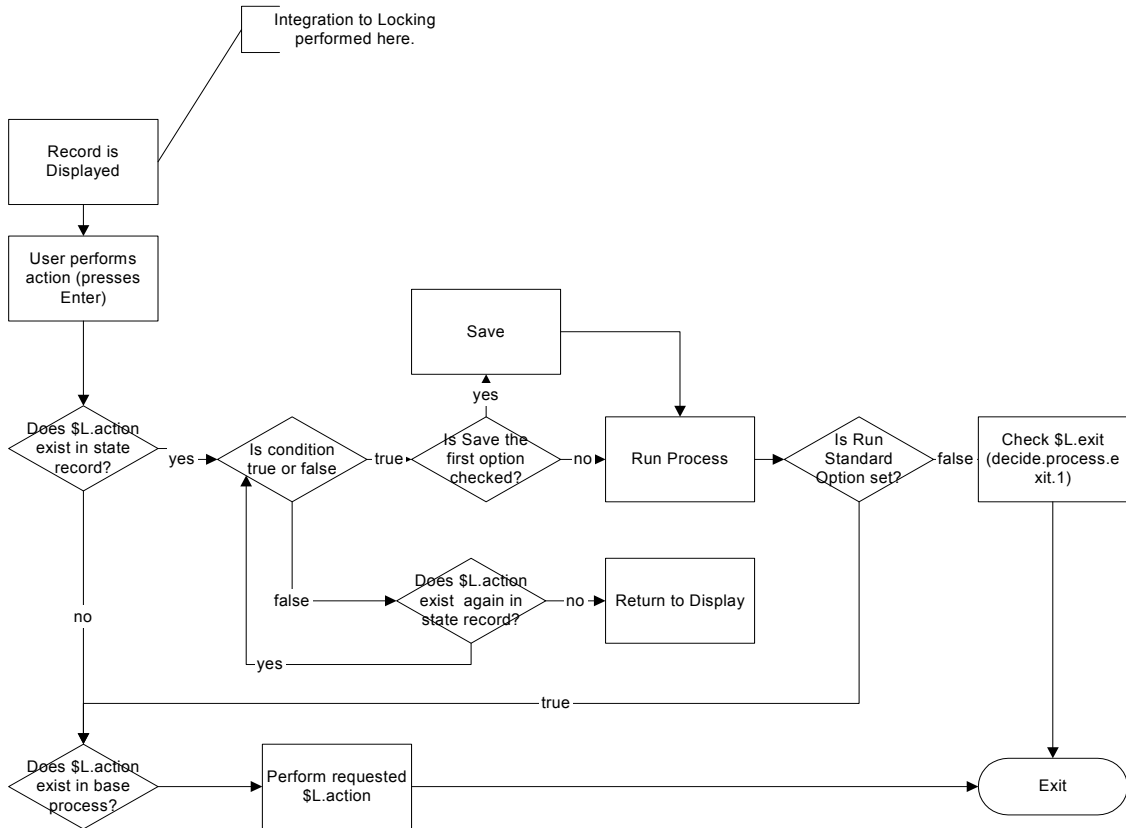


Figure 1-2: Doc Engine Flow Part 2 (After the Record is Displayed)

Accessing the Document Engine

To access the Document Engine:

- 1 Start a ServiceCenter Client and log in as an administrator.
- 2 Click the Utilities tab in the ServiceCenter main window.
- 3 Click the Tools button.
- 4 Click the **Document Engine** tab. From here, you can access the three areas that control the Documentation Engine: select either **Objects**, **States**, or **Processes**.

Objects

Objects are a base set of definitions that determine behavior of screens and panels and set the definitions and governing rules for behavior. Objects should correspond on a one-to-one basis with dbdicts. Each file within ServiceCenter can have an associated Object record, although it is not required. If a file does not have an associated Object record, the Default Object is used.

- Objects have a one-to-one relationship with database dictionaries.
- Database dictionaries without a corresponding Object record will use the DEFAULT Object, and will have standard database functionality.
- All Objects should have an open, list and default State defined.
- If a record has a field called “State,” then the value of that field will override the default state.

The Object record sets up the definitions and governing rules for the behavior of the file within the Document engine.

The definitions that are set within the Object file include:

- The ServiceCenter file name and unique key structure.
- The application used to create the object’s profile (which determines what actions may be taken against a record by an operator).
- The State records used in specific circumstances (see the States section for more details).
- The category, phase, and paging file names for the object (if used).
- The name of the number record to be used for this object.
- How locking is to be used by this object.
- Which variables should be available to processes that run against this object.
- Which global lists should always be available when using this object.
- How alerts are processed against this object (if any).
- How approvals are processed for this object (if any).

Objects Shipped with ServiceCenter

For a list of the Objects shipped with ServiceCenter, perform a true search from the Objects panel.

Creating Objects

To create an Object:

- 1 Access the Document Engine panel. See *Accessing the Document Engine* for steps.
- 2 Click Objects. The Objects panel opens.
- 3 Using the tabs on the Objects panel, fill in the fields required to create an Object that will perform the functions you desire. See the field descriptions that follow.

Figure 1-3: The Object Info tab

Fields on the Object Definition panel

There are six tabs on the Objects panel, and three fields not associated with a tab:

File Name - Enter a file name for the object. We recommend using the dbdict name that this object corresponds to.

Common Name - Enter a common name for the object. The common name can be a simple name, such as request, or a number that corresponds to a number in the sc.message file

Unique Key - The Unique Key for the Object should match the corresponding dbdict unique key field.

The Object Info Tab

General properties and Object behavior are specified with the fields under the Object Info tab.

Description Field - Enter a short description of the Object, if desired.

Profile Application - Enter the RAD application or profile that determines if a user can perform certain functions, such as add, delete, and so on. For example, rm.environment.

Profile Variable - This field is optional. Enter a variable that can be accessed any time this Object is called, without accessing the environment record. For example, \$G.ocmq.environment.

Number Record Name - This field is optional. Define a number class inside the object, which can be referred to in a Process or RAD code.

Category File Name - Enter the file name that links with the category file associated with this Object, if applicable. When displaying a record of this type, if a field called category exists, then the Object will go to the Category file and select a record with a corresponding name. If found, the system will store the Category File Name as a variable: \$L.category

Phase File Name - Enter the file name that links with the phase file associated with this Object, if applicable. When displaying a record of this type, if a field called phase exists, then the Object will go to the Phase File and select a record with a corresponding name. If found, the system will store the Phase File Name as a variable: \$L.phase.

Paging File - Enter the name of the file to store pages in. Pages are created every time a record is updated, creating a detailed audit trail.

Master Format Control - Enter the name of the master format control record, if one exists for the record. Master Format Control allows you to define in one record the Format Control statements that apply to all phases in an area, for example Change Management Request Phases.

Joindef - Join Definition, if used, or variable pointing to a join. For example, "joindef in \$L.category".

Open State - Enter the State to use upon opening.
For example, rmq.view.

Close State - Enter the state to use upon closing.

List State - Enter the State to use for listing results.
For example, rmq.list.

Default State - Enter the default State of the Object.
For example, rmq.view.

Search State - Enter the State to use for searching.
For example, rmq.search.

Browse State (text mode only) - Defines the State to use when records use locking. Essentially, this field defines a read-only State.

Manual State - List all possible states other than default.

The Locking/Revisions Tab

The Locking/Revisions panel is used to determine the locking behavior for the Object, as well as revision behavior.

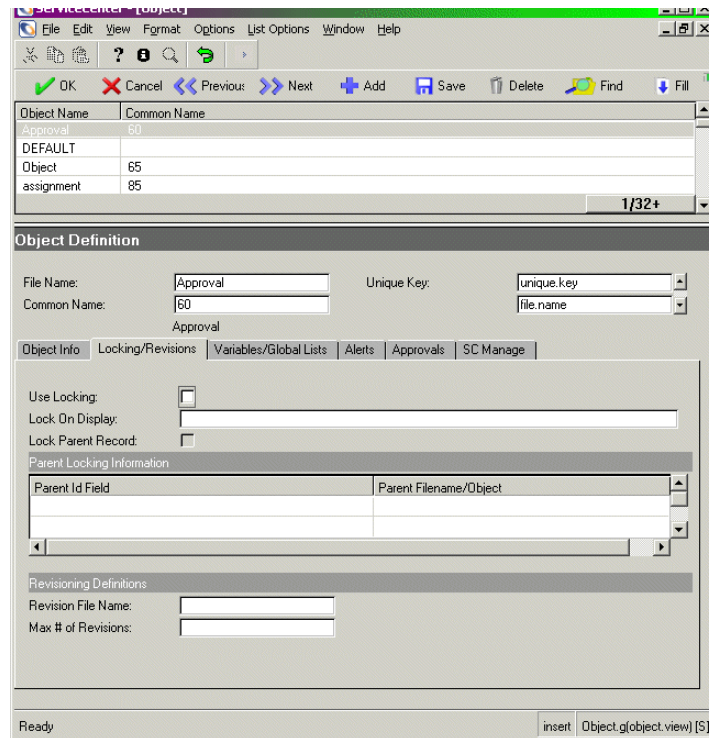


Figure 1-4: The Locking/Revisions tab

Use Locking - Select this check box to enable locking.

Lock on Display - Enter a condition that evaluates to True or False. The system will attempt to lock the record as soon as it is brought up. **Use Locking** must be enabled for this field to work.

Lock Parent Record - Locks the current record, and the record's parent.

Parent Id Field - Enter a field name in the current record that contains the ID of the parent.

Parent File Name/Object - The name of the file that contains the parent record.

Revision File Name - Revisions will be saved to the file name you enter here.

Max # of Revisions - The total number of revisions allowed for this Object. If left blank, then there is no upper limit.

Variable/Global Lists Tab

The Variable/Global Lists Tab is where you set local variables and global list properties. Global lists are built and stored in memory and available to the Object for as long as the Object persists.

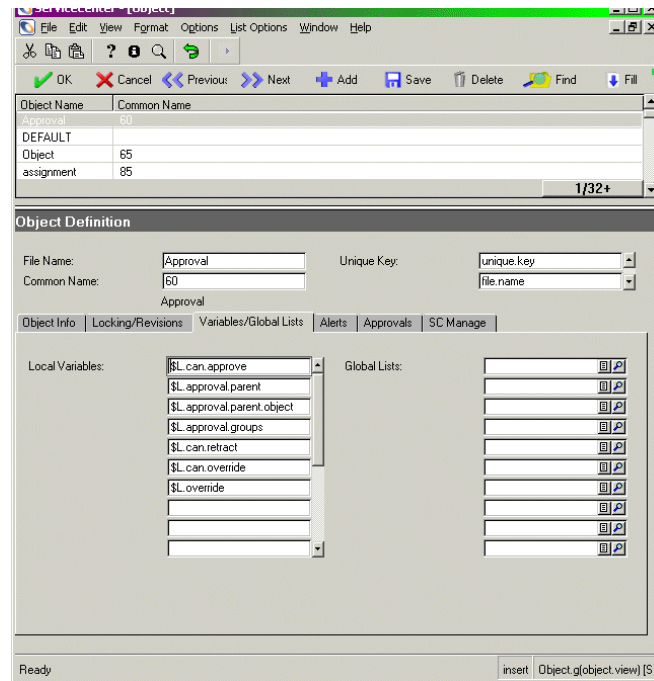


Figure 1-5: The Variable/Global Lists Tab

Local Variables - Enter a list of local variables that are not part of the base function, but can be used in Processes. Local variables are defined by the user and must exist prior to being called. Local variables are assigned to the Object you are creating and are available to all Processes and States associated with the Object. Local variables are not available to other Objects.

Global Lists - Global lists, once created, are available to all Processes. Global lists can be run on login, if they exist in the global list file. Global lists are available every time you access the Object. See *Global Lists* on page 177 for more information.

The Alerts Tab

The Alerts tab is where you set variables and conditions for alerts. Any object that has a unique key can use these alerts.

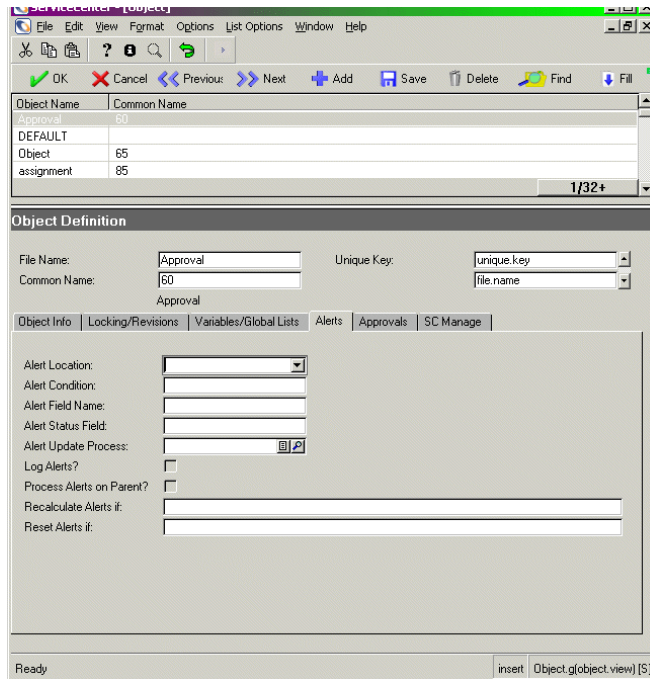


Figure 1-6: The Alerts Tab

Alert Location - Specify the location to store alerts. You can enter one of the the following:

- **Phase:** Store alerts in the Phase record defined on the Object Info tab.
- **Category:** Store alerts in the category file defined on the Object Info tab.
- **Record:** Store alerts in the record itself.

Alert Condition - Enter a logical condition to determine whether or not to process the alert. For example, open in `$L.file~=false`

Alert Field Name - Enter the field name that contains the actual alert name, as defined by **Alert Location**.

Alert Status Field - Enter the field in the current record in which to put the alert status, after the alert is processed.

Alert Update Process - Enter additional functions that the system will perform after the alert runs.

Log Alerts? - If selected, then alerts are logged to the Alertlog file.

Process Alerts on Parent? - If you have selected the Locking Parent Record field on the Locking/Revisions tab and you select this checkbox, then when the alert is activated it will register against the parent record.

Recalculate Alerts If - Logical condition as to when to recalculate conditions on existing alerts.

Reset Alerts If - Logical condition as to when to delete existing alerts and recalculate all conditions from scratch

The Approvals Tab

The Approvals Tab is where approval and notification options are set for the Object. Approvals are defined in the ApprovalDef file.

The screenshot shows the 'Object Definition' dialog box with the 'Approvals' tab selected. The 'File Name' is 'Approval', 'Common Name' is '60', 'Unique Key' is 'unique.key', and 'file.name' is 'file.name'. The 'Approval' object is selected. The 'Approvals' tab contains the following fields:

- Approval Condition: [Text Field]
- Approval Location: [Dropdown Menu]
- Approval Field Name: [Text Field]
- Approval Status Field: [Text Field]
- Approval Groups: [Text Field]
- Approval Type: [Dropdown Menu]
- Approval FC: [Text Field]
- Approval Process: [Text Field]
- Denial Process: [Text Field]
- Preapprove On Open: [Text Field]
- Log Approvals?: [Checkbox]
- Recalculate Approvals if: [Text Field]
- Reset Approvals if: [Text Field]
- Approval Notification: [Text Field]
- Denial Notification: [Text Field]
- Retraction Notification: [Text Field]
- Final Approval Notification: [Text Field]
- Final Denial Notification: [Text Field]

Figure 1-7: Approval Tab

Approval Condition - Logical condition defining when to use approvals.

Approval Location - Where the approval information is stored: record, phase, object, or category.

Approval Field Name - Enter the field name that contains the actual approval name, as defined by Approval Location.

Approval Status Field - Enter the field in the current record in which to put the approval status after the record is approved or denied.

Approval Groups - Variable that contains the groups the current user must be in to issue approvals for this object.

Approval Type - There are four pre-defined approval types:

- **All must approve:** The record is approved when all members of the approving group issue an approval.

- **One must approve:-** The record is approved with one approval from any member of the approving group.
- **Quorum** - The record is approved as soon as a majority of the approving group indicate approval.
- **All must approve - immediate denial** - All approvers must approve the record. The first denial causes the status to change to Deny. All approvers do not need to register their approval action.

Approval FC - Enter the Format Control record to run when an approval is performed.

Approval Process - Enter the Process that runs when the record is approved.

Denial Process - Enter the Process that runs when the record is denied.

Preapprove On Open - This field determines whether or not the record should be automatically approved. If the user belongs to one of the pending approval groups, the approval is processed automatically. If the user does not belong to one of the pending approval groups, the approval does not occur automatically, and must then go through the regular approval process

Enter true, false, or an expression that evaluates to either value. This field is true by default.

Log Approvals? - Select this check box to log approvals.

Recalculate Approvals If - Logical condition as to when to recalculate conditions on existing approvals.

Reset Approvals If - Logical condition as to when to delete existing approvals and recalculate all conditions from scratch.

Approval Notification - Enter the notification that will run when an approval is performed.

Denial Notification - Enter the notification that will run when a denial is performed.

Retraction Notification - Enter the notification that will run when retracting a previous action.

Final Approval Notification - Enter the notification that will run when the record is approved.

Final Denial Notification - Enter the notification that will run when the record is denied.

The SC Manage Tab

The screenshot shows the 'SC Manage' tab in a software interface. The tab is active, and the following fields are visible:

- SC Manage Condition: true
- SC Manage Display Format: sc.manage.generic
- SC Manage Default Inbox: All my Approvals
- SC Manage Default Query: [Empty]
- Thread Inbox -> Search?: [Empty]
- Thread Search -> List?: [Empty]
- Thread Inbox -> Edit?: [Empty]
- Allow Add Condition: false
- Add/Open Application: [Empty]
- Parameters table:

Names	Values

The status bar at the bottom indicates 'Ready' and 'insert Object.g(object.view) [S]'.

Figure 1-8: The SC Manage tab

The fields on the SC Manage tab control Queues and how they are displayed as well as threading and who can create inboxes. These same fields can be found in Data Policy, for files that do not have an associated Object record.

SC Manage Condition - Enter a condition that allows only certain users to add queues as inboxes. For example, `index("SysAdmin", $lo.ucapex)>0`

SC Manage Display Format - This field determines how to display queues for data on a file by file basis. ServiceCenter ships with a default display format: `sc.manage.generic`. Peregrine recommends you do not change the `sc.manage.generic` file.

SC Manage Default Inbox - State the default inbox for this queue. By specifying a user inbox record for a particular user, a specific list of inboxes can be set up for the SC Manage queues. If a user does not have a record, the DEFAULT user inbox record is used.

SC Manage Default Query - Enter a default query to run, if no default inbox was selected.

Default Query Description - Enter a description of the above field. You can associate a message with this field. For example, `scmsg(492, "us")`.

Thread Inbox -> Search? - Enter true, false, or an expression that evaluates to either to specify whether or not a new thread is opened when a user moves from an inbox to a Search screen.

Search Format - Enter a default search format.

Thread Search -> List? - Enter true, false, or an expression that evaluates to either to specify whether or not a new thread is opened when a user moves from a Search screen to a List.

Thread List -> Edit? - Enter true, false, or an expression that evaluates to either to specify whether or not a new thread is opened when a user moves from a List to an Edit screen.

Thread Inbox -> Edit? - Enter true, false, or an expression that evaluates to either to specify whether or not a new thread is opened when a user moves from an Inbox to an Edit screen.

Allow Add Condition - Enter an expression that evaluates an operator's ability to add a record.

Add/Open Application - Fill in the name of the application to call when a record is added or opened.

Parameters

- **Names** - Enter the parameter names to be passed to the application specified in the Add/open Application field.
- **Values** - Enter parameter values to be passed to the application specified in the Add/open Application field.

The Inboxes Tab

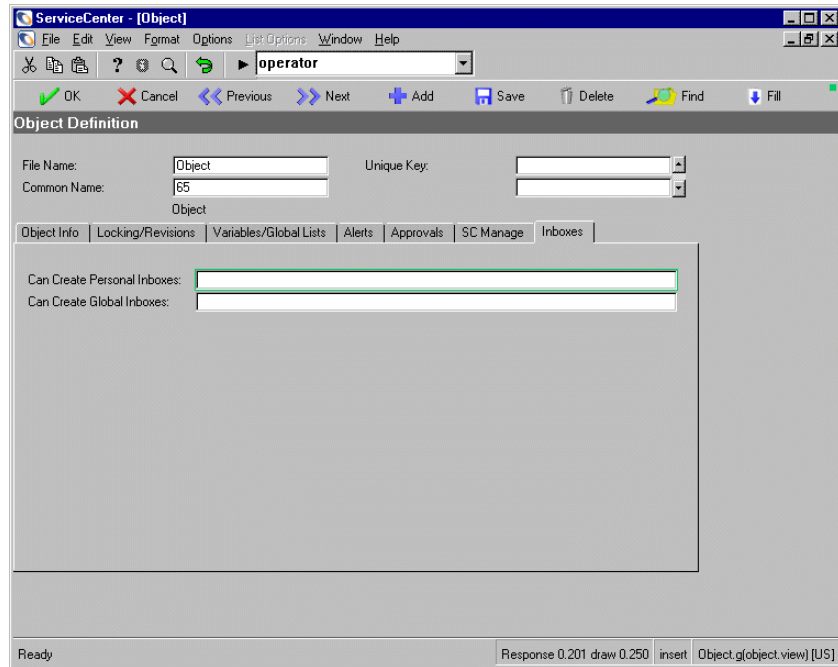


Figure 1-9: The Inboxes Tab

Can Create Personal Inboxes - Allows the user or group to create personal inboxes for their own use. Enter true, false, or an expression that evaluates to true.

Can Create Global Inboxes - Allows the user or group to create global inboxes for all Service Management users. Enter true, false, or an expression that evaluates to true.

For more information on creating inboxes, see the *ServiceCenter User's Guide*.

States

States are called by Objects, and defined by Processes. ServiceCenter ships with over 40 pre-defined States.

The State record contains information on how a record looks and acts at a specific period in time.

The definitions that are set in a State record include:

- The State name.
- Which display screen to use in order to display the record(s).
- What format to use when displaying the record(s).
- Whether or not a user can modify the record (input condition).
- An initialization process that may be run when a record or list is first displayed.
- What process to run when a specific display option (action) is triggered by the user

Creating States

To create a new State:

- 1 Access the Document Engine. See *Accessing the Document Engine* on page 21 for steps.
- 2 Using the tabs on the States panel, fill in the fields required to create a State that will perform the functions you desire. See the field descriptions later in this chapter.

To modify an existing State:

- 1 Access the Document Engine. See *Accessing the Document Engine* on page 21 for steps.

- 2 Enter the name of the State you want to modify in the State field, or press Search to search for the State.

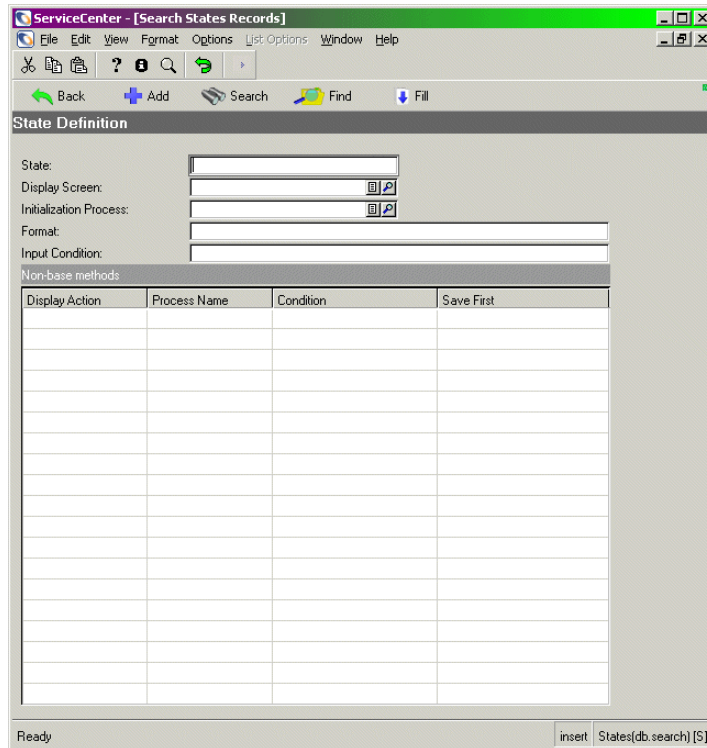


Figure 1-10: The State Definition Panel

Fields on the State Definition panel

State - Enter the name of the State.

Display Screen - Enter the Display screen to associate with the State.

Initialization Process - Enter the name of a Process to run prior to running the State.

Format - Enter the Format that the record will be displayed in. This format is stored as a phase or category, and can be a variable.

Input Condition - The Input Condition determines whether or not the record is read only. Enter False for read only condition, otherwise enter True.

Display Action- The action parameter that comes from a display action, after input from user.

Process Name - The Process that the State will call.

Condition - Enter an expression. If True, then the process will run.

Save First - Choose to run the save Process, before you run this State. Enter True to save first, otherwise enter False.

Processes

Processes are the smallest discreet units of work available to the Document Engine and are the level where the data is manipulated. Users can create their own Process or use one of the 80 Processes that ship with ServiceCenter.

The Process panel consists of pre-RAD, RAD, and post-RAD expressions, entered on the Initial Expressions, RAD, and Final Expressions tabs, respectively.

Expressions are written using standard ServiceCenter expressions.

Creating Processes

To create a Process:

- 1 Access the Document Engine. See *Accessing the Document Engine* for steps.

- 2 Using the tabs on the Process panel, fill in the fields required to create a Process that will perform the functions you desire. See the field descriptions later in this chapter.

To modify an existing Process:

- 1 Access the Document Engine.
- 2 Enter the name of the Process you want to modify in the Process Name field, or press Search to search for the Process.

The Process Definition Panel

The Process panel is where you define new Processes, or edit existing Processes. There are four fields not associated with a tab:

The screenshot shows a 'Process Definition' panel with the following fields and controls:

- Process Name:** A text input field containing 'add.contact.record'.
- Save Cursor Position?:** A checkbox that is currently unchecked.
- Run Standard Process when complete?:** A checkbox that is currently unchecked.
- Run in Window?:** A checkbox that is currently unchecked.
- Window Title:** An empty text input field.

Figure 1-11: The Process Panel

Process Name - The name of the process.

Save Cursor Position? - Check this box if you want the cursor to remain at its insertion point after the process is run.

Run Standard Process when complete? - If selected, then the system runs a standard process after completing the current action or Process. A standard process could be save, for example. If you have created a save process and want to run the save process that comes with the Document Engine after completing the save process you defined, check this box.

Run in Window? - Check this box if you want the process to run in a separate window.

Window Title: - If you checked the **Run in Window?** check box, type in a title for the window.

The Initial Expressions tab

The screenshot shows the 'Initial Expressions' tab selected in a software interface. The main area contains a table with 10 empty rows for entering expressions. The status bar at the bottom shows 'Ready' and 'Response 0.71 draw 0.110 insert Process.view_g(db.search) [US]'.

Figure 1-12: The Initial Expressions tab

Initial expressions run prior to the RAD code defined on the RAD tab and are written using standard ServiceCenter expressions. Enter any initial RAD expressions on the first available line of the table.

The RAD tab

The screenshot shows the 'RAD' tab selected in a software interface. The main area contains several input fields and a table. The 'Expressions evaluated before RAD call' field is highlighted with a green border. Below it are fields for 'RAD Application' and 'Condition'. A table with two columns, 'Parameter Names' and 'Parameter Values', is present. Below the table are 'Post RAD Expressions' fields. The status bar at the bottom shows 'Ready' and 'Response 0.71 draw 0.110 insert Process.view_g(db.search) [US]'.

Figure 1-13: The RAD tab

Expressions Evaluated before RAD call - Enter an expression to run prior to the RAD arrays defined later on this panel.

RAD Application - Enter the name of the RAD application to associate with this record.

Condition - Enter a condition associated with the RAD Application field.

Parameter Names - Enter the parameter names to be passed to the RAD application.

Parameter Values - Enter parameter values to pass to the RAD application.

Post RAD Expression - Enter a RAD expression that will run upon completion.

The Final Expressions tab

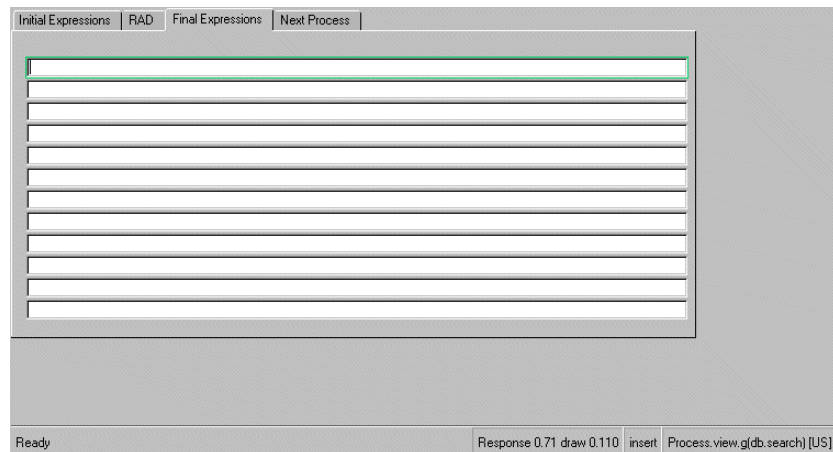


Figure 1-14: The Final Expressions tab

Final expressions are associated with the base action from the state record. Final expressions run after this process is complete and are written using standard ServiceCenter expressions.

The Next Process tab

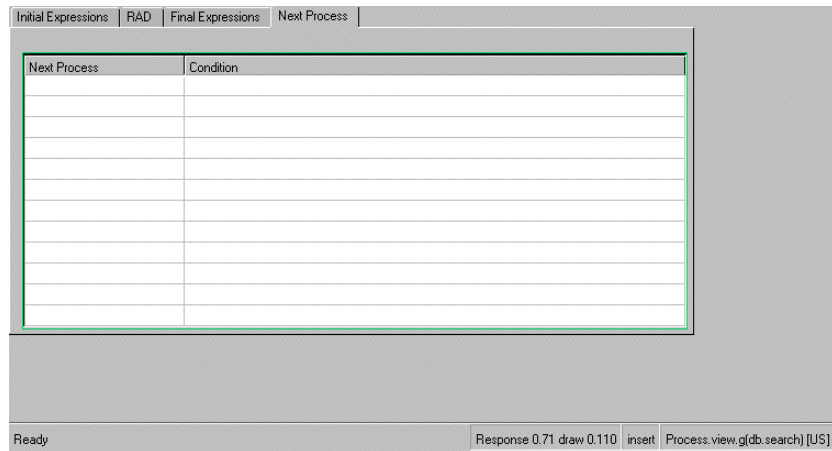


Figure 1-15: The Next Process tab

Next Process - Enter the Name of the next Process to run, if applicable.

Condition - Enter a condition associated with the Next Process field.

Standard Variables

The following is a list of available Standard Variables:

\$L.action - the display action value from the display option.

\$L.sql - the current query

\$L.sort - the current sort order

\$L.exit - internal exit parameter

\$L.file - The current file variable

\$L.mult - Flag that is true if there are multiple records in the \$L.file variable

\$L.env - The current environment record

\$L.object - The object record

\$L.file.save - A copy of the record in its original state

\$L.category - The category record (if available)

\$L.phase - The phase record (if available)

\$L.bg - Background flag

Variables that are available in View mode (when viewing a single record)

\$L.fc - associated format control record

Example: Add a Menu Option to the Options Menu

This example adds a menu option to the options menu of the `am.display.joinfile` format. The purpose of this menu option is to ping a device according to the IP Address in the record.

The basic flow for this example is demonstrated in the flow chart below. For each process step shown in the flowchart, a more detailed step-by-step procedure follows.

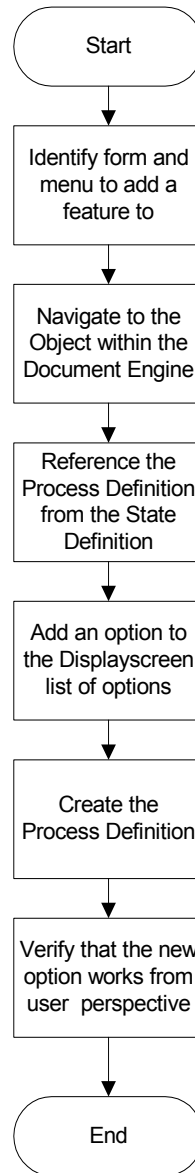


Figure 1-16: Example Process Flow

Identify the screen name and the form name:

- 1 Login as *administrator*.
- 2 From the **Services** tab, select **Inventory Management > Assets**.
- 3 Select **Screen Name** from the **View Menu**. A check mark appears next to **Screen Name** to indicate it is turned on. The screen name, **am.search**, is displayed in the lower right corner of the message line.
- 4 In the **Type** field, select **Computer** from the drop-down menu.
- 5 Click **Search**.

Identify the Object:

- 1 Press **CTRL+Shift+D+E** to open the RAD Debugger.
- 2 Type `d file.name in $L.object`.
- 3 Press **Enter**.

The word **device** appears in the debugger. Therefore, the file name of the object is **device**.

- 4 Click on the **X** in the upper-right corner to close the debugger.
- 5 Click **OK** to return to the **Asset Information** search screen.

Navigate to the object within the Document Engine:

- 1 Open a new session of ServiceCenter.
- 2 **Login** as *administrator*.
- 3 Select the **Utilities** tab.
- 4 Click the **Tools** button.
- 5 Select the **Document Engine** tab.
- 6 Click **Objects**.
- 7 Type **device** in the **File Name** field.
- 8 Press **Enter**.
- 9 Select **device** from the record list.

The **Object Definition** is displayed. Notice that the search state is **am.search**, which is the screen name from earlier in the procedure.

Reference the Process Definition from the State Definition

- 1 From the **Object Definition** record, click the magnifying glass next to the **Default State** field.

Note: If using a pre-5.0.1 client, click the triangle next to the **Default State** field.

- 2 The **State Definition** form is displayed.
- 3 Type `zping` at the bottom of the **Display Action** column.
- 4 Type `zam.zping.process` at the bottom of the **Process Name** column.
- 5 Type `true` at the bottom of the **Condition** column.
- 6 Click **Save**.
- 7 Click on the `zam.zping.process` field.
- 8 Click **Find**.
- 9 The **Process Definition** form is displayed.
Since ServiceCenter did not find the process name, it gives the user the opportunity to create one.

Create the Process Definition

- 1 Insert your cursor in the **Initial Expression** field.
- 2 Type `$L.command="ping;" + nullsub(network.address in $L.file, "0.0.0.0")`
- 3 Select the **RAD** tab.
- 4 Type `us.launch.external` in the **RAD Application** field.
Note: `us.launch.external` accepts the same type of commands as the Windows command line.
- 5 Type `true` in the **Condition** field.
- 6 Type `name` in the **Parameter** field.
- 7 Type `$L.command` in the **Parameter Values** field.
- 8 Click **Add**.
- 9 Click **OK**.

Add an option to the display screen list of options.

- 1 From the **State Definition** record, click the magnifying glass next to the **Display Screen** field.
- 2 The `am.display.joinfile` record is displayed.
- 3 Click the **Options** tab.
- 4 Scroll down the list to **Option 926**.

Notice that the Action for Option 926 is “do nothing”. Since the RAD Application, us.launch.external, performs everything that needs to be done to execute a ping, 926 is a good one to copy.

Also notice that Option 927 does not exist.

- 5 Click the **926** button.
- 6 The **display screen** record for **Option 926** is displayed.
- 7 Type **zping** in the **Action** field.
- 8 Replace **926** with **927** in the **GUI option** field.
- 9 Replace **926** with **927** in the **Text option** field.
- 10 Type **Ping this Device** in the **Default Label** field.
- 11 Delete the text in the **Condition** field.
- 12 Type **true** in the **Condition** field.
- 13 Delete the text in the **RAD Application** field.
- 14 Delete the text in the **Names** field.
- 15 Delete the text in the **Values** field.
- 16 Click **Add**.

A confirmation message appears in the message line.

Test the Document Engine Addition

- 1 Revert to the first session of ServiceCenter which is at the Asset Information search screen(am.search),.
- 2 In the **Type** field, type **Computer**.
- 3 Click **Search**.
- 4 Find a computer that has the **IP Address** field populated.
- 5 Insert your cursor in the **IP Address** field.
- 6 Select **Options Menu > Ping this Device**.

The command line appears and attempts to ping the IP Address from the Computer record. If the machine you are working on has network access, the ping will succeed. If it does not have network access, the ping will fail.

2 Validity Table Processing

CHAPTER

The validity table allows you to define a field's valid values in a list of values, a list of ranges, a secondary file, and customized RAD subroutines. These values are then used for validating operator-entered data and for *lookup* processing (similar to ServiceCenter Find and Fill). These various validity definitions are consolidated in one file, resulting in easy maintenance. This can be of significant benefit when multiple forms access one database.

The value of a field is validated against a list of values (e.g. 10, 20, 30, or 40), a list of ranges (e.g. 50 through 60), a secondary file (similar to the Format Control secondary file query) and customized RAD subroutines. The fields within a file are validated in a particular order (e.g. **fieldx**, **fielda**, **fieldg**), in alphabetical sequence (e.g. **fielda**, **fieldg**, **fieldx**) or according to a field validation sequence number. The default processing of the validation routine validates the value of a field against the definitions in the validity table. If the current value of a field is invalid, a summary list of valid values and ranges is presented to the user. If no values or ranges are defined for the field, a QBE list of records from the secondary file is presented. The user can then correct the invalid value and continue processing; return to the calling application; or force the application to accept the invalid value. The validity table also allows you to override this processing and execute customized RAD code.

After determining that a field contains a valid value or range, user-defined processing statements are executed. Validity processing continues to the next field in the validity definition table until all specified fields are validated.

Each validity table definition contains four levels of definition:

- Field Summary
- Secondary File Query Information
- Values Summary
- Range Summary

When it is necessary to differentiate the maintenance procedures of these different levels, the order of the documentation is Field level, Secondary Queries level, Values level, and Ranges level. For instance, the section on *updating* shows how to update Validation Records at the Field Level, the Value Level, and then the Range Level.

Important: All applications and procedures described in this chapter refer to a unique maintenance application that specifically supports validity table maintenance. If you access any of the validity table databases with any other application (such as Database Manager), you must ensure that all data is properly structured in order for the validity definitions to work correctly at run time. Peregrine Systems recommend that you always follow the procedures outlined in this chapter.

Accessing the validity File

You can access the validity file from three locations:

- Menu button
- Database Manager
- Command line

Menu Access

To access a validity record from the menu button

- 1 Select the Utilities tab in the system administrator's home menu.

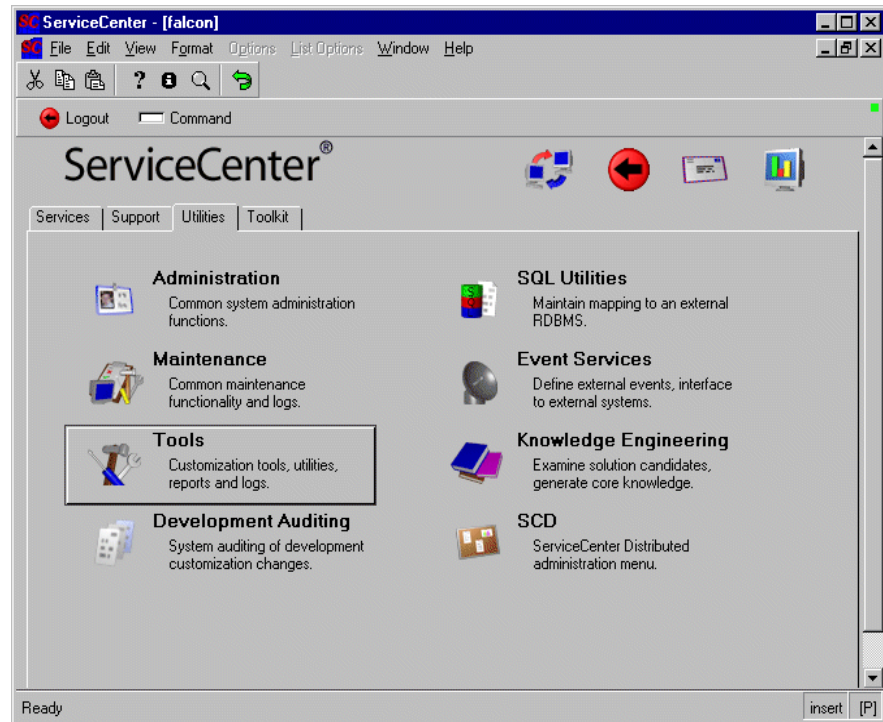


Figure 2-1: Utilities tab in the system administrator's home menu

- 2 Click Tools.

The Tools menu is displayed.

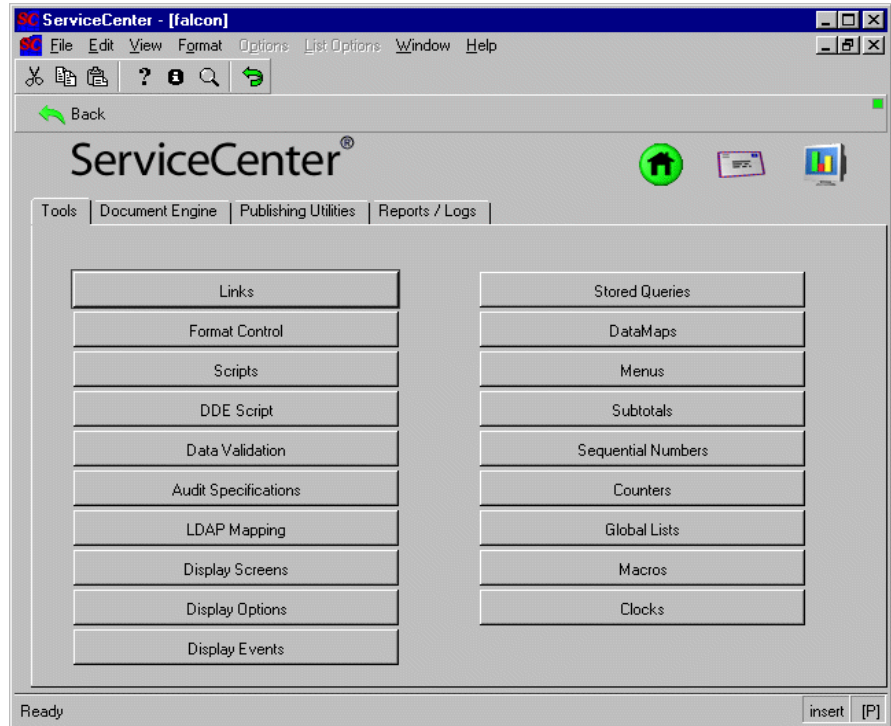


Figure 2-2: Tools menu in the system administrator's home menu

3 Click Data Validation.

The Validity Table Specifications form is displayed (Figure 2-5 on page 55).

Database Manager Access

To access a validity record from Database Manager

- 1 Select the Toolkit tab in the system administrator's home menu.

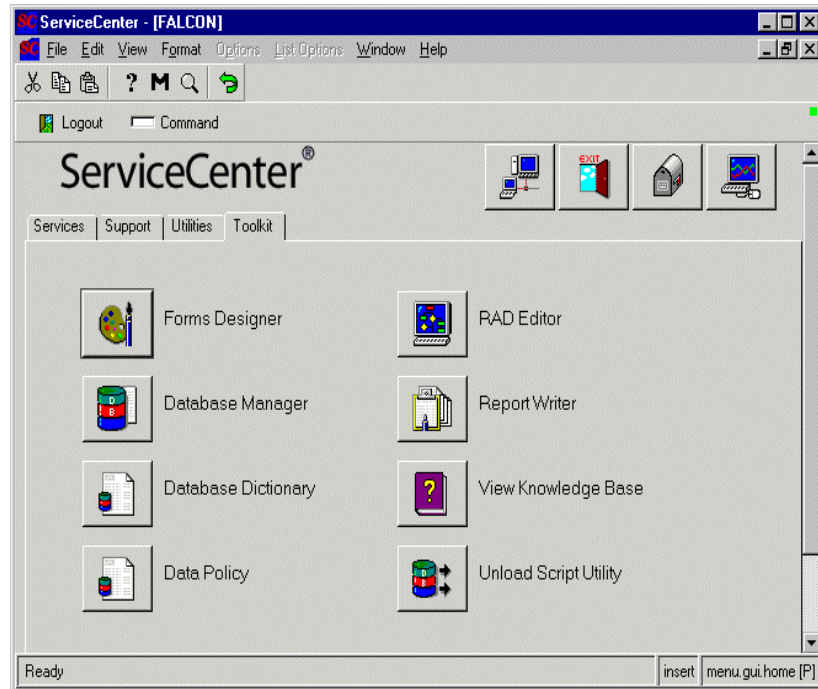


Figure 2-3: Toolkit tab in the system administrator's home menu

- 2 Click Database Manager.

- 3 Type validity in the File field of the Database Manager dialog box.

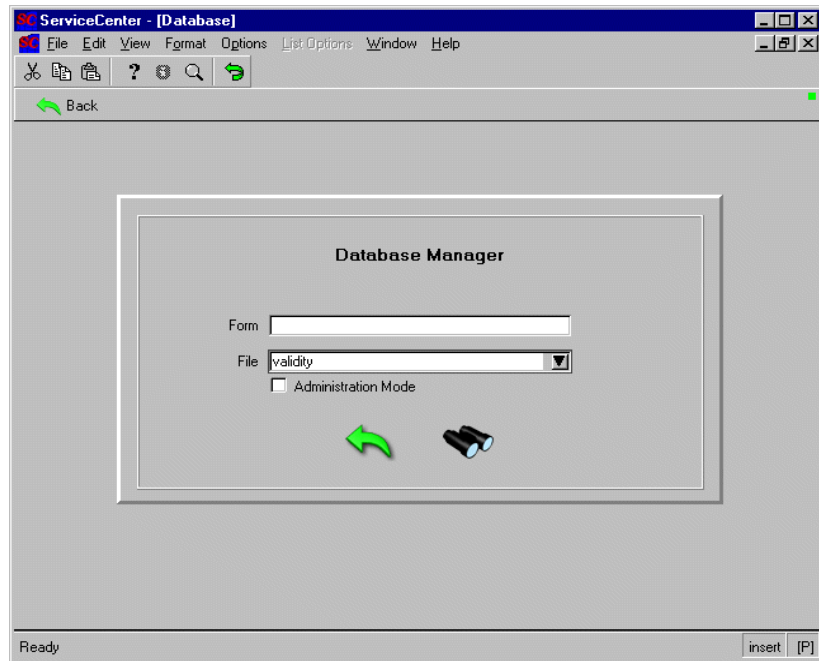


Figure 2-4: Database Manager dialog box

- 4 Click Search or press Enter.
The Validity Table Specifications form is displayed (Figure 2-5 on page 55).

Command Line Access



- To access validity records from a command line
- 1 Click Command in the system administrator's home menu.
 - 2 Type validity in the command line displayed.
 - 3 Press Enter.
The Validity Table Specifications form is displayed.
 - 4 Enter a field name or a file name in the form.
 - 5 Click Search or press Enter.

The requested record is displayed.

ServiceCenter - [Data Validation]

File Edit View Format Options List Options Window Help

OK Cancel Previous Next Add Save Delete Find Fill

Validity Table Specifications
- SUMMARY -

Field Name: type Sequence: 1
Files/Formats: upgrade Field Type: Character
Array?
Case:
Field Desc: Unique ID: upgrade

Standard Info Secondary File Query Info Alternate Application

Validate Cond: true
Allow NULL: false
NULL Default:
Bypass Cond: false
Weight Factor:
Error Msg: The type is not valid. Select one from the list.
Prompt Panel:

Values Summary Range Summary

application	
dbdict	
file	
module	

Ready insert [P]

Figure 2-5: Validity table specifications summary form

Field Definitions

Standard Info Tab

Field	Description
Field Name	<p>The name of the field within a particular table that is to be validated. This is a required field. Refer to <i>Validity Validation Rules</i> on page 82. You can use one validity table definition to validate the same field name that is defined in multiple files. For example, if five files (<i>cm3r</i>, <i>cm3t</i>, <i>problem</i>, <i>device</i>, <i>operator</i>) each contain a field named state with a total of 50 valid values, you can define state once in the validity table, and then update the Files/Formats field to contain the names of the five files. The data type of Field Name must be the same in each of the files named. Use the global variable <i>\$vrfield</i> to reference the value of Field Name in the record being validated. Reference all other data fields in the record being validated with the <i>\$val</i> file variable.</p>
Files/Formats	<p>The names of the files and forms to which this field validation applies. Normal validity table processing validates all fields associated with the file of the data record being validated; however, there are circumstances when only the fields defined to a particular form need to be validated and the remainder of the fields associated with that file should be ignored. When this occurs, you can define that form name here to change validity processing to validate only the fields associated with that form. This is done by passing the name of the form to the validation routine (see the text parameter value on page 90).</p> <p>The validity validation routine requires a minimum of one definition for this field. Each value must be a valid file or form name. The validation routine searches the <i>dbdict</i> file first. If an entry is not in the Database Dictionary, the routine searches the <i>format</i> file. If the entry is not a valid form name, a QBE list of valid file names is displayed. From this list you can invoke a search against the <i>format</i> file to select a valid form name.</p> <p>Note: If you select a form, it must be associated with a valid file and the Field Name must be defined in that file. If the form is not associated with a file, you are prompted to select one from a list of valid files. Refer to <i>Validity Validation Rules</i> on page 82.</p>

Field	Description
Field Desc	A brief description of this field. You may enter any word or phrase that best describes the field's function. This is an optional field.
Sequence	<p>The default field validation sequence for the fields within a file is by field name. Sequence allows you to override this order by defining numbers for each of the fields. It is an optional field.</p> <p>Note: You can also override the default sequence by passing a list of fields to process. Refer to <i>Invoking Validity Table Processing</i> on page 88.</p> <p>If the file <i>contacts</i> had validity table definitions for the fields contact.name, address, phone, state, and zip, then the default validation sequence is: address, contact.name, phone, state, zip (alphabetical sequence). To override this order, the definitions are updated to contain the following sequence numbers (which are shown in parenthesis): name (1), address (2), phone (1), state (2), zip (3). The validation sequence will now be: contact.name (1), phone (1), address (2), state (2), zip (3). The fields that contain the same sequence number are validated in alphabetical order. Also note that when a list of fields is passed to the validation routine, Sequence is ignored.</p> <p>The default value is NULL. In an MVS environment, if some of the fields have a number and others do not, the NULL sequence number fields are validated <i>before</i> the non-NULL sequence number fields. In a UNIX environment, the NULL sequence number fields are validated <i>after</i> the non-NULL sequence number fields.</p>

Field	Description
Field Type	<p>Data type of this field. Supported data types are:</p> <ul style="list-style-type: none">■ Number■ Character■ Date/Time■ Logical <p>The validity validation routine extracts this value from the Database Dictionary of the file named in the Files/Formats field, when the record is validated; however, it can be manually modified. Normally, manual maintenance is only required when a field's data type is changed in the Database Dictionary record. If a field's data type is changed, be sure to update the Values/Ranges to reflect the new data type.</p> <ol style="list-style-type: none">1 Update the appropriate Database Dictionary to reflect the correct data type.2 Logoff of ServiceCenter and log back on to ensure that you are working with the most current version of the Database Dictionary.3 Update the Values/Ranges definitions for the field whose data type has changed.4 Invoke the validation routine. <p>The value in the Field Type field is updated to reflect the new data type.</p>
Array	<p>A boolean field that indicates whether or not this field is an array. It is an optional field. The default is <i>false</i>. If the condition evaluates to <i>true</i> at run time, the validity routine validates each element of the array against the defined values, ranges, or secondary query. Refer to <i>Validating Arrays</i> on page 136.</p>

Field	Description
Case	<p>An optional field that converts the character data entered on the Validity Correction Screen to upper case (<i>uc</i>) or lower case (<i>lc</i>). If this field is NULL, the data is not altered. This field affects character fields only. It is ignored for all other data types. The validation routine assumes that the data passed to it is in the correct case and attempts to validate the data with its case setting as is. If the data does not validate, a Validity Correction Screen is displayed which shows the Values/Ranges that the user can select from. The corrected data entered by the user is in lower case. If the Values/Ranges for the field require upper case data, then a value of <i>uc</i> converts the operator entered data to upper case before it is re-validated. This same logic also applies to the <i>lc</i> setting.</p> <p>Use this field when the Field Name defined in the validity table is converted to upper case (<i>ctrl</i> setting of 256) or lower case (<i>ctrl</i> setting of 128) in the form in which data is entered for the file.</p>
Unique ID	<p>A required character field that, in combination with the Field Name and Sequence fields, uniquely identifies the validity record. If you leave this field blank at <i>add</i> or <i>update</i> time, a validation routine automatically copies the contents of the first element in the Files/Formats array to this field. Refer to Validity Validation Rules, page 82</p>
Validate Cond	<p>An optional field that defines the conditions that must exist before the validity table record is processed. If this field evaluates to <i>false</i> at run time, field validation is bypassed. If this field evaluates to <i>true</i>, field validation occurs. The default is <i>false</i>.</p>
Min Elements	<p>An optional field that defines the minimum number of elements in the array that must meet the validation requirements for this field. If the Array parameter is <i>false</i>, the validity routine ignores this field. The validity routine validates either the number of elements defined in this field, or the number of elements contained in the array at run time, <i>whichever is the greatest number of elements</i>. If Min Elements is 5 and at run time the array contains 2 elements, then the validity routine will validate 5 elements. If Min Elements is 5 and at run time the array contains 7 elements, the validity routine will validate 7 elements.</p>

Field	Description
Allow Null	A boolean condition that specifies if NULL values are allowed for the field. If the condition evaluates to <i>true</i> at run time, NULL values are allowed. If the condition evaluates to <i>false</i> (the default), NULL values are not allowed. It is an optional field. Operators with <i>SysAdmin</i> capability can enter NULL values even when Allow Null evaluates to <i>false</i> .
NULL Default	Defines the value to substitute for a field when original value is NULL. It is an optional field. The NULL Default value must be a Valid Value or Range. When a field is substituted with the NULL Default value, it is then processed through normal Validity table processing. Therefore, if the NULL Default it is not a valid Value/Range, it is treated as being in error, and the operator will have to correct it.
Bypass Cond	A boolean condition that specifies if the user will have the option of bypassing validity table processing when a field is in error. If the validation routine determines that a field contains an invalid Value/Range, the user has the option of forcing the application to use the current (invalid) value. The default is <i>false</i> . Users with <i>SysAdmin</i> capability have the option to bypass validity table processing even if the Bypass Condition evaluates to <i>false</i> .
Weight Factor	A numeric field used in Change Management processing to calculate the Risk Factor of a Change Request or Task. It is used with the Weight Value field of the Value/Range definitions. It is an optional field. Refer to <i>Change Management</i> in the Application Administration Guide for more information on the Risk calculation process.
Error Msg	<p>The message sent to the user in place of the default message. More meaningful error messages can be sent to the user when the default message does not adequately convey the meaning of the error. The message (either the default or user-specified) is sent when a field contains an invalid Value/Range. When this field is blank, the following default message is sent:</p> <p><i>The value 'x' is not valid for the field 'fieldname'</i></p> <p>You can concatenate literal and string variable data to create messages. For example,</p> <p><i>Fieldname "+nullsub(\$vrfield, "unknown")+ " is invalid...</i></p> <p>Note: Use the function <i>nullsub</i>, as in this example, when including variable data in a concatenated message. Otherwise, a variable whose value is NULL will render the entire message NULL.</p>

Field	Description
Prompt Panel	The name of the form used to display the valid Values/Ranges when an error is detected for a field. The default format is <i>validate.fields.array</i> . You can use the Prompt Panel to create special correction prompts on a field-by-field basis. The following variables are available for use on any special format you create:

Variable	Description
\$vrfield	Current value of the field. In other words, this is the incorrect value you are prompting the user to correct.
\$vrorig	Copy of the original incorrect value. This value remains constant throughout the correction process for a given field. This variable serves as a reminder to the user as to the field's original value.
\$field.case	Represents the case conversion that is in effect for a given field.
\$vfldsave	Name of the input field the operator is being prompted to correct.
\$valid.select	An array that contains a summary list of the valid values for a given field. The ctrl setting on the format definition for this input field <i>cannot</i> have the option for <i>no operator input</i> . In other words, it cannot have an option of <i>4</i> .
\$range.select	An array that contains a summary list of the valid ranges for a given field. The ctrl setting on the format definition for this input field <i>cannot</i> have the option for <i>no operator input</i> . In other words, it cannot have an option of <i>4</i> .

Field	Description
Values Summary	<p>A summary list of the Value Descriptions. The Validity Table Maintenance routine builds this list when Value definitions are added or updated. Manual modifications should not be applied to the list.</p> <p>Note: This is the same list provided to the user at run time when the field is in error. The items in this list should be meaningful to the users of the application.</p>
Range Summary	<p>A summary list of the range descriptions. This list is comprised of the value from the Min Desc field, the conjunction, and the Max Desc field. (These fields are defined in the Validity Range Detail Specification form.) The validity table maintenance routine builds this list when range definitions are added or updated. Manual modifications should not be applied to the items in this list.</p> <p>Note: This is the same list provided to the user when a field is in error. The items in this list should be meaningful to the users of the application.</p>

Secondary File Query Info tab

The screenshot shows the 'Secondary File Query Info' tab of a form. The fields are as follows:

- Condition:**
- Filename:** **Query Type:**
- Allow Many?:**
- Val Query:**
- Val Sorts:**
- Val Error Msg:**
- Select Query:**
- Select Sorts:**
- No Recs Opt:** **QBE Fmt:**
- Lookup Fields:** **Fields to Copy:**
- Val Fields:**

Figure 2-6: Secondary File Query Info tab in the Validity Table Specifications Form

Field	Description
Condition	A boolean field that controls when the Secondary File Query processing is performed. Processing is performed when the condition evaluates to <i>true</i> . It is not performed when the condition evaluates to <i>false</i> . The default is <i>false</i> . Refer to the section Refer to Run-Time Evaluation, page 134, for a complete description of when this condition is evaluated.
Filename	The file name against which the Secondary Query is performed. It is a optional field. If the field is NULL at validation time, the query is not performed. It must be a valid Database Dictionary name.
Query Type	<p>An optional field that defines which type of query is executed. The valid choices are <i>v</i> (validate) or <i>d</i> (duplicate check). The default is <i>v</i>.</p> <p>A Query Type of <i>v</i> causes the validity routine to execute the Val Query against a specified Filename. If no records are found, it is treated as an error condition and the operator is prompted to select a valid record from the QBE list that is produced by executing the Select Query against the Filename.</p> <p>A Query Type of <i>d</i> causes the validity routine to execute the Val Query against a specified Filename. If a record is found, it is treated as a potential error condition. The validity routine issues the Val Error Msg to the operator; however, processing continues to the next field. The operator is not prompted to correct data. This is similar to the UNIX <i>ping</i> function.</p>
Allow Many?	A logical field that controls the action taken when many (more than one) record is found by executing the Val Query . If more than one record is found and this field evaluates to <i>false</i> , then the user must select one of the records from the list or execute the Lkup Query . If a record is selected from the list, the fields in the lookup record are copied to the record being validated per the definitions in the <i>Fields to Copy</i> section. If more than one record is found and this field evaluates to <i>true</i> , then the field is considered valid; however, no fields are copied to the record being validated.

Field	Description
Val Query	<p>The query executed against the Filename file. When it is executed, one of three conditions will exist:</p> <ul style="list-style-type: none"> ■ One record is found: The data in the field being validated is considered valid and processing continues to the next field ■ No records are found: The data in the field being validated is considered invalid, and the Select Query is executed ■ Many records are found: Refer to the definition for the Allow Many? field.
Val Sorts	<p>An optional field that defines the key used when validity tables execute the Val Query. If this key is not valid at run time, the validity routine assumes that all values are true for the field being validated.</p>
Val Error Msg	<p>When a field is in error, a message is sent to the operator describing the error condition. When no records are found executing the Val Query, the format of the default message is:</p> <p style="text-align: center;"><i>No records found executing the secondary query</i></p> <p>When many records are found executing the Val Query, the format of the default message is (Note: this message is sent only when the Allow Many? option is <i>true</i>):</p> <p style="text-align: center;"><i>Many records found executing the secondary query</i></p> <p>The Val Error Msg field overrides the default message with a customized message. It allows for more meaningful error messages to be sent to the user when the default message does not convey the meaning of the error.</p> <p>You can concatenate literal and string variable data to create messages. For example,</p> <p>"Fieldname "+nullsub(\$vrfield, "unknown")+ " is invalid..."</p> <p>Use the nullsub function, as in this example, when including variable data in a concatenated message. Otherwise, a variable whose value is NULL will render the entire message NULL.</p>
Select Query	<p>The query executed if no records are found by executing the Val Query. The records selected from this query are all of the possible valid records from which the user can select. If no records are selected using this query the No Recs Opt is evaluated to determine how processing should continue.</p>
Select Sorts	<p>An optional field that defines the key used when validity tables execute the Select Query. If this key is not valid at run time, the validity routine assumes that all values are true for the field being validated.</p>

Field	Description
No Recs Opt	An optional field that controls if processing should continue to the next field or return to the data record when no records are selected using the Select Query . The value Proceed causes processing to continue to the next validity field. The value Return causes processing to return to the data record. The default is Proceed .
QBE Format	The name of the QBE form that displays the records selected by using the Val Query and Select Query . This is an optional field. If it is NULL, the format used is <i>filename.qbe</i> . As a standard, it is recommended that the first field in the QBE format be the field that is copied from the Secondary File to the record being validated. This will help your users to understand which field is being copied. The other fields in the QBE list should help define the value of the field being copied.
Lookup Fields	The names of the input fields copied from the record selected using the Val Query or the Select Query to the record being validated. This is a required field if a Filename is defined. Each entry is validated against the Filename Database Dictionary. If any element is not a valid field name, a window is opened from which you must select a valid field name. The data type of each element must be the same as the data type defined at the Field Level in this Validity record. There is a one-to-one relationship between this array and the Val Fields array.
Val Fields	The names of the input fields in the record being validated that are updated with data from the record selected using the Val Query or the Select Query . Each entry is validated against the first Database Dictionary defined in the Files/Formats array. If any element is not a valid field name, a window is opened from which you must select a valid field name. The data type of each element must be the same as the data type defined at the field level in this validity record. There is a one-to-one relationship between this array and the Lookup Fields array.

Alternate Application tab

Figure 2-7: Alternate Application tab in the Validity Table Specifications Form

Field	Description
Alt Application	The name of an alternate RAD application to execute in place of the standard validity table processing. It is an optional field. This alternate application must perform all of the processing necessary to validate the data field. You are responsible for ensuring that the variables used in the alternate application do not corrupt variables used by Validity processing. If variables are corrupted, the results of validity processing are unpredictable. The <i>normal</i> exit from this application causes Validity processing to continue to the next field in the list. The <i>error</i> exit from this application causes Validity processing to return to the data record being validated.
Names	The names of the input fields defined on the parameter panel of the alternate application to which data from the validity routine is passed. There is a one to one relationship between the elements in this array and the Values array.
Values	The values passed from the validity routine to the alternate application. There is a one to one relationship between the elements in this array and the Names array.

Value Summary Detail

You can display the details of a single entry in the Values Summary array, or view the details for all the entries displayed in a scrollable form.

To display the details of a single entry in the Value Summary array

- ▶ Place the cursor on a line in the Value Summary array and select **Options > Edit Line**.

The details for that entry are displayed.

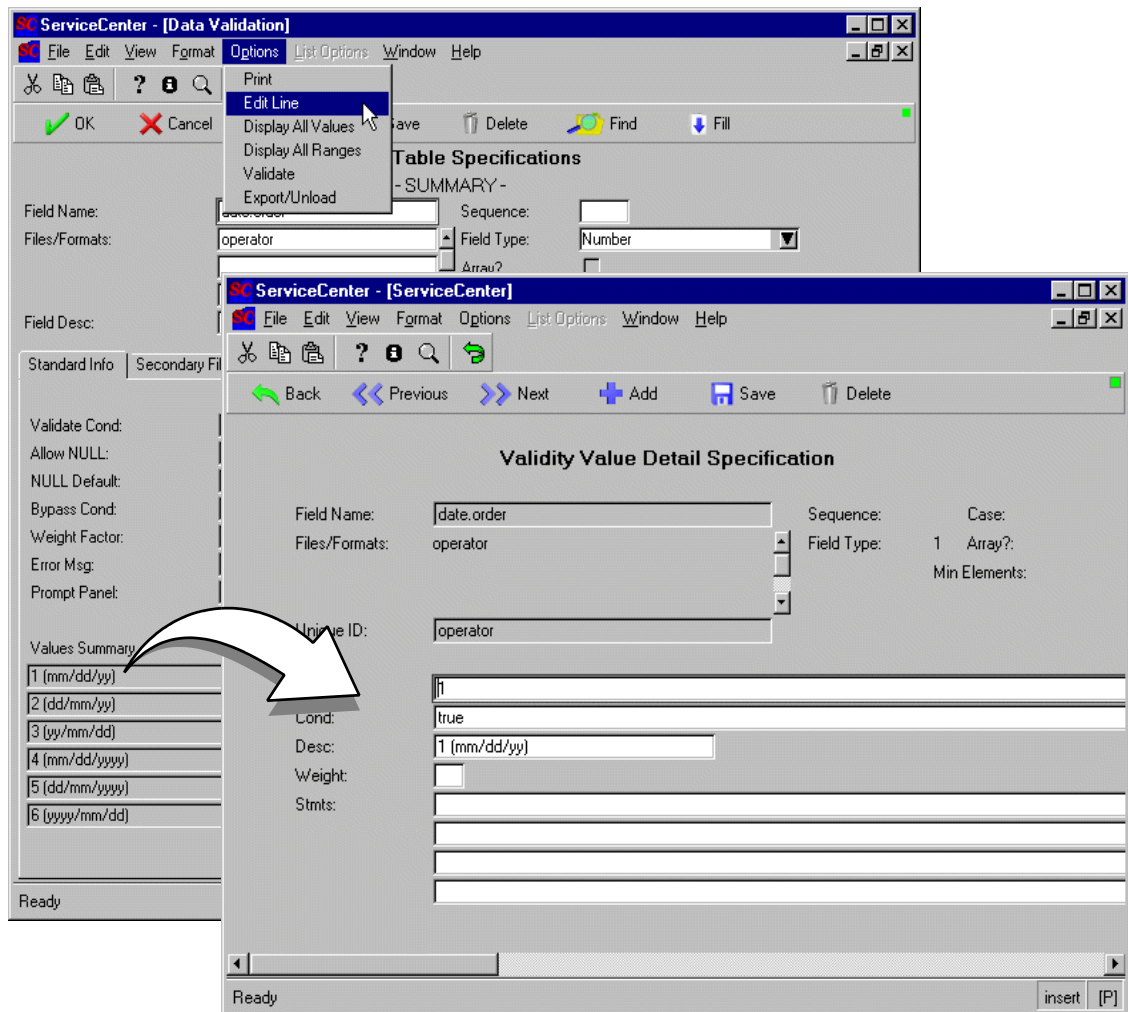


Figure 2-8: Values Summary details—single entry

To display the details of all entries in the Value Summary array

- ▶ Select Options > Display All Values in a validity record.

Details of all the entry in the Value Summary array are displayed in a scrollable list.

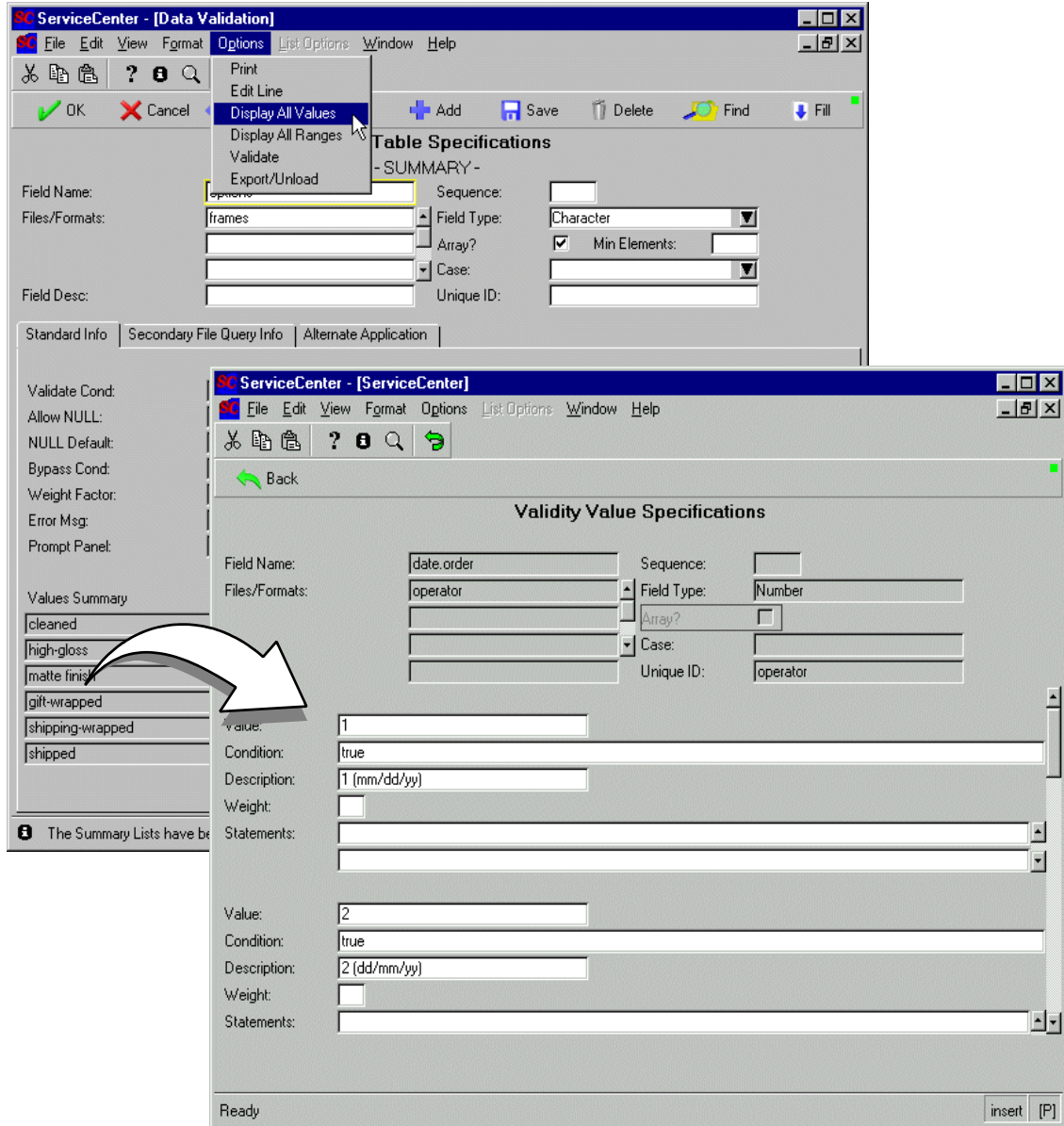
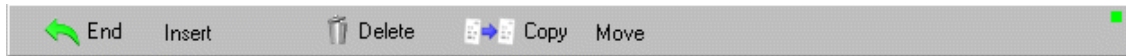


Figure 2-9: Values Summary details—all entries

Note: You may edit or sort the contents of this record as you would any table in ServiceCenter. Select **Options > Edit Table** to display the typical table editing controls in the tool tray.



Field Definitions

Field	Description
Value	<p>A valid literal for this field. Any number of literals can be defined for each field. Non-string values are entered as string data and, at run time, converted to the appropriate data type based on the specified Field Type.</p> <p>Note: Under most circumstances, Values are used to validate data types 1 (number), 2 (character), and 4 (logical). Normally, data types of 3 (date/time) are not defined as a Value.</p>
Cond	<p>Defines the condition under which the specified Value is allowed. If the condition evaluates to <i>true</i>, then the specified value is allowed. If the condition evaluates to <i>false</i>, the specified value is not allowed. Blanks are treated the same as an evaluation of <i>false</i>. For example, the following condition ensures that character.field starts with s:</p> <pre>character.field in \$val#"s"</pre> <p>Note: If all Value Conditions and Range Conditions evaluate to <i>false</i>, then all values are considered valid for this field.</p>
Desc	<p>The description of the Value. It allows for clarification of the Value. For instance, the Field reason has three valid values (mt, nw, fx). Some users may not understand the meaning of these values. By making the description maintenance, new, and fix respectively, the selection process is easier and more meaningful for the user. The Value Summary list reflects the value of each Desc field. If Desc is left blank, the value in the Value Summary list is the actual Value.</p> <p>Note: The Validation Routine validates against the Field Values of the definition, not the Value Desc's.</p>

Field	Description
Weight	The weight this field has when used for risk calculation in Change Management. This is an optional field. It is used in conjunction with the Risk Level field in Change Management. Refer to <i>Change Management</i> in the <i>Application Administration Guide</i> for more information.
Stmts	<p>The processing statement(s) executed when a field contains the value of this definition. It can be thought of as <i>if, then</i> processing. <i>IF</i> the field contains the specified value, <i>THEN</i> execute these statements. This is an optional field.</p> <p>Note: If the Value is selected during look-up processing, the Stmts are executed.</p> <p>The Stmts field is similar to a RAD process Panel. All normal ServiceCenter functions can be used. These expressions can manipulate data in any of the variables or fields that are currently available. For more information on how to structure expressions, refer to <i>Format Control (Calculations)</i> and <i>System Language</i> in <i>System Tailoring</i>.</p>

Range Summary Detail

You can display the details of a single entry in the Range Summary array, or view the details for all the entries displayed in a scrollable form.

To display the details of a single entry in the Range Summary array

- Place the cursor on a line in the Range Summary array and select **Options > Edit Line**.

The details for that entry are displayed.

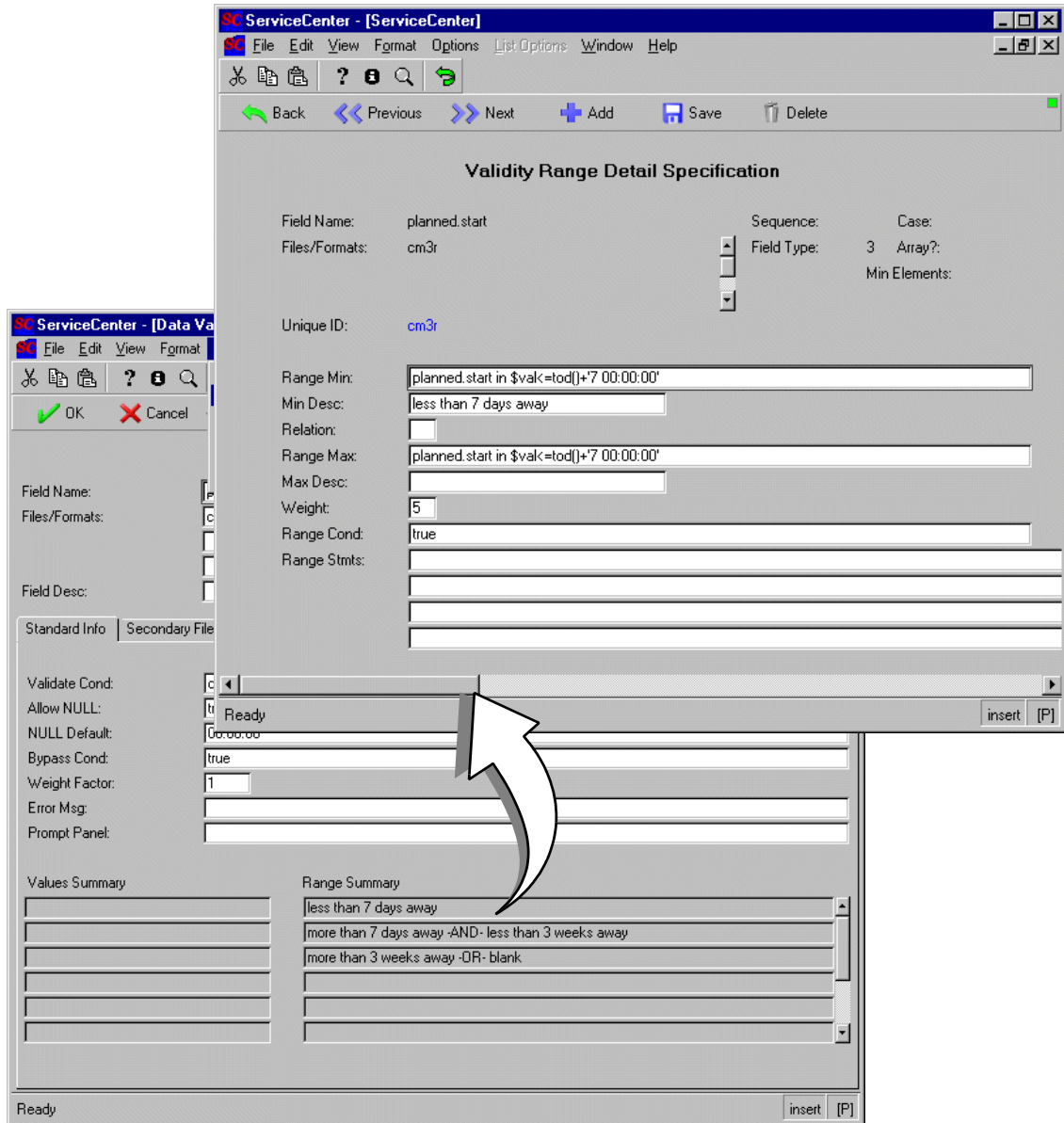


Figure 2-10: Range Summary details—single entry

To display all the values in the Range Summary array

- ▶ Select Options > Display All Ranges in a validity record.

Details of each entry in the Range Summary array are displayed in a scrollable list.

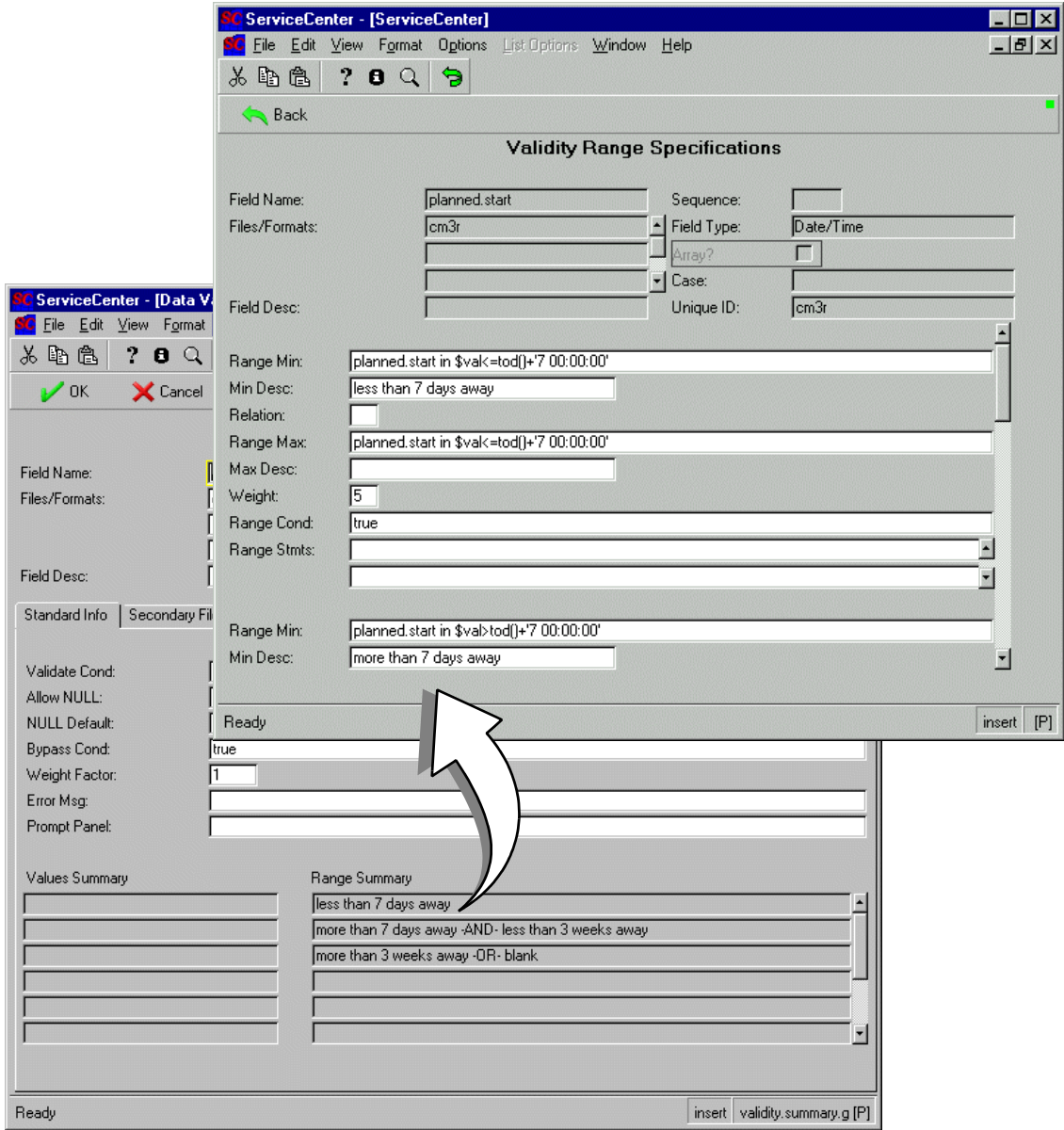
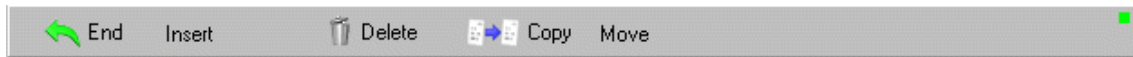


Figure 2-11: Range details for the Validity Table Specification form

Note: You may edit the contents of this record as you would any table in ServiceCenter by selecting **Options > Edit Table**. The tool tray now displays typical table editing controls.



Field Definitions

Field	Description
Range Min	<p>The processing statement executed at run time to determine the minimum value of the range. Normally, the syntax of the statement is:</p> <pre>\$vrfield ? <value></pre> <p>Where \$vrfield contains the value of Field Name in the record being validated.</p> <p>? is the comparison operator (e.g. =, >, <, ~)</p> <p>value is the actual minimum value.</p> <p>Note: Both a Range Min and Range Max must be defined to constitute a valid range.</p> <p>To validate that a field is not NULL, specify the Range Max as <i>true</i> and the Range Min as:</p> <pre>not null(field in \$val)</pre> <p>For example, if the field were NULL, then the Range Min condition would evaluate to <i>false</i>. A condition of <i>true</i> and <i>false</i> results in <i>false</i>, therefore NULL would not be allowed. However, if the field were NOT NULL then the Range Min condition would evaluate to <i>true</i>. A condition of <i>true</i> and <i>true</i> results in <i>true</i>.</p>
Min Desc	<p>The description of the Range Min. The Range Summary reflects the value of the Min Desc and the Max Desc.</p>

Field	Description
Relation	<p>Allowable values are and, or. The relation must be spelled out. The substitute characters for and or or (i.e. &,) cannot be used. The Relation defines the conjunction of the Range Min and Range Max processing statements.</p> <p>The conjunction of the Range Min and Range Max definitions by the relational operator or is:</p> <p style="padding-left: 40px;">range minimum OR range maximum</p> <p>For example:</p> <p style="padding-left: 40px;">\$vrfield>30 OR \$vrfield<15</p> <p>Normally, Range Definitions have one minimum and one maximum value, as shown in the example above. However, it is possible to specify more than one minimum and maximum value. For example:</p> <p style="padding-left: 40px;">\$vrfield>=5 and \$vrfield<=10 (Range Minimum)</p> <p style="padding-left: 40px;">OR (Range Conjunction)</p> <p style="padding-left: 40px;">\$vrfield=20 and \$vrfield<=25 (Range Maximum)</p> <p>This definition allows the ranges 5 through 10 and 20 through 25 to be valid for a particular field.</p>
Range Max	<p>The processing statement executed at run time to calculate the maximum value of the range. Normally, the syntax of the statement is:</p> <p style="padding-left: 40px;">\$vrfield ? <value></p> <p>Where \$vrfield contains the value of Field Name in the record being validated.</p> <p>? is the comparison operator (e.g. =, >, <, ~)</p> <p>value is the actual maximum value.</p> <p>Note: Both a Range Min and Range Max must be defined to constitute a valid range.</p>
Max Desc	<p>The description of the Range Max. The Range Summary reflects the value of the Min Desc and the Max Desc.</p>
Weight	<p>The <i>weight</i> this range carries for risk calculation in Change Management. This is an optional field used in conjunction with the Risk Level field in Change Management. Refer to <i>Change Management in the Application Administration Guide</i> for more information.</p>

Field	Description
Range Cond	<p>Defines the condition under which the specified range is allowed. If the condition evaluates to <i>true</i>, the specified range is allowed. If the condition evaluates to <i>false</i>, the specified range is not allowed. Blanks are treated the same as an evaluation of <i>false</i>.</p> <p>Note: If all Value Conditions and Range Conditions evaluate to <i>false</i>, then all values are considered valid for this field.</p>
Range Stmt	<p>The processing statement(s) executed when a field's value is within the range of this definition. It can be thought of as <i>if, then</i> processing. <i>IF</i> the field contains the specified value, <i>THEN</i> execute these statements. This is an optional field.</p> <p>Note: Range Stmt are executed during <i>look-up</i> processing. The Range Stmt field is similar to a RAD process Panel. All normal ServiceCenter functions can be used. These expressions can manipulate data in any of the variables or fields that are currently available. For more information on how to structure expressions, refer to <i>Format Control</i> (Calculations) and <i>System Language</i> in <i>System Tailoring</i>.</p>

Creating Validity Table Definitions

To create a new validity table definition, complete the appropriate records in the following order:

- Task 1: Add field level definitions
- Task 2: Add value definitions
- Task 3: Add range definitions
- Task 4: Validate the validity definitions

Task 1: Add Field Level Definitions

To add field level definitions:

- 1 Access the validity table specifications form using one of the procedures discussed in *Accessing the validity File* on page 50.
- 2 Enter the name of the field for which you want to create a validation record in the **Field Name** field.
- 3 Enter the name of the file or files in which this field is found in the **Files/Formats** field.

- 4 Click Search if you are unsure if a record already exists for this field.
- 5 If no record exists, click New to add your new record.

The following message appears in the status bar: *Data Validation record added.*

The screenshot shows the 'ServiceCenter - [Data Validation]' window. The title bar includes 'File Edit View Format Options List Options Window Help'. Below the title bar is a toolbar with icons for OK, Cancel, Add, Save, Delete, Find, and Fill. The main area is titled 'Validity Table Specifications - SUMMARY -'. It contains several input fields and dropdown menus:

- Field Name: company
- Files/Formats: contacts
- Field Type: Character
- Field Desc: (empty)
- Sequence: (empty)
- Array?:
- Case: (empty)
- Unique ID: (empty)

Below these fields are three tabs: 'Standard Info', 'Secondary File Query Info', and 'Alternate Application'. The 'Standard Info' tab is active and contains several more input fields:

- Validate Cond: (empty)
- Allow NULL: (empty)
- NULL Default: (empty)
- Bypass Cond: (empty)
- Weight Factor: (empty)
- Error Msg: (empty)
- Prompt Panel: (empty)

At the bottom of the form are two sections: 'Values Summary' and 'Range Summary', each with a list of empty input fields. The status bar at the very bottom of the window displays the message 'Data Validation record added.' and an 'insert [P]' button.

Figure 2-12: New validity record

- 6 Complete the remaining fields in the form (including the other tabs) with the exception of the **Values Summary** and **Range Summary** arrays.
- 7 Select **Options > Validate** to validate the information you have entered in the form.

If you have entered an invalid value in the form, a message appears in the status bar specifying the error. For example, if you misspell the name of the file, the following message appears: *The name <file name> is not a valid dbdict or format. Select a dbdict.* A QBE list of file names is displayed, allowing you to select the correct file. If you switch back to the form without selecting a valid file name, the following message appears in the status bar: *The Validity record is NOT valid.*

- 8 Click Save when you have successfully validated your record.

The following message appears in the status bar: *Data Validation record updated.*

ServiceCenter - [Data Validation]

File Edit View Format Options List Options Window Help

OK Cancel Add Save Delete Find Fill

Validity Table Specifications
- SUMMARY -

Field Name: company Sequence:
Files/Formats: contacts Field Type: Character
Field Desc: Array?
Case: Uppercase
Unique ID:

Standard Info | Secondary File Query Info | Alternate Application

Validate Cond: true
Allow NULL: false
NULL Default:
Bypass Cond: false
Weight Factor:
Error Msg: The name in this field must be a valid Company Code from the company file.
Prompt Panel:

Values Summary Range Summary

Data Validation record updated. insert [P]

Figure 2-13: Updated validity record

Task 2: Add Value Definitions

To add a value definition:

- 1 Put the cursor in the first line of the **Value Summary** array.
- 2 Select **Options > Edit Line**.

The Value Detail Specification form is displayed (Figure 2-11 on page 72). For field definitions, refer to page 73.

- 3 Complete the record as necessary and click Add.

The record is added and the buttons in the tool tray change. The following message appears in the status bar: *Range/Value successfully added.*

The screenshot shows a window titled "ServiceCenter - [ServiceCenter]" with a menu bar (File, Edit, View, Format, Options, List Options, Window, Help) and a toolbar with icons for Back, Previous, Next, Add, Save, and Delete. The main area is titled "Validity Value Detail Specification" and contains the following fields:

- Field Name:
- Files/Formats:
- Unique ID:
- Value:
- Cond:
- Desc:
- Weight:
- Stmts:

Additional fields on the right include Sequence, Case (uc), Field Type (2), Array?, and Min Elements. The status bar at the bottom shows "Range/Value successfully added." and an "insert [P]" button.

Figure 2-14: New value detail specification record

- 4 Click Back to return to the validity record.

The value you entered in the Desc field appears in the Values Summary array.

- Put the cursor in the next blank line of the Values Summary array and repeat steps 2-4 until you have added all the values for your field.

Figure 2-15: New validity record with Values Summary elements added

- Click Save to add the record to the validity file.

Task 2: Add Range Definitions

To add a range definition

- Put the cursor in the first line of the Range Summary array.
- Select **Options > Edit Line**.

The Range Detail Specification form is displayed (Figure 2-8 on page 67). For field definitions, refer to page 69.

- Complete the record as necessary and click Add.

For additional information on writing conditions and expressions, refer to *System Language in System Tailoring, Volume 2*.

Note: It is possible that the same range could be allowed under multiple conditions within the same validity record; therefore, the system does not check for duplicates field values.

Important: If your character field starts with a < or > symbol, you must insert a blank space before it. These symbol are dropped by the ServiceCenter parser unless preceded by a blank space.

The record is added and the buttons in the tool tray change. The following message appears in the status bar: *Range/Value successfully added*.

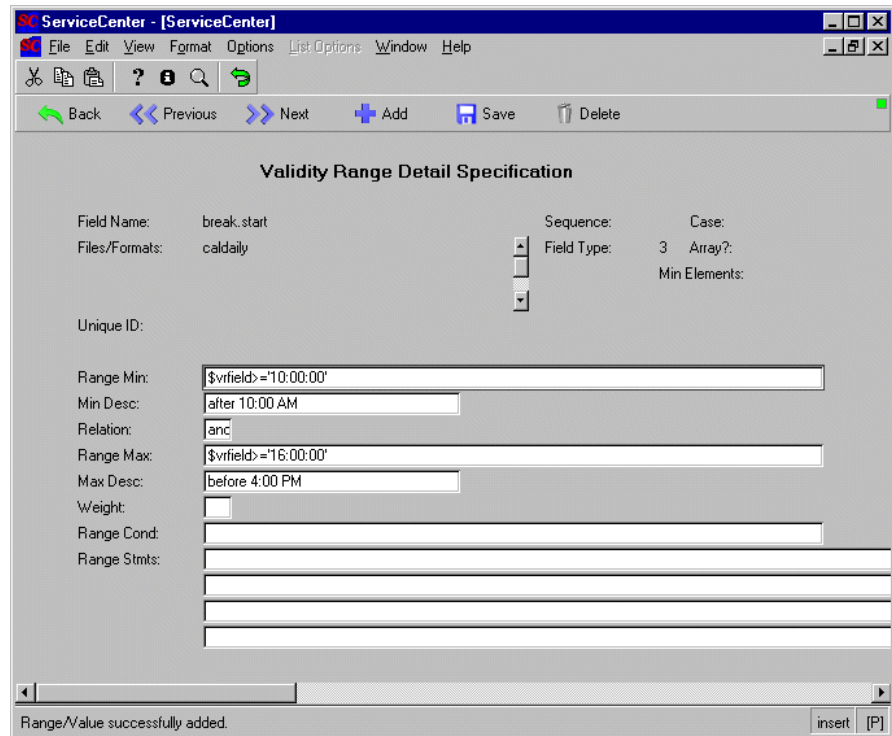


Figure 2-16: New range detail specification record

- 4 Click Back to return to the validity record.

The system combines the values you entered in the Min Desc and Max Desc fields with the relational operator you specified to produce the string that appears in the Range Summary array.

ServiceCenter - [Data Validation]

File Edit View Format Options List Options Window Help

OK Cancel Add Save Delete Find Fill

Validity Table Specifications
- SUMMARY -

Field Name: Sequence:

Files/Formats: Field Type:

Array?

Field Desc: Case:

Unique ID:

Standard Info | Secondary File Query Info | Alternate Application

Validate Cond:

Allow NULL:

NULL Default:

Bypass Cond:

Weight Factor:

Error Msg:

Prompt Panel:

Values Summary

Range Summary

Summary Lists successfully rebuilt. insert [P]

Figure 2-17: New validity record with Range Summary elements added

- 5 Put the cursor in the next blank line of the Range Summary array and repeat steps 2-4 until you have established all the ranges for your field.
- 6 Click Save to add the record to the validity file.

Task 4: Validate the Validity Definitions

The validation routine ensures that field, file, form, and application names referenced in the definition are valid. It also prompts for a data type if you have left the **Field Type** field blank.

Important: Peregrine recommends that you validate each record before using it to validate data records. However, the only requirement is that you define a field *type* for each validity record before using it to validate data records.

To validate your validity record

- ▶ Select **Options > Validate**.

If the validation routine detects an invalid entry, the appropriate message appears in the status bar and a form is displayed from which to select the correct entry.

Validity Validation Rules

- Each entry must have at least one entry in the **Files/Formats** field.
- Each entry in the **Files/Formats** field must be a valid Database Dictionary name or form name.
- The **Field Name** must be defined in each Database Dictionary listed in the **Files/Formats** array or to the Database Dictionary associated with the form.
- The data type of the **Field Name** must be the same for each file listed in the **Files/Formats** array or for the file associated with the form.
- The **Unique ID** is a user-provided value that uniquely identifies the record.
- The unique key of the *validity* table contains the following fields: *field.name*, *sequence*, and *filename*. Each combination of the **Field Name**, **Sequence** and each element of the **Files/Formats** array in the validity record must also be unique within the validity table. This check is performed before the record is added or updated. If duplicates are found, the add/update action is not performed, and the following message is displayed: *This record contains an invalid duplicate key*.
- The value in the **Alt Application** field must be a valid RAD application.
- The **Secondary File Query Filename** value must be a valid Database Dictionary name.

- The **Lookup Fields** must be defined to the **Secondary Query Filename** and their data types must be the same as the **Field Name**.
- The **Val Fields** must be defined to the **Files/Format** file and their data types must be the same as the **Field Name**.
- The **QBE Format** must be a valid form name.
- The value in the **No Recs Option** field must be the word **Proceed**, or **Return**. If neither is selected the system defaults to **Proceed** when the record is validated.

Invalid File or Form Name

If any element of the **Files/Formats** field is not a valid Database Dictionary name or a valid form name, the following message appears in the status bar: *The name <name> is not a valid dbdict or format. Select a dbdict.* A QBE list of valid file names is displayed. Double-click on the correct file to replace the invalid entry.

If you have entered an invalid form name in the **Files/Formats** field and want to view a QBE list of valid forms, select **Options > Display All Formats**. A QBE list of valid forms is displayed. Double-click on the correct form to replace the invalid entry. If the form you select is not associated with a file, you must respond indicating the file with which the form will be associated at run time.

Note: The file associated with the form must contain the **Field Name**. If it does not, and the first element in the list is being validated, you are prompted to select a valid field name. If the second or greater element is being validated, you are returned to the original record with an error message.

To switch back to the list of files, select **Options > Display Table List**.

Invalid Field Name

If you enter an invalid field name, the following message appears in the status bar: *The field name <field name> is not valid. Select one from the list.* A list of valid field names for the file you have entered is displayed. To select a valid field, click on a field in the list and press Enter. The invalid entry is replaced by the field you have selected.

Invalid Application Name

If you have entered an invalid application name in the **Alt Application** field in the **Alternate Application** tab, a QBE list of valid applications is displayed and the following message appears in the status bar: *The Application Name is not valid. Select one from the list.* Double-click on the correct application to replace the invalid entry.



Deleting Validity Record Components

You can delete entire validity records or parts of records using several different methods.

- Delete the entire validity record and all its components
- Delete a value or range definition from an individual record
- Delete value and range definitions from a table format

To delete an entire validity record

- 1 Open a validity record using one of the methods discussed in *Accessing the validity File* on page 50.

- 2 Click Delete.

The following prompt is displayed: *Are you sure you want to delete this record?*

- 3 Click Yes.

To delete a single value or range definition

- 1 Open a validity record using one of the methods discussed in *Accessing the validity File* on page 50.
- 2 Place the cursor in the line containing the value or range definition you want to delete
- 3 Select **Options > Edit Line**.
- 4 Click Delete in the detail form.
- 5 The following prompt is displayed: *Are you sure you want to delete this validity detail?*
- 6 Click Yes.
- 7 You are returned to the validity record.
- 8 Click Save.

To delete value or range definitions from a table

- 1 Open a validity record using one of the methods discussed in *Accessing the validity File* on page 50.
- 2 Select **Options > Display All Values** or **Display All Ranges** to display a table of value or range definitions.
- 3 Select **Options > Edit Table** to access the table editing controls.
- 4 Put the cursor anywhere in the record you want to delete and click Delete.

Warning: You are not prompted to confirm the command. The record is removed from the table when Delete is clicked.

- 5 Repeat the process with any other value or range definitions you want to delete.
- 6 Click End to exit the table edit mode.
- 7 Click Back to return to the validity record.
- 8 Click Save.

Printing Validity Definitions

A detailed report of a validity record has four main sections that display all the information contained in the record and in the detailed summary forms:

- Field specifications
- Query specifications
- Values specifications
- Range specifications.

Note: Any scalar field or array element that is greater than 60 characters in length is printed in its entirety with the use of continuation lines. All elements of all arrays are also printed. All continuation lines start with “***”.

To print a detailed report of an entire validity record

- 1 Access the validity record you want to print using one of the methods described in *Accessing the validity File* on page 50.

Note: To generate a detailed report, the validity record must contain Values or Range definitions.

- 2 Place the cursor in a Summary field containing a value.
- 3 Select **Options > Edit Line**.

A detail specification record is displayed for that summary item.

- 4 Select **Options > Print Detail Entry**.

The following message is displayed in the status bar: *Print of Validity Table successfully scheduled*.

The job is added to the schedule file, which will print the record currently being viewed. The report is run by the report background scheduler and is printed on the printer selected in **File > Printing Options** (Client or Server).

The report shown in Figure 2-18 on page 87 is a sample of a detailed validity report.

ServiceCenter - [spool]

File Edit View Format Options List Options Window Help

OK Cancel Previous Next Add Save Delete

Name: Validity Table Detail Number: 841 Page: 1
 Date: 08/28/01 07:15 Printer: sysprint Operator: falcon

08/28/01 07:15 ServiceCenter

Validity Table Detail

Page: 1

Operator: falcon Selection: field.name="priority" and files="cm3t"

Sequence:

Field Name: priority

Field Specifications

Files/Formats: cm3t

Sequence:

Case:

Field Type: 2

Array?

Min Elements:

Validate Cond: false

Allow Nulls: false

Null Default: 2

Bypass Condition: false

Weight Factor:

Error Message:

Prompt Format:

Query Specifications

Condition:

Filename:

Query Type:

Allow Many?

Val Query:

Val Sorts:

Val Error Msg:

Select Query:

Select Sorts:

No Recs Opt: proceed

QBE Format:

Lookup Fields:

Val Fields:

Alt Appl:

Names:

Ready insert [P]

Figure 2-18: Validity table detail report

Invoking Validity Table Processing

You can invoke validity table processing by calling the RAD application `validate.fields` in the Subroutines process of Format Control. Validity table processing is invoked to achieve two goals:

- Validate the fields in a record during record processing
- Allow *look-up* processing during record editing (similar to ServiceCenter Find and Fill capability)

Validating Fields During Record Processing

During record processing, the fields in a record can be validated one at a time or in a particular order.

Note: To invoke validity table processing in Change Management, refer to *Change Management* in the *Application Administration Guide*.

Important: You must validate each validity table definition before using it to validate data records. Refer to *Task 4: Validate the Validity Definitions* on page 82, for instructions.

To invoke field validation in a record during record processing

- 1 Access the Format Control record for the form for which you want to invoke validity table processing. If a record does not already exist, you must create one.
Refer to *Format Control* in *System Tailoring* for details on accessing and using Format Control.
- 2 Display the Subroutines process.

Note: The view in Figure 2-19 on page 89 is of the expanded form.

ServiceCenter - [Format Control frames]

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: frames View:

Subroutines

Application Name:

Comments:

Names	Values
second.file	\$file
<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>

Error Message:

Add:

Update:

Delete:

Before:

Display:

Initial:

Ready insert formatctrl.maint.subs.v.g [P]

Figure 2-19: Subroutines process of a Format Control record

3 Enter `validate.fields` in the Application Name field.

Note: The logical values in the Add, Update, and Before fields are examples only. Refer to the Format Control documentation for instructions on how to decide when Format Control is processed.

- 4 Enter the appropriate parameters for `validate.fields` in the **Name** and **Value** fields using the information from the following table.

Name	Value	Description
second.file	\$file	Executes all validity table records (all fields) defined for the file being processed
name	<field name>	Executes only one validity table definition (one field) defined for the file being processed Note: You can repeat this type of definition in the Format Control Subroutines section as many times as necessary. However, it is more efficient to call the subroutine once and pass a list of the fields to be validated.
names	<list of fields> or \$<variable>	Executes a specific list of validity table definitions defined for the file associated with the form For example, define a \$ variable in either the Initializations or Calculations process of the Format Control record as: \$fieldlist={"reason", "risk", "planned.start"} In the Subroutine call, specify the Value input as: Name: names Value: \$fieldlist Note: You can use any variable name you choose, however, you are responsible for ensuring that your variable name does not corrupt a variable used by the application <i>validate.fields</i> .
text	current.format() or <form name>	Executes a specific list of validity table definitions defined for the form just displayed to the user

- 5 Enter a message in the **Error Message** field.

The recommended error message is: *The original record is displayed*. If a message is not defined, an unfriendly error message is issued when returning to the data record.

- 6 Click Save.

Parameter Panel Field Definitions—validate.fields

ServiceCenter - [Forms Designer]

File Edit View Format Options List Options Window Help

End Continue

application label

comments

Name of Field to Process: name

Array of Fields to Process: names

Calculate Risk Only: boolean1

Store Calc Results: prompt

Risk Maximum: number1

File Variable to Process: second.file

Online Lookup: cond.input

Suppress Not Found Msg: string1

Format Name: text

Error Exit should return to the previous RIO panel, not \$error

normal error

Ready insert validate.fields [P]

Figure 2-20: validate.fields parameter panel field names

Label	Field	Description
Name of Field to Process	name	If one field is to be validated or if the <i>look-up</i> function is being invoked, specify that fieldname here. This is an optional field. Note: When calling <i>validate.fields</i> from a RAD application to invoke the look-up function, use the <i>cursor.field.name()</i> RAD function. By doing this, one function key can be used to look-up any of the fields on the form.
Array of Fields to Process	names	The optional ARRAY containing the list of fields to validate. Note: If both Name of Field to Process and Array of Fields to Process fields are NULL, all fields defined for the specified file are validated.
Calculate Risk Only	boolean1	An optional boolean flag that initiates risk calculation processing for Change Management. The default is <i>false</i> .
Store Calc Results	prompt	The name of the field that will contain the results of the risk calculation. This is an optional field.
Risk Maximum)	number1	The maximum value that can be calculated for risk. This is an optional field.
File Variable to Process	second.file	The file variable that contains the current version of the record to be validated. This is a required field.
Online Lookup	cond.input	An optional boolean flag that, when <i>true</i> , causes the validity routine to present a list of all valid values/ranges for a specified field rather than attempt to validate the value in the current record. The default is <i>false</i> .
Suppress Not Found Msg	string1	An optional boolean flag that, when <i>true</i> , causes the validity routine to suppress a <i>not found</i> message when no validity definitions are found for a particular field. The default is <i>false</i> .
Format Name	text	An optional field that defines the name of the format being validated. Normal <i>validate.fields</i> processing retrieves all Validity Table records for the File being validated. However, there may be circumstances where you need to validate only certain fields that are defined for a particular form. In those circumstances, pass the name of the form through this parameter. In doing so, only those validity records whose Files/Formats field contains the name of the form are selected for validation.

Label	Field	Description
Normal	normal	The name of the panel to exit to if all fields are valid. When <i>online lookup</i> is <i>true</i> , the normal exit is usually back to the <i>rio</i> panel from where lookup was invoked.
Error	error	In ALL cases, should exit to the previous <i>rio</i> panel. It should not exit to \$error.

Look-up Processing

Calling the validity routine while a record is being edited simulates the Find and Fill functions. This gives the user the ability to determine the valid values or ranges of a field before attempting to process a record. The validity look-up routine can be executed from three places:

- Format Control Additional Options process
- Format Control Subroutines process
- Displayoptions

Format Control Additional Options

You can use Format Control to set up a menu option called Validity Lookup. When selected from the Options menu of a form being edited, this option performs a validity check on the field of focus in the form for which validity specifications have been defined. Invalid fields are highlighted and a message is displayed in the status bar giving details on the type of error incurred.

To create a Validity Lookup option in Format Control

- 1 Access the Format Control record for the form for which you want to invoke validity table processing. If a record does not already exist, you must create one.

Refer to *Format Control* in *System Tailoring* for details on accessing and using Format Control.

- 2 Display the Additional Options process.

Note: The view in Figure 2-21 on page 94 is of the expanded form.

ServiceCenter - [Format Control frames]

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: frames Use as Master View:

Form to Display: frames Allow Edit

Allow Input

Additional Options

Option	Condition for Option	values
1	true	cursor.field.name()
Command	Validity Lookup	\$file
Application	validate.fields	val("true", 4)
Error Message	The original record is displayed.	
Reset on Return	true	
Names	name	cursor.field.name()
	second.file	\$file
	cond.input	val("true", 4)
Option	Condition for Option	
Command	Description	
Application	Comments	
Error Message		
Reset on Return		

Ready insert formatctrl.maint.options.v.g [F]

Figure 2-21: Additional Options process of a Format Control record

3 Enter the following field values:

Field	Value
Condition for Option	true
Command	Validity Lookup
Application	validate.fields

Field	Value
Error Message	The original record is displayed. Note: This is the recommended message. If this field is left blank, you will receive an unfriendly error message when you return to the data record.
Reset on Return	true

- 4 Enter the appropriate parameters for `validate.fields` in the **Name** and **Value** fields using the information from the following table:

Names	Values
name	<code>cursor.field.name()</code>
second.file	<code>\$file</code>
cond.input	<code>val("true", 4)</code>

Note: The data passed to the boolean parameter must be a type 4 (hence the VAL() statement for the Value). As an alternative, a \$variable could be defined in the *initialization expressions* (e.g. `$flag=true`) and then passed to the subroutine.

- 5 Click Add to add a new Format Control record, or click Save to save your updates.

Format Control Subroutines

You can invoke validity lookup processing by calling `validate.fields` from the Subroutines process of Format Control. A validity check is performed automatically when you attempt to add or save a form containing an invalid field value. Invalid fields are highlighted and a message is displayed in the status bar giving details on the type of error incurred.

To call `validate.fields` from Format Control

- 1 Access the Format Control record for the form for which you want to invoke validity table processing. If a record does not already exist, you must create one.

Refer to *Format Control* in *System Tailoring* for details on accessing and using Format Control.

2 Display the Subroutines process.

Note: The view in Figure 2-22 on page 96 is of the expanded form.

Format Control Maintenance - Subroutines

Name: frames View:

Subroutines

Application Name: validate.fields

Comments:

Names	Values
name	cursor.field.name()
boolean1	false
second.file	\$current.file.variable
cond.input	true

Error Message: The original record is displayed.

Add: true

Update: true

Delete:

Before: true

Display:

Initial:

Format Control record updated. insert formatctrl.maint.subs.v.g [P]

Figure 2-22: Subroutines process of a Format Control record

3 Enter the appropriate parameters for `validate.fields` in the Name and Value fields using the information from the following table:

Names	Values
name	cursor.field.name()
boolean1	false
second.file	\$current.file.variable
cond.input	true

- 4 Click Add to add a new Format Control record, or click Save to save your updates.

Displayoptions

You may also create a Validity Lookup option in the Options menu by creating a displayoption record. For detailed information on creating displayoptions, refer to *Display Application* in *System Tailoring*.

Use the following field values in the displayoption record:

Field	Value
Default Label	Validity Lookup
Condition	true
RAD Application	validate.fields
Separate Thread?	true

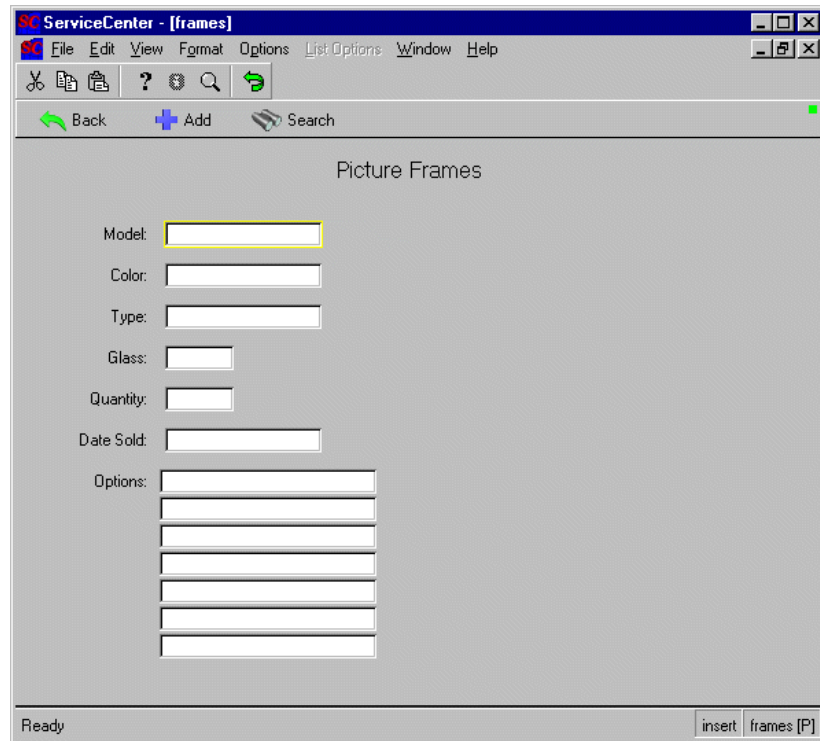
Use the following parameter values in the displayoption record:

Names	Value
name	cursor.field.name()
second.file	\$L.file
cond.input	val("true", 4)

Examples

The following examples show how validity table processing runs when invoked from a Format Control option, a Format Control subroutine call, or a displayoption.

This example uses a file called frames, a sample table used to track the selling of picture frames. The fields in this file represent different data types that can be validated against various range and value specifications. The frames form is shown in Figure 2-23 on page 98.



The screenshot shows a window titled "ServiceCenter - [frames]". The menu bar includes File, Edit, View, Format, Options, List Options, Window, and Help. Below the menu bar is a toolbar with icons for Cut, Copy, Paste, Help, Settings, Search, and Refresh. A secondary toolbar contains Back, Add, and Search buttons. The main area is titled "Picture Frames" and contains the following fields:

- Model:
- Color:
- Type:
- Glass:
- Quantity:
- Date Sold:
- Options:

The status bar at the bottom shows "Ready" on the left and "insert frames [P]" on the right.

Figure 2-23: Sample form for the frames file

The Database Dictionary for the frames file is shown in Figure 2-24 on page 99.

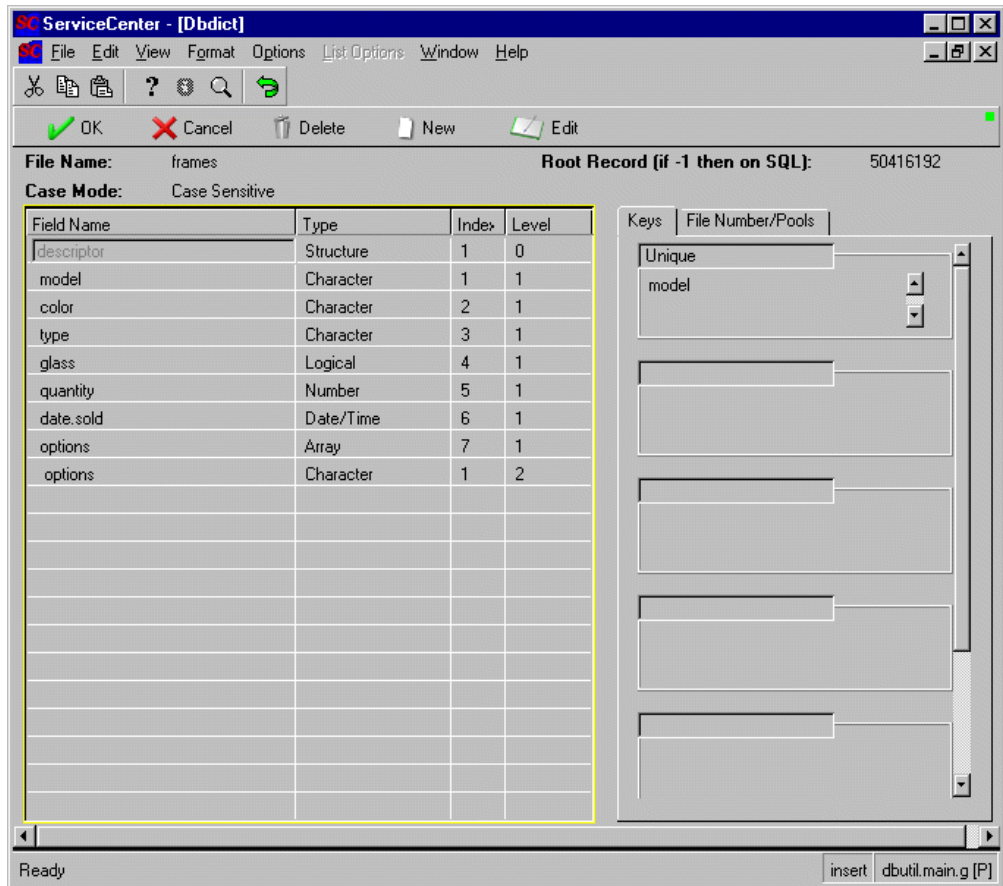


Figure 2-24: Database Dictionary for the frame file

The following values are valid for the indicated fields:

Field	Value
color	Blue, gold, silver, red, natural
	Note: The acceptable values for the color field are contained in a separate file called colors, in which each color is defined as a separate records.
type	Wood, plastic, metal
glass	True or false, NULLS are not allowed

Field	Value
quantity	Must be greater than 0, but less than 50
date.sold	Must be greater than or equal to today's date and less than 7 days from today's date. The System Administrator has the authority to bypass this constraint.
options	Cleaned, high-gloss, matted finish, gift wrapped, shipping wrapped, shipped, carry-out

Validity Table Specifications

You must create a validity table record for each field in the frames file that you want to validate.

Color Definitions

The validity table specifications for the **color** field in the frames file are listed in the following table.

Field	Value
Field Name	color
Files/Formats	frames
Field Type	Character
Validate Cond	true
Allow NULL	false
Bypass Cond	true
Condition	true
Filename	colors
	Note: All valid colors are defined in the <i>color</i> file. This file and a colors form were created for the purpose of this example.
Allow Many	false
Val Query	color=\$vrfield
Val Error Msg	The color is not valid. Select one from the list
Select Query	true
No Recs Opt	Proceed
Lookup Fields	color
Val Fields	color

ServiceCenter - [Data Validation]

File Edit View Format Options List Options Window Help

OK Cancel Previous Next Add Save Delete Find Fill

Validity Table Specifications

- SUMMARY -

Field Name: color Sequence:

Files/Formats: frames Field Type: Character

Array?

Field Desc: Case:

Unique ID:

Standard Info Secondary File Query Info Alternate Application

Validate Cond: true

Allow NULL: false

NULL Default:

Bypass Cond: true

Weight Factor:

Standard Info Secondary File Query Info Alternate Application

Condition: true

Filename: colors Query Type:

Allow Many?: false

Val Query: color=\$vfield

Val Sorts:

Val Error Msg: The color is not valid. Select one from the list.

Select Query: true

Select Sorts:

No Recs Opt: Proceed QBE Fmt:

Fields to Copy

Lookup Fields: color Val Fields: color

Validity.summary.g [P]

Figure 2-25: Validity table specifications for the color field

Date Sold Definitions

The validity table specifications for the `date.sold` field in the frames file are listed in the following table.

Field	Value
Field Name	date.sold
Files/Formats	frames
Field Type	Date/Time
Validate Cond	true
Allow NULL	false
Bypass Cond	true
Error Msg	The date must be greater than today and less than 7 days from now.
Range Min	<code>\$vrfield>=date(tod())</code>
Min Desc	> today's date
Relation	and
Range Max	<code>\$vrfield<=date(tod())+'7 00:00:00'</code>
Max Desc	< 7 days from Today
Range Cond	true

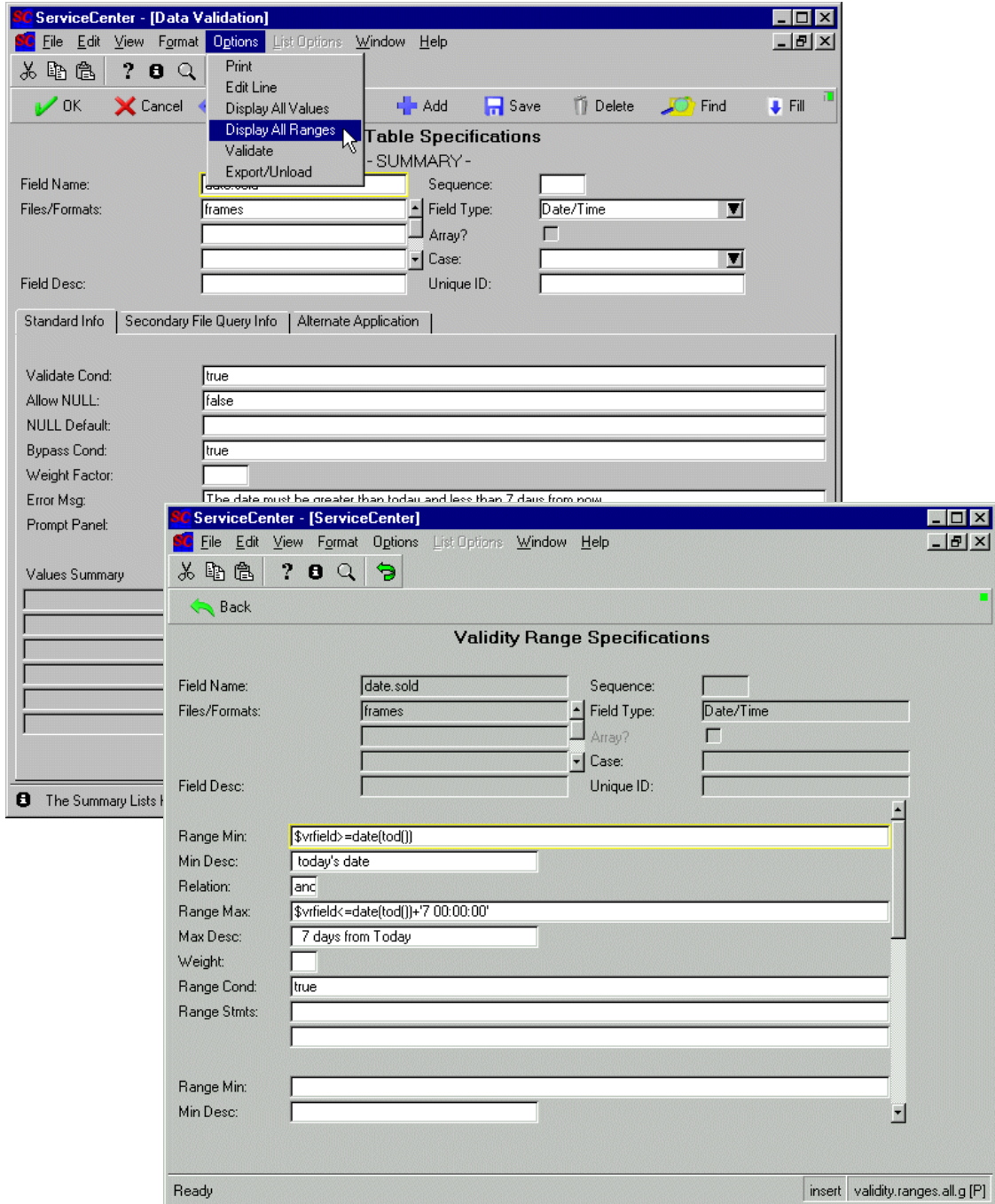


Figure 2-26: Validity table specifications for the date.sold field

Glass Definitions

The validity table specifications for the **glass** field in the frames file are listed in the following table.

Field	Value
Field Name	glass
Files/Formats	frames
Field Type	Character
Validate Cond	true
Allow NULL	true
Bypass Cond	true
Value	true
Cond	true
Desc	true (glass provided)
Value	false
Cond	true
Desc	false (glass not provided)

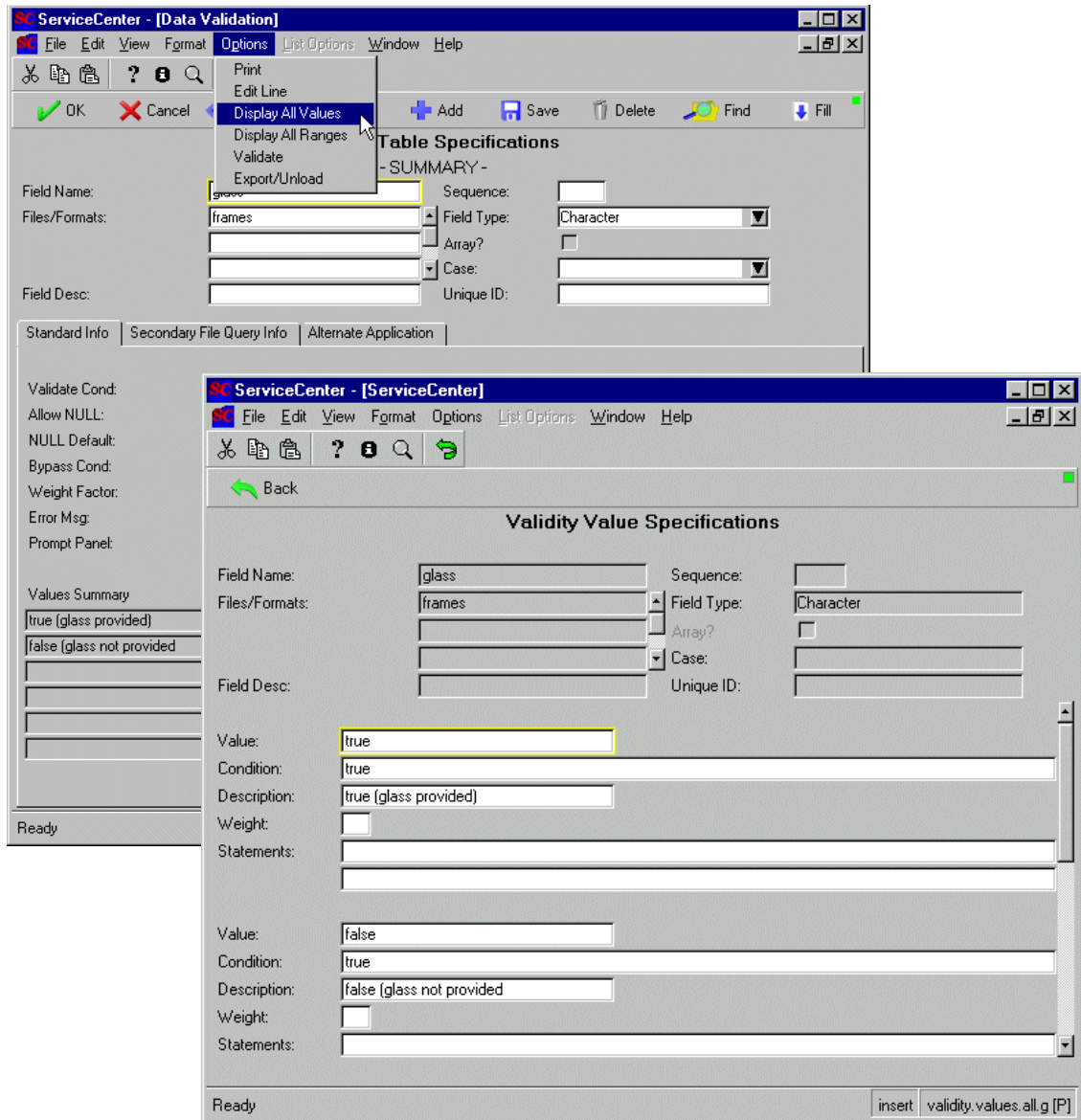


Figure 2-27: Validity table specifications for the glass field

Options Definitions

The validity table specifications for the **options** field in the frames file are listed in the following table.

Field	Value
Field Name	options
Files/Formats	frames
Field Type	Character
Array	true
Validate Cond	true
Allow NULL	true
Bypass Cond	true
Error Msg	Select a valid option.
Value	cleaned
Cond	true
Desc	cleaned
Value	high-gloss
Cond	true
Desc	high-gloss
Value	matte finish
Cond	true
Desc	matte finish
Value	gift wrapped
Cond	true
Desc	gift wrapped
Value	shipping wrapped
Cond	true
Desc	shipping wrapped
Value	shipped
Cond	true
Desc	shipped

Field	Value
Value	carry-out
Cond	true
Desc	carry-out

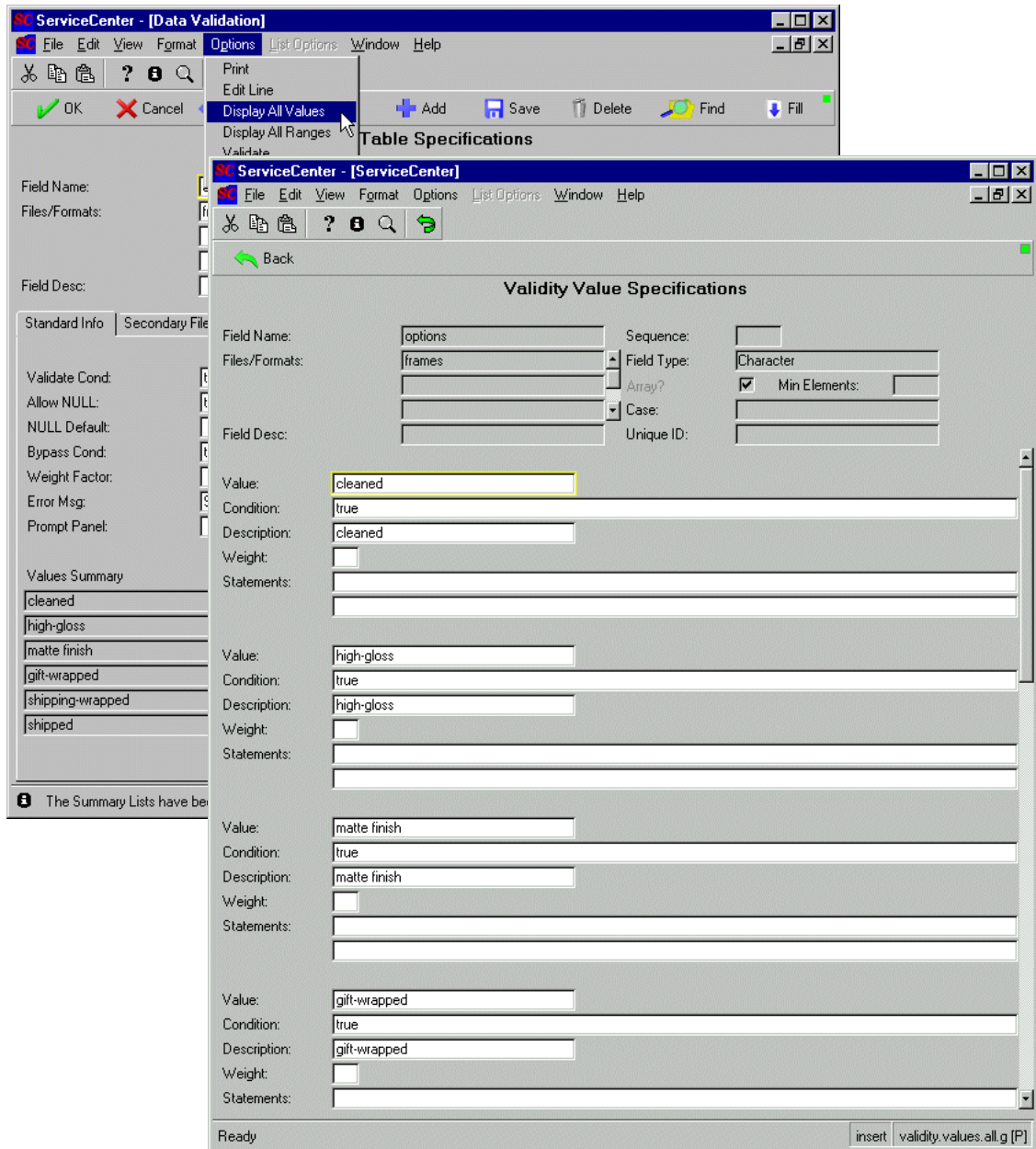


Figure 2-28: Validity table specifications for the options field

Quantity Definitions

The validity table specifications for the **quantity** field in the frames file are listed in the following table.

Field	Value
Field Name	quantity
Files/Formats	frames
Field Type	Number
Validate Cond	true
Allow NULL	false
Bypass Cond	false
Range Min	$\$vrfield > 0$
Min Desc	> zero
Relation	and
Range Max	$\$vrfield \leq 50$
Max Desc	< = fifty
Range Cond	true

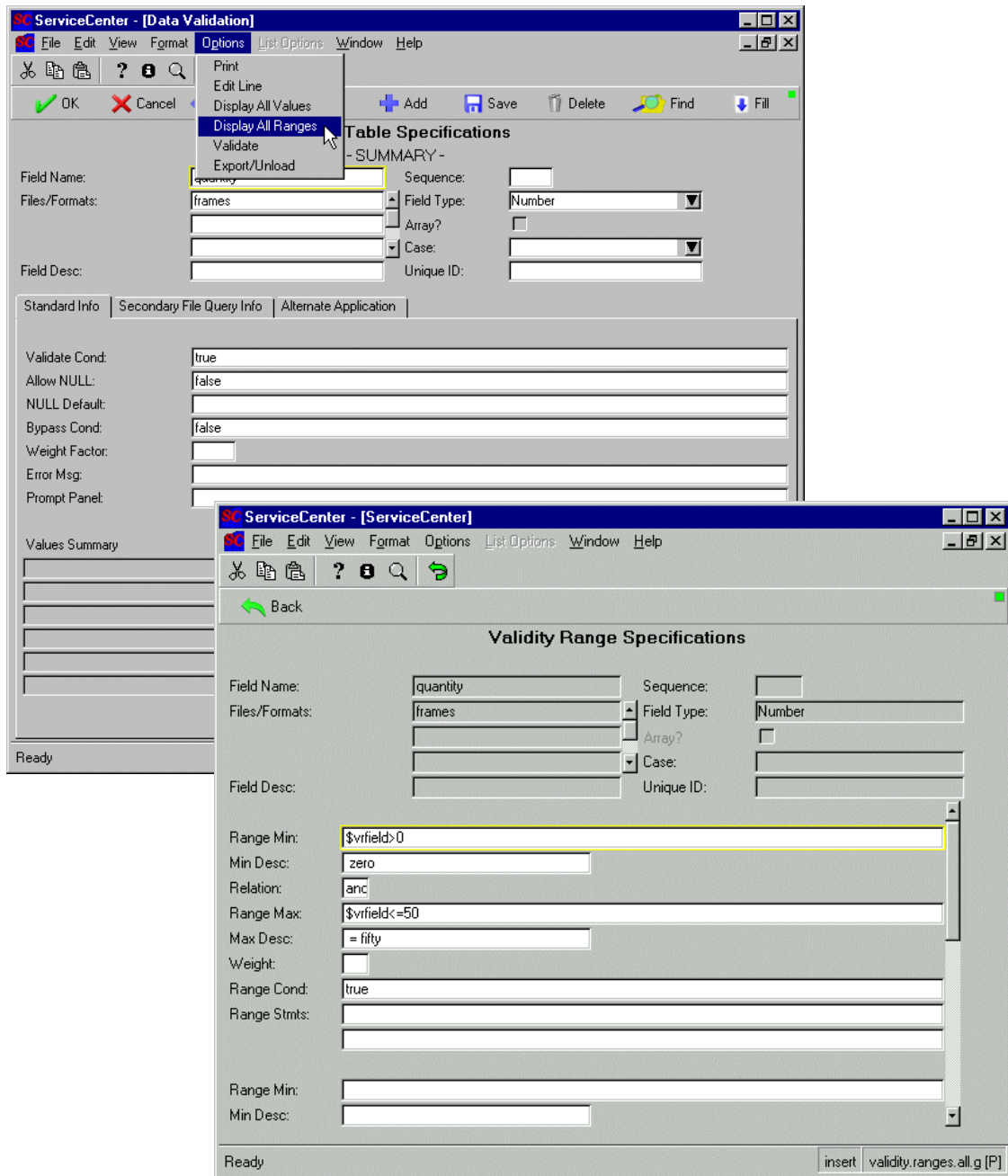


Figure 2-29: Validity table specifications for the quantity field

Type Definitions

The validity table specifications for the **type** field in the frames file are listed in the following table.

Field	Value
Field Name	type
Files/Formats	frames
Field Type	Character
Validate Cond	true
Allow NULL	false
Bypass Cond	true
Error Msg	Type must be wood, plastic, or metal.
Value	wood
Cond	true
Desc	wood
Value	plastic
Cond	true
Desc	plastic
Value	metal
Cond	true
Desc	metal

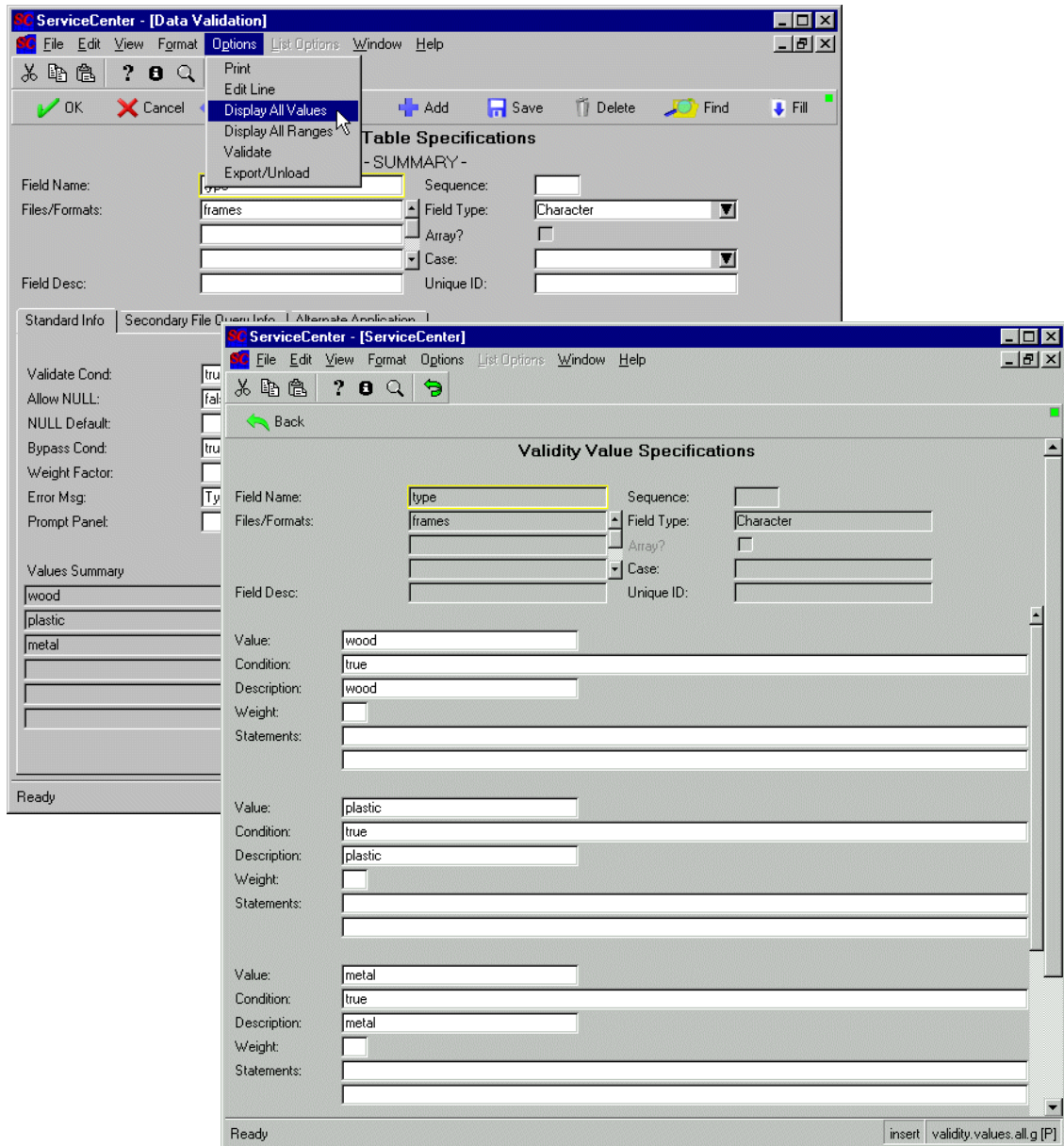


Figure 2-30: Validity table specifications for the type field

Invoking Validity with Format Control

For this example, we will create an Options menu option for Validity Lookup that allows the user to manually validate each field on the `frames` form. We will also invoke a RAD call that can validate all the fields on the form automatically when the user attempts to save a record in the `frames` file. The two Format Control processes invoked are:

- Additional Options
- Subroutines

Additional Options Values

Using the procedure discussed on page 93, create an Additional Options record for the `frames` form using the following values:

Field	Value
Form to Display	frames
Option	1
Condition for Option	true
Command	Validity Lookup
Description	Validity lookup
Application	validate.fields
Error Message	The original record is displayed.
Reset on Return	true

Enter the appropriate parameters for `validate.fields` in the **Name** and **Value** fields using the information from the following table:

Names	Values
name	cursor.field.name()
second.file	\$file
cond.input	val("true", 4)

Note: The data passed to the boolean parameter must be a type 4 (hence the VAL() statement for the Value). As an alternative, a \$variable could be defined in the *initialization expressions* (e.g. \$flag=true) and then passed to the subroutine.

Subroutine Values

Using the procedure discussed on page 95, create a Subroutines call for the frames form using the following values:

Field	Value
Application Name	validate.fields
Error Message	The original record is displayed
Add	true
Update	true
Before	true

Enter the appropriate parameters for `validate.fields` in the Name and Value fields using the information from the following table:

Names	Values
second.file	\$file

Invoking Validity with a displayoption

A Validity Lookup menu option can also be created with a Display application *displayoption*. For information on locating Screen IDs and creating displayoptions, refer to *Display Application* in *System Tailoring*.

Example 1: Run-Time Validation

We are now ready to create and validate a record using the validity specifications we have defined. In the following example, we will enter several invalid values and attempt to add the record to the frames file. By doing so, we will invoke the run-time validation process we have set up in *Subroutine Values* on page 115.

To validate a form at run-time

- 1 Open a new picture frame record in Database Manager.
- 2 Complete the form with the following values:

Field	Value
Model	1500-A
Color	purple
Type	wood
Glass	true
Quantity	5
Date Sold	09/30/01
Options	antique cleaned stained carry-out

Note: The values in the Color, Date Sold, Options #1, and Options #3 fields are invalid. Assume that the current date is 08/09/01.

The screenshot shows a window titled "ServiceCenter - [frames]" with a menu bar (File, Edit, View, Format, Options, List Options, Window, Help) and a toolbar (Back, Add, Search). The main area is titled "Picture Frames" and contains a form with the following fields:

- Model: 1500_A
- Color: purple
- Type: wood
- Glass: true
- Quantity: 5
- Date Sold: 09/30/01
- Options: antique, cleaned, stained, carry-out, (empty), (empty), (empty)

A box labeled "Invalid values" has arrows pointing to the Color, Date Sold, and the first three Options fields. The status bar at the bottom shows "Ready" and "insert frames [P]".

Figure 2-31: New record in the frame file prior to validation

3 Click Add.

The validation routine detects the invalid color value and displays a QBE list of valid values from which to choose. The following message is displayed in the status bar: *The color is not valid. Select one from the list.*

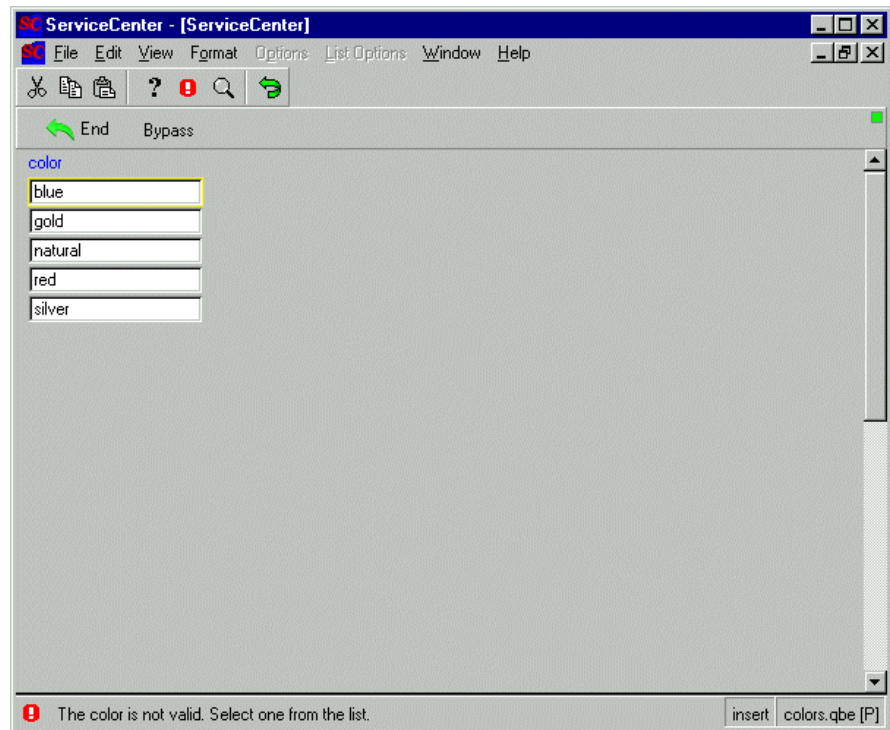


Figure 2-32: QBE list of valid field values for the Color field

- 4 Make one of the following choices:
 - Select a valid color and press Enter.
 - Click End to return to the record without changing it.
 - Click Bypass to bypass current validity table processing and force the application to accept the current value in the field. To have the capability of bypassing the validation, the **Bypass Cond** field in the validity record for this field must evaluate to *true*.

If you select a valid color or bypass the validation of the **Color** field, the next field is validated. In this example, the Validity Table Correction Screen is displayed indicating that an invalid value has been entered in the **Date Sold** field. The following customized error message is displayed in the status bar: *The date must be greater than today and less than 7 days from now.*

The invalid date from the form is displayed in two places initially: the **Corrected Value** field; and in red in the **Original Value** field. The valid range you specified in the validity record is displayed in the **Valid Ranges** array.

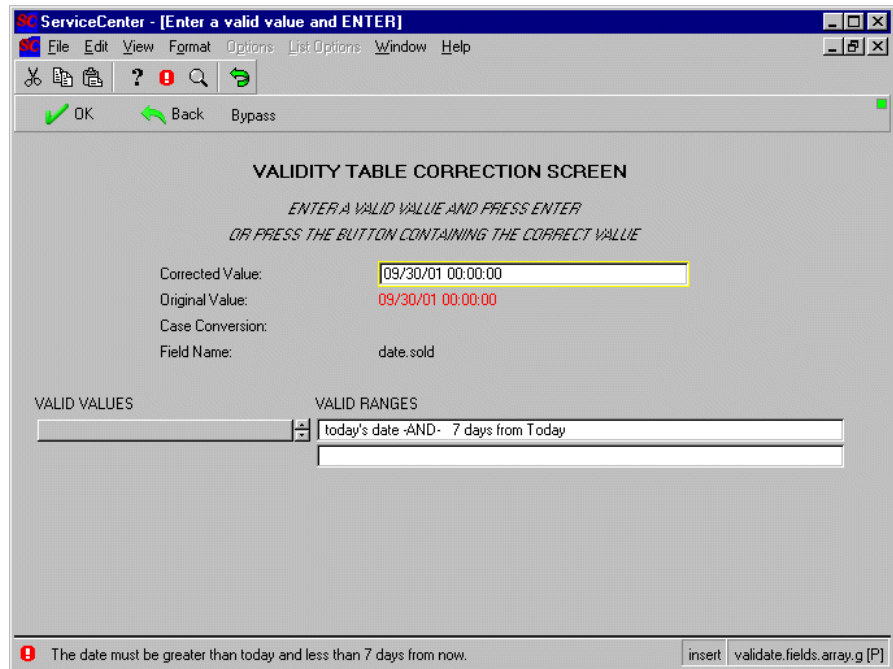


Figure 2-33: Correction screen for an incorrect range value

- 5 Change the date in the **Corrected Value** field to the correct date.
 - 6 Click OK to apply the corrected date to the Date Sold field.
- Note:** You may bypass this validation and move to the validation of the next field or click Back to return to the original record without applying any changes.

The next field is validated. In this example, the process detects an invalid value in the Options array. A drop-down list of valid options is displayed. The following message is displayed in the status bar: *Select a valid option.*

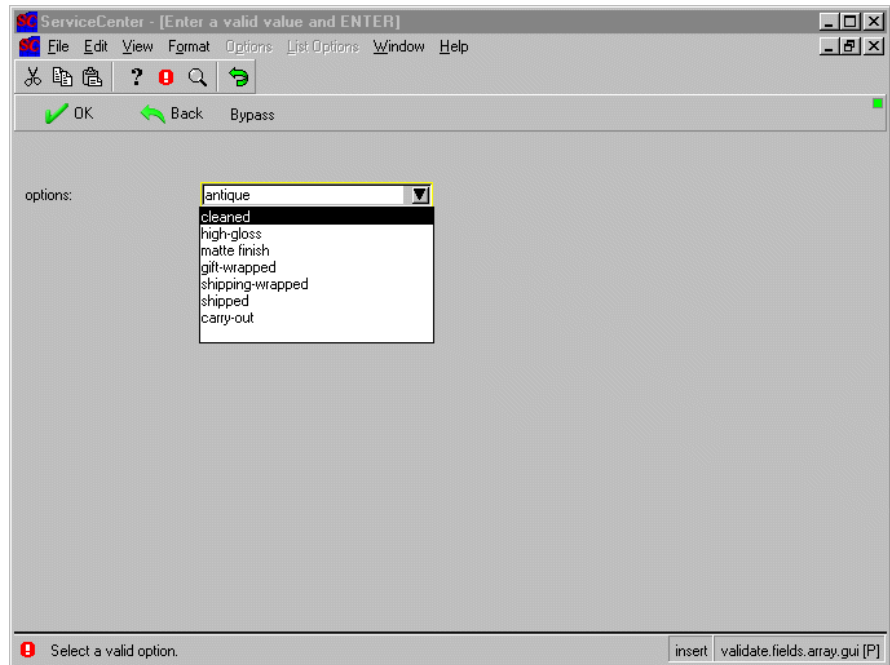


Figure 2-34: Drop-down list of valid options for the frame file

- 7 Select a valid option from the list.
- 8 Click OK or press Enter.
The next option in the array is validated.
- 9 Repeat the selection and validation process as many times as needed to remove invalid options from the new record.

When you have validated all the fields in the form, the record is added to the frames file. The original record is displayed and the following message appears in the status bar: *frames record added*.

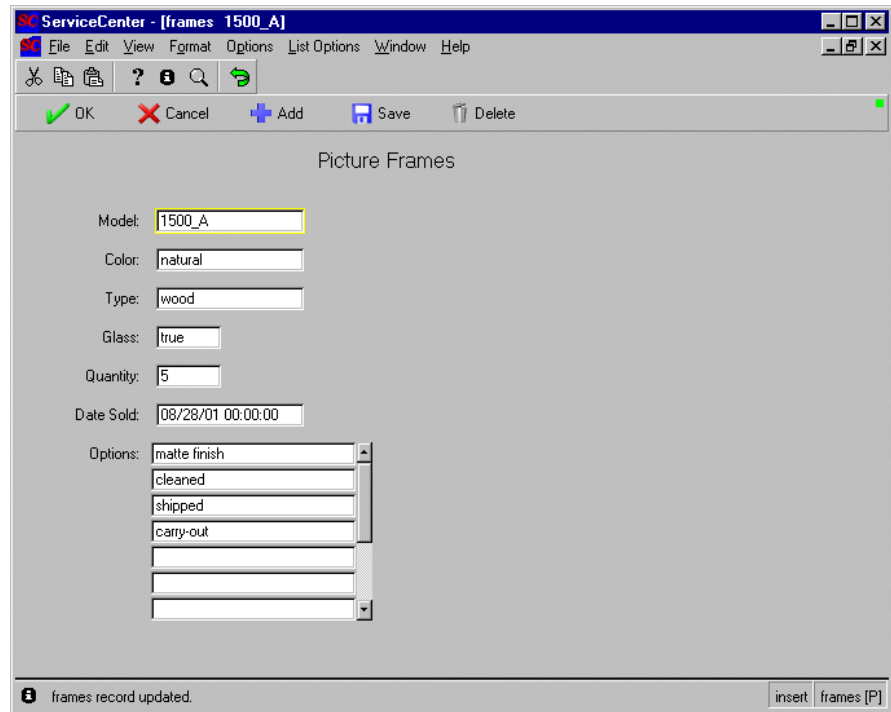


Figure 2-35: Valid record added to the frames table.

Example 2: Assisted Entry

In this example, we demonstrate how you can use validity table processing to complete the fields in a new record. In this case, you do not need to know the acceptable values for each field and can use the validation process like the Fill function.

To enter data using validity table processing

- 1 Open a new picture frame record in Database Manager.
- 2 Enter a value in the Model field.
- 3 Click Add.

Because the Color field is blank and the validity record for this field does not allow a NULL value, validity processing prompts you to select a valid color.

- 4 Select a color from the QBE list displayed (Figure 2-32 on page 118) and press Enter.

Because the **Date Sold** field is blank and the validity record for this field does not allow a NULL value, the Validity Table Correction Screen is displayed (Figure 2-33 on page 119), and the following message appears in the status bar: *The date must be greater than today's date and less than 7 days from now.*

- 5 Select one of the following procedures:
 - Enter a valid date and click OK.
 - If you do not want to enter a date at this time, leave this field blank and click Bypass.

Because the **Quantity** field is blank and the validity record for this field does not allow a NULL value, the Validity Table Correction Screen is displayed (Figure 2-33 on page 119), and the following message appears in the status bar: *The value NULL is not valid for the field quantity.*

- 6 Add the quantity in the **Corrected Value** field.
- 7 Click OK or press Enter.

Because the **Type** field is blank and the validity record for this field does not allow a NULL value, a drop-down list is displayed, and the following message appears in the status bar: *Type must be wood, plastic, or metal.*

- 8 Select a valid frame type from the list.
- 9 Click OK or press Enter.

The picture frame record is displayed with the required (and non-bypassed) fields completed. The following message appears in the status bar: *frames record added*.

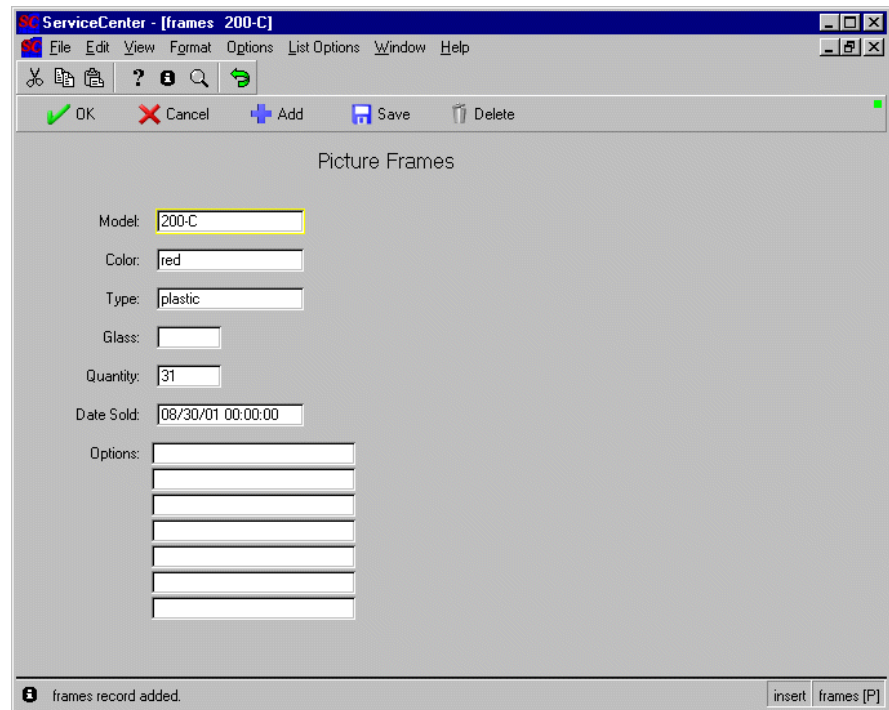


Figure 2-36: Record partially completed by validity table processing

10 Complete the remaining fields in the record manually or use the Validity Lookup option (explained in the following section).

11 Click Save.

Validity table processing continues with the **Options** field. If invalid options are entered, the system displays a drop-down list of acceptable values from which to choose.

12 Select the appropriate options from this list and click OK or press Enter.

The following message appears in the status bar: *frames record updated*.

Example 3: Validity Lookup Option

In the previous example, we were able to save the record with two blank fields (**Glass** and **Options**), because the validity record for those fields allowed NULLs. If we do not know the acceptable values for the **Options** field, however, we are likely to enter invalid data that will invoke validity table processing when we attempt to save the record. We can avoid errors when selecting options in this array by using the Validity Lookup option we created for the **frames** form in Format Control.

To use the Validity Lookup option

- 1 Access the same record created in the previous example (Figure 2-36 on page 123).

This record was validated with the assisted entry method, which left two fields, the glass field and the options field blank. Because the validation specifications for these fields permit NULL values, validation was not performed on these fields before the record was saved. Validity Lookup executes validity table processing for these fields and helps you select a valid value.

- 2 Put the cursor in the first blank line of the **Glass** field.
- 3 Select **Options > Validity Lookup**.

A form containing a drop-down list of valid selections for the Glass field is displayed. Because this is a logical field, there are only two selections.

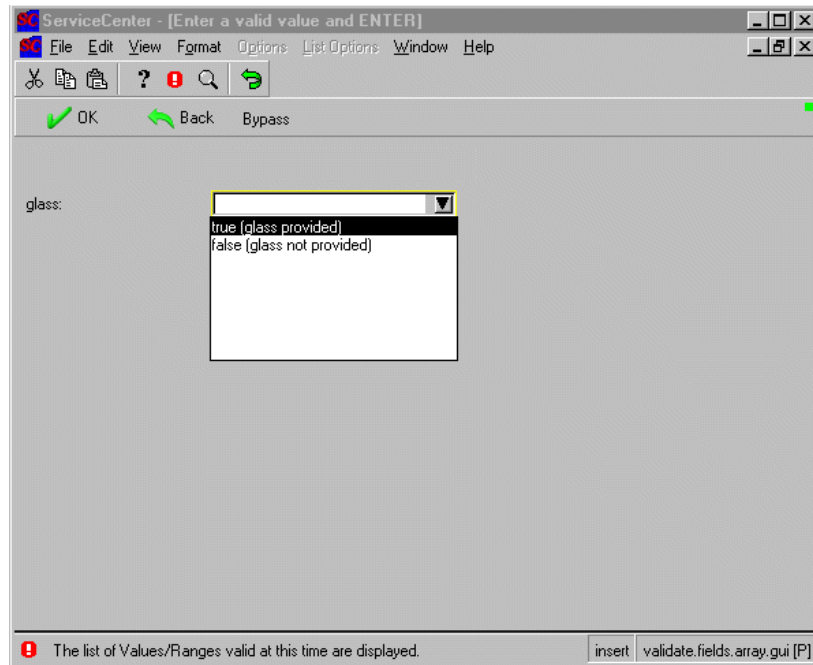


Figure 2-37: Drop-down list of valid selections for the Glass field

- 4 Select either true or false.
- 5 Click OK or press Enter.
- 6 Put the cursor in the first blank line of the Options array.
- 7 Select Options > Validity Lookup.

A form containing a drop-down list of valid options is displayed.

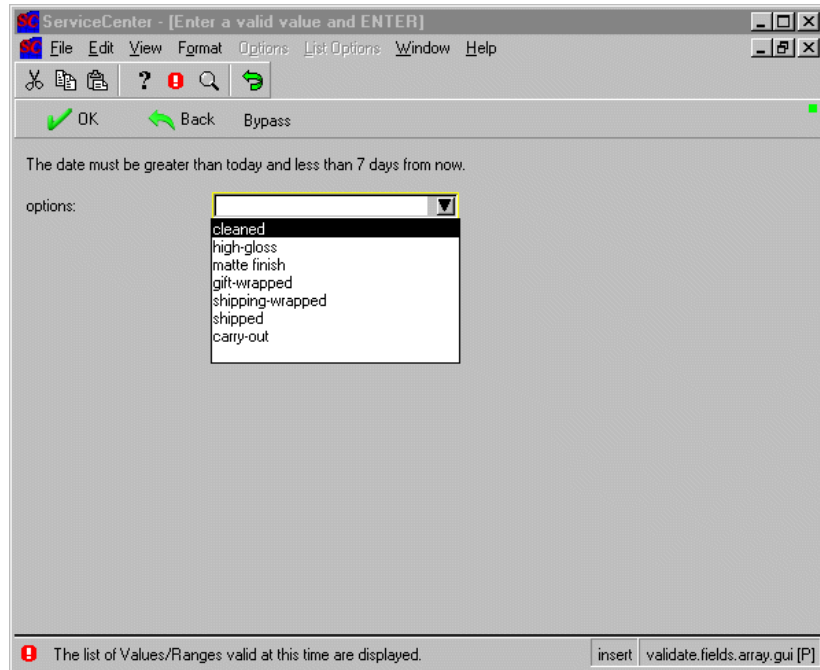


Figure 2-38: List of valid options and date error message.

Note: The date appearing on this form is no longer valid according to the range defined in the validity table specifications. When you attempt to save this record, the validation routine for the **Date Sold** field will be invoked. You must either correct the date or bypass the validation routine to save the record.

- 8 Select a value from the list.
- 9 Click OK or press Enter.
- 10 Put the cursor in the next blank line of the **Options** array and repeat the validation process until all the appropriate options have been selected.
- 11 Click Save.

Example 4—Special Processing

The following special processing statements can be used to define unique relationships between fields:

- **Special validation conditions:** In the first example, a certain model of picture frame (the 1900 Deluxe) comes with a blue, wooden frame and includes the glass. We want the validation routine to enter this information automatically when processing records for the 1900 series. When any other model number is entered, we want normal validity table processing to occur.
- **Modified processing order:** In the second example, we will alter the default processing order (alphabetical) and have the fields validated in the order in which they appear in the form.

Special Validation Conditions

The processing statements that establish the default values for the 1900 Deluxe picture frame must be defined as value details in the validity table.

To establish the value detail specifications

- 1 Open a blank Validity Table Specifications record.
- 2 Enter the following values:

Field	Value
Field Name	model
Files/Formats	frames
Field Type	character
Validate Cond	model in \$file="1900"
Allow NULL	false
Bypass Cond	false

- 3 Click New.
- 4 Select **Options > Display All Values**.
The Validity Value Specifications form is displayed.

5 Add the following value specifications:

Field	Value
Value	1900
Cond	true
Desc	1900 (Deluxe)
Stmts	color in \$val="blue" date.sold in \$val=tod() type in \$val="wood" glass in \$val=true

6 Click Back to return to the validity record.

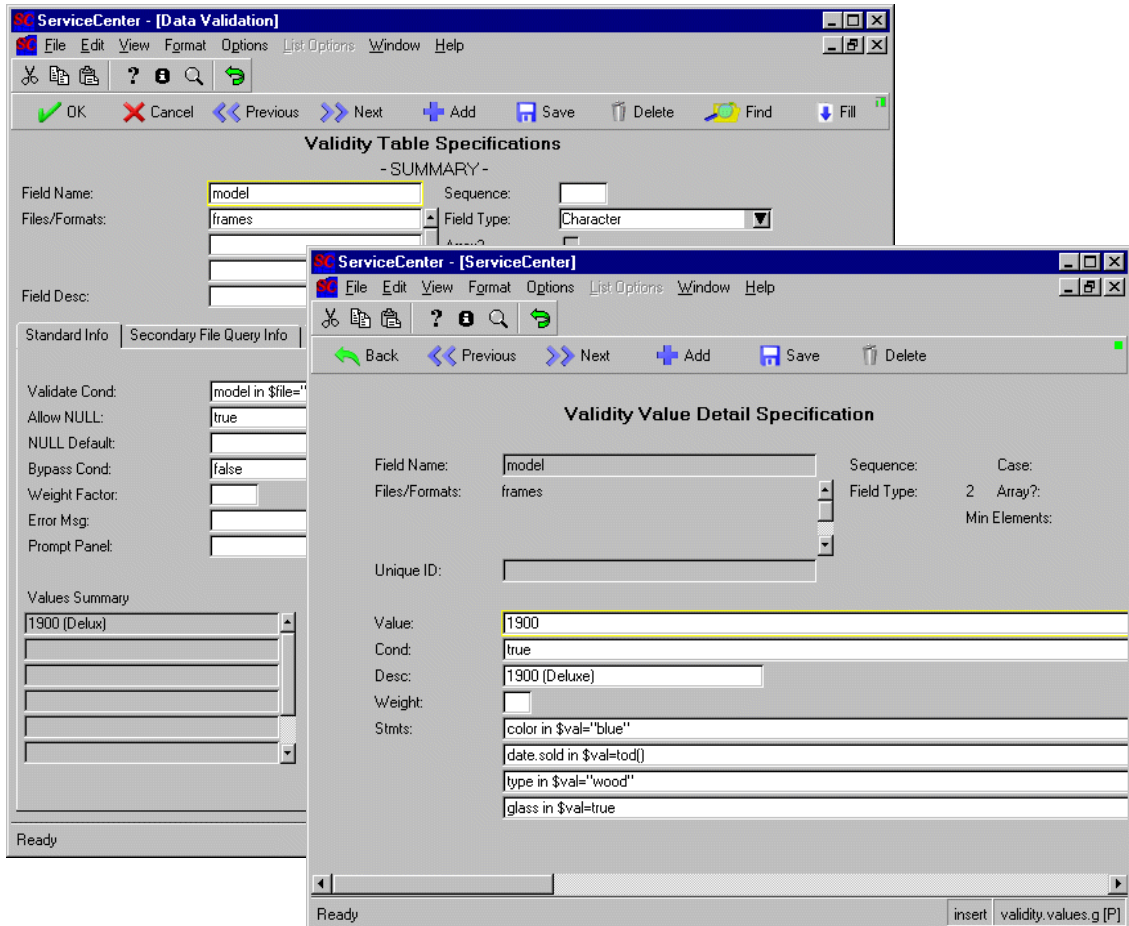


Figure 2-39: Validity table specifications for special processing

The value statements for the **model** field will be executed when the **model** field contains the value of 1900. When this occurs, certain fields in the **frames** file will be set to the values specified in the **Stmts** array. If any value defined in these statements is not valid (the validity table definition for the field does not contain the same value), the user will be prompted to correct the value. To allow for normal validity table processing of model numbers other than 1900, you must enter a statement in the **Validate Cond** field that applies the special processing conditions to the 1900 model only.

Fields named in value statements do not have to be defined in the validity table. For example, the frames file might have a field called **shipped.by** that is not validated by the system. If you want to have all your Model 1900 picture frames shipped by UPS, set the value in this field to **UPS** for by adding the following condition statement to the **Stmts** array:

```
shipped.by in $val="UPS"
```

Modified Processing Order

To change the processing order of the validation routine, you must create an initialization expression in the Format Control record that invokes validity for the frames file. The new order is defined by a variable that is passed to the validation routine (**validate.fields**) in the Subroutines process. For information on editing Format Control records, refer to *Format Control in System Tailoring*.

To redefine the validity routine processing order using Format Control

- 1 Open the frames Format Control record.
- 2 Enter the following statement in the **Initialization Expressions** field:

```
$fieldlist={"model", "color", "type", "glass", "quantity", "date.sold", "options"}
```

The variable **\$fieldlist** defines the new validation processing order of the fields in the frame file.

Note: Alternately, you could use sequence numbers to control the processing order for the fields in the frames file.

Important: Be sure to use curly braces {} in this expression.

3 Click Save.

ServiceCenter - [Format Control frames]

File Edit View Format Options List Options Window Help

OK Cancel Add Save Delete

Views Queries Calculations Validations Subroutines Add Options Privileges

Format Control Maintenance - Main Information

Name: View:

File Name:

System:

Query Format:

Default QBE Fmt:

Save Copy Stored Form Name Run Script Use Default Sort

Default Sort sequence for queries

Initialization Expressions

Ready insert formatctrl.maint.g [P]

Figure 2-40: Format Control record containing an initialization expression

4 Click Subroutines.

The Subroutines process is displayed.

- 5 Pass the variable you have defined (\$fieldlist) to `validate.fields` by adding the following parameter and value to the subroutine call:

Parameter	Value
names	\$fieldlist

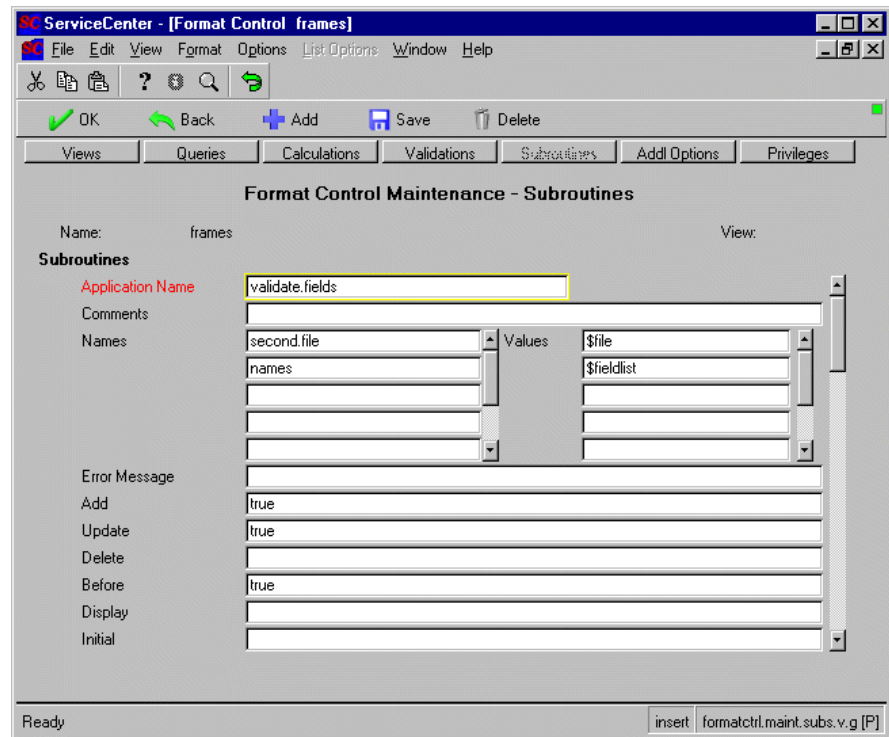


Figure 2-41: Format Control subroutine call—passing a new variable

- 6 Click Save.

Testing your Modifications

To test the changes you have made to the validity processing of the *frame* file

- 1 Open a blank record in the frames file.
- 2 Enter 1900 in the Model field.
- 3 Click Add.

The Validity Table Correction Screen for the **Quantity** field is displayed. The value in this field was not predefined in the Format Control initialization expression and cannot be NULL.

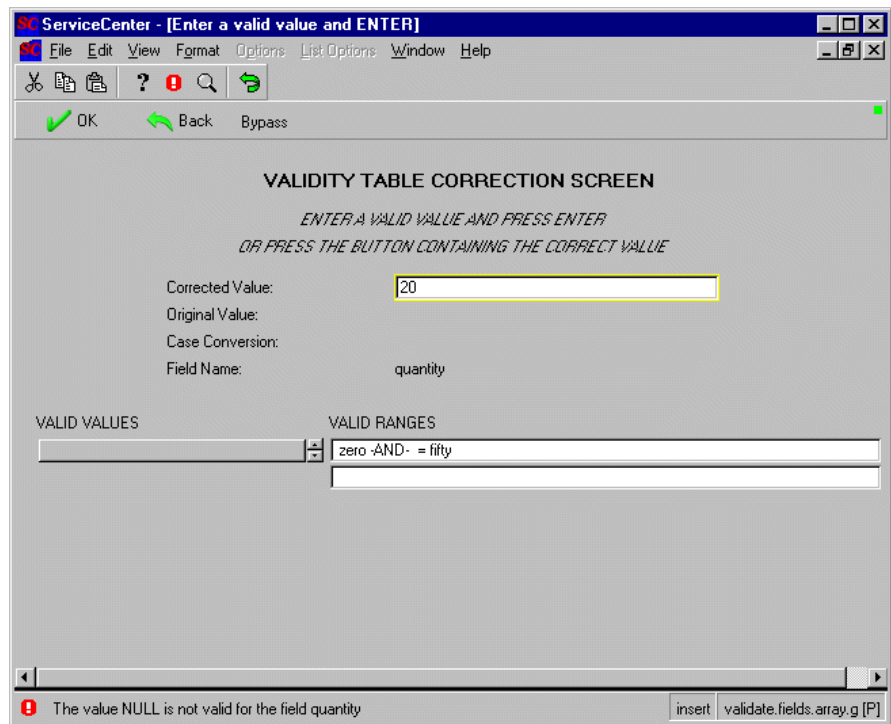


Figure 2-42: Validity Table Correction Screen for the Quantity field

- 4 Enter a quantity in the Corrected Value field.
- 5 Click OK or press Enter.

The picture frame record for the Model 1900 is displayed with the predefined values entered in the appropriate fields. The following message is displayed in the status bar: *frames record added*.

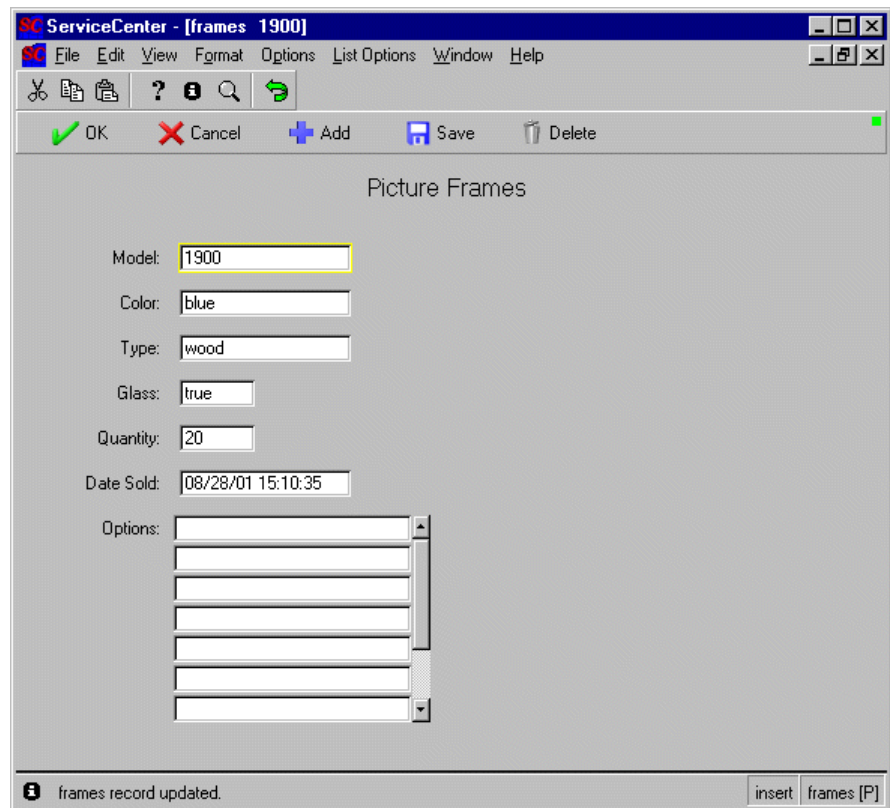


Figure 2-43: Automatically validated picture frame record

- 6 Place the cursor in the **Options** field and select **Options > Validity Lookup** to display a list of valid options.
- 7 Select the options that apply to the Model 1900 Deluxe frame.
- 8 Click Save.

Run-Time Evaluation

The following steps are a summary of the sequence of events and the options available to a user when validity table processing is invoked against a file.

- 1 Extract all field definitions associated with the file. If no validity table definitions exist, all values for all fields are valid.

- 2 Are any Values or Ranges defined for the field? If no, proceed to step 11. If yes, proceed to the next step.
- 3 Is the **Cond** field equal to *true*? If no, proceed to step 11. If yes, proceed to the next step.
- 4 Is any one of the Values equal to the value of the field and does the **Cond** field for that Value evaluate to *true*? If yes, then the field is valid—proceed to step 11. If not, proceed to the next step.
- 5 Is the field value within any of the defined Ranges and does the **Cond** field for that Range evaluate to *true*? If yes, then the field is valid—proceed to step 11. If not, proceed to the next step.
- 6 Do *any* Value or Range conditions evaluate to *true*? If yes, then stop and display all of the Values and Ranges that evaluate to *true*. This is the Validity Table Correction Screen. Click Exit (F3) to return to the original record. Click Bypass (F4) to bypass the validity process for that field (proceed to step 11). Click OK (F2) or press Enter to cause the result of the *cursor.field.contents()* function to be re-evaluated (proceed to Step 4).
- 7 Evaluate the Secondary File Query condition. If it is *true*, proceed to the next step. If it is *false*, proceed to step 11 if the value in the **No Recs Option** field is **proceed**. Return to the original record if the value in the **No Recs Option** field is **return**.
- 8 Execute the Secondary File Query against the indicated file. If one record is found, the value is valid (proceed to step 11). If no records are found, the value is not valid (proceed to step 9). If more than one record is found, display that list of records to the user. From here, the user can select a record (proceed to step 10), or select Bypass (F4) to return to the record.
- 9 Execute the Select Query against the secondary file. If no records are found, proceed to step 10. If one or more records is found, then display that list. Click End (F3) to return to step 6. Press Enter to proceed to step 10.
- 10 Copy the contents of **Field to Copy** to the field being validated. Proceed to step 11.
- 11 Get the Validity Record of the next field to process. If there are more records, proceed to step 2. If there are no more records, then proceed to the next step.
- 12 Update the data record with the corrected values and return to the calling application.

Validating Arrays

The validity table processor can validate scalar and array data fields. The valid scalar data types are number (1), character (2), date/time (3), and logical (4). You can validate arrays (8) of these data types. Other data types are not supported. When you validate an array you can either:

- Validate all elements of the array against one validity definition, or
- Validate each element of the array against a different validity definition.

When you create a Validity Table definition for an array, keep the following items in mind.

- Each element of the array must validate against the same data type. In other words, you cannot validate one element against a character field and then validate another against a number.
- Is each element of the array validated against the same or different tables?
- If validating against one table, what is the minimum number of elements in the array that must contain valid data?
- If validating against multiple tables, which elements must validate against which table?

Example 5—Basic Array Validation

This example shows the validity definitions necessary to support the validation of the *cap.exec* (Execute Capabilities) field in the *operator* file. This array contains the capability words for each operator. In this example, the validity routine checks each element of the array against the *capability* field in the *capability* file.

Figure 2-44 on page 137 shows the standard validity table information for the *cap.exec* field in the *operator* file.

The following field values have been entered:

Field	Value
Field Name	cap.exec
Files/Formats	operator
Field Type	Character
Array	true

Field	Value
Unique ID	operator
Validate Cond	true
Allow NULL	true
Bypass Cond	true
Error Msg	A capability word is invalid.

The screenshot shows a software window titled "ServiceCenter - [Data Validation]". The window has a menu bar (File, Edit, View, Format, Options, List Options, Window, Help) and a toolbar with icons for OK, Cancel, Previous, Next, Add, Save, Delete, Find, and Fill. The main area is titled "Validity Table Specifications - SUMMARY -".

Fields and values shown in the form:

- Field Name: cap.exec
- Files/Formats: operator
- Field Desc: (empty)
- Sequence: (empty)
- Field Type: Character
- Array?:
- Case: (empty)
- Unique ID: operator
- Min Elements: (empty)
- Validate Cond: true
- Allow NULL: true
- NULL Default: (empty)
- Bypass Cond: true
- Weight Factor: (empty)
- Error Msg: A capability word is invalid.
- Prompt Panel: (empty)

At the bottom, there are sections for "Values Summary" and "Range Summary", each with five empty rows. The status bar at the bottom left says "Ready" and the bottom right says "insert validity.summary.g [P]".

Figure 2-44: Validity table specifications for the cap.exec field in the operator file.

Figure 2-45 on page 139 shows the Secondary File Query Info tab for the cap.exec validity record.

The following field values have been entered:

Field	Value
Condition	true
Filename	capability
Allow Many	false
Val Query	capability=\$vrfield
Val Error Msg	"The capability word "+str(\$vrfield)+" is not valid. Please select one from list."
Select Query	true
No Recs Opt	Proceed
Lookup Fields	capability
Val Fields	\$vrfield

Note: The Val Query and Val Fields fields reference the variable \$vrfield. This is a temporary variable that contains the data from the scalar field or the current array element. Refer to Additional Tips for more information.

Standard Info	Secondary File Query Info	Alternate Application
Condition:	<input type="text" value="true"/>	
Filename:	<input type="text" value="capability"/>	Query Type: <input type="text"/>
Allow Many?:	<input type="text" value="false"/>	
Val Query:	<input type="text" value="capability=\$vrfield"/>	
Val Sorts:	<input type="text"/>	
Val Error Msg:	<input +str(\$vrfield)+"="" from="" is="" list."="" not="" one="" please="" select="" type="text" valid.="" value="The capability word "/>	
Select Query:	<input type="text" value="true"/>	
Select Sorts:	<input type="text"/>	
No Recs Opt:	<input type="text" value="Proceed"/>	QBE Fmt: <input type="text"/>
Lookup Fields:	<input type="text" value="capability"/>	Val Fields: <input type="text" value="\$vrfield"/>
	<input type="text"/>	<input type="text"/>
	<input type="text"/>	<input type="text"/>

Figure 2-45: Secondary File Query Info tab

Operator Record

In the operator record in Figure 2-46 on page 140, the IMAdmin capability is incorrect. Validity is invoked with a Format Control Subroutines call to `validate.fields`. For procedures on invoking validity table processing from Format Control, refer to *Invoking Validity with Format Control* on page 114.

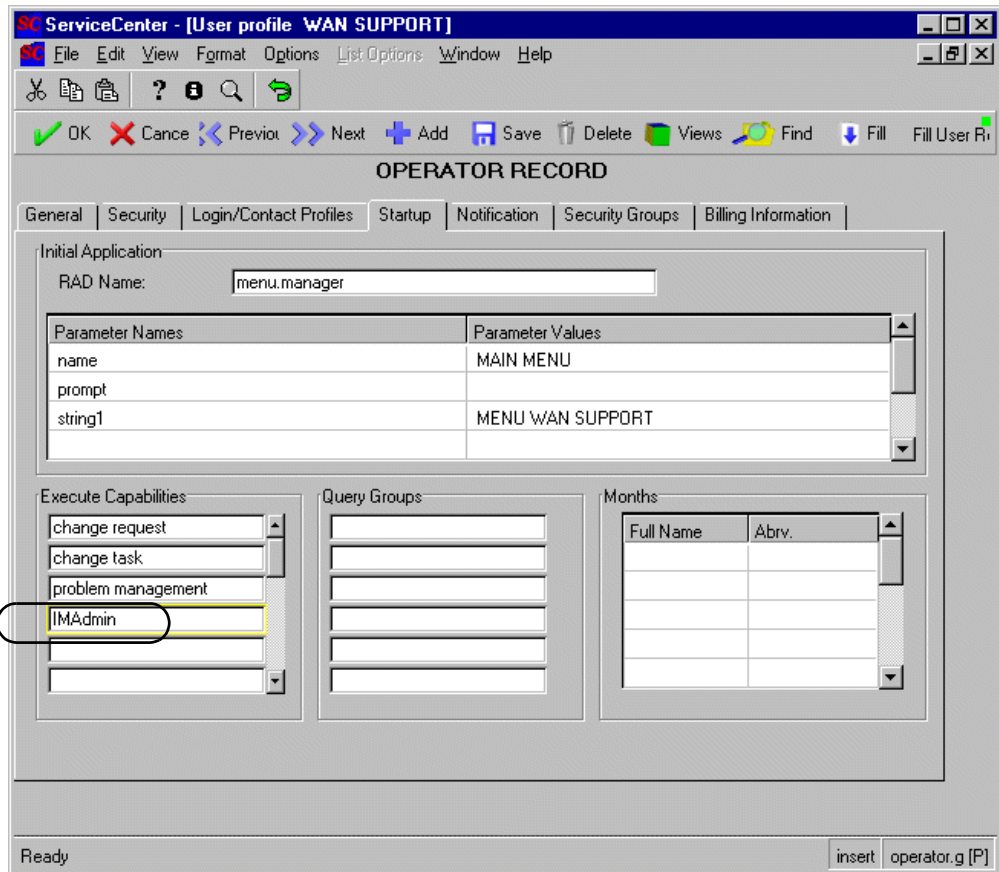


Figure 2-46: Operator record with incorrect capability

When the operator record is updated, validity table processing detects the invalid capability word and displays a QBE list of valid capabilities from the capability file.

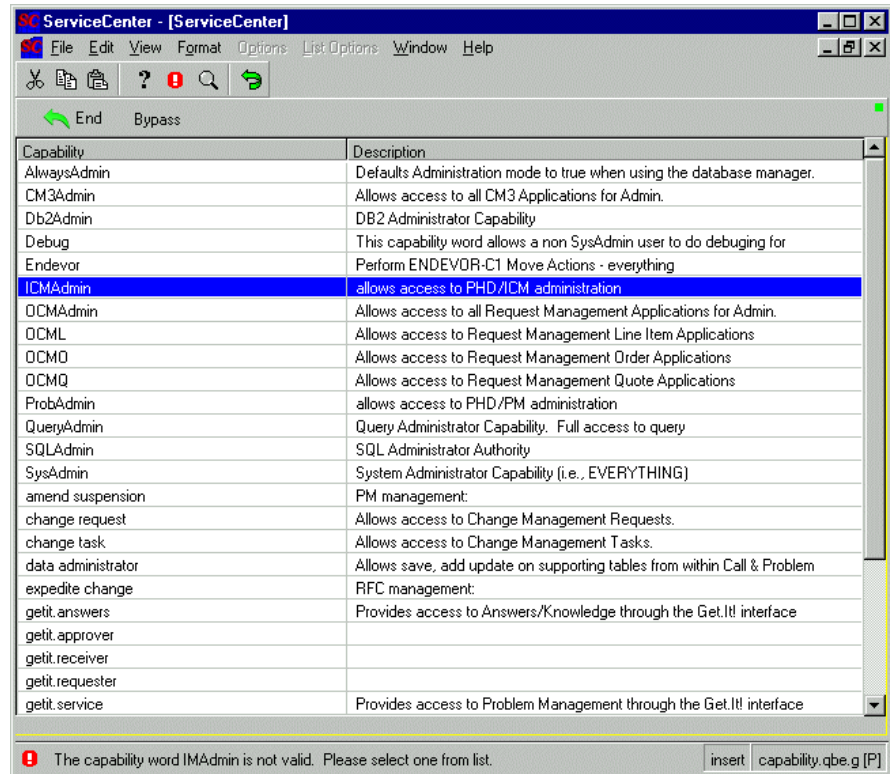


Figure 2-47: QBE list of valid capability words

Double-click on valid capability word to replace the invalid capability word in the operator record (Figure 2-48 on page 142).

Initial Application

RAD Name:

Parameter Names	Parameter Values
name	MAIN MENU
prompt	
string1	MENU WAN SUPPORT

Execute Capabilities

- change request
- change task
- problem management
- ICMAdmin**

Query Groups

Months

Full Name	Abrv.

Figure 2-48: Corrected capability word

Additional tips

The validity routine establishes the `$array.cnt` global variable when validating an array. This counter keeps track of which element within the array is being validated. You can refer to this variable in your validity definition to control validity table processing. The best example is to set the **Validate Cond** to check this field for a particular value so that a particular array element validates against a particular validity definition. If you modify the contents of this variable, the results of validity processing are unpredictable.

The validity routine establishes the `$vrfield` global variable when validating scalar or array data. This variable contains the contents of the scalar field or of the array element. You can refer to this variable in your validity definition to control validity table processing. An example of when to use this variable is in the **Val Query** field. For instance, `name=$vrfield`. You can also refer to `$vrfield` in the **Val Fields** array. This is especially helpful when validating an array element against a table. By using the `$vrfield` variable you can copy the contents of a field from the lookup file to the element being validated without regard to which element is being validated.

3 Notification Engine

CHAPTER

The Notification Engine is primarily responsible for sending messages generated by ServiceCenter events, such as opening or closing an incident ticket. Administrators can edit these messages, add new messages, change the conditions under which the messages will be sent out, as well as select who will receive the messages.

The notification database stores the list of messages that are sent for a given event, the conditions under which each will be sent, and who should receive the message. Messages may vary for each recipient and differ in type.

The advantages of a centralized message system are many; consistency across modules, centralized notifications for easy update and customization, and significantly, the ability to customize notifications with RAD expressions without having to purchase a RAD license or having to touch the RAD layer of ServiceCenter.

Most message applications create output events and establish the connection via Event Services. Users can add or edit notifications for Format Control. There is no such restriction on message types. However, you must be able to call the application that is set in your definition.

The Notification File

The notification file works with the **message**, **msgtype**, and **On Call Schedules** files to define notifications for common system events, how those notifications should be delivered, and who they should be delivered to. Administrators can modify the notification arguments that trigger the notification, as well as define who receives the notification.

Opening the Notifications File:

- 1 Start a ServiceCenter Client and log in as an administrator.
- 2 Click the **Utilities** tab in the ServiceCenter main window.
- 3 Click the **Administration** button.
- 4 Click the **Notifications** tab. From here, you can access the areas that control the notification engine: select either **Notifications**, **Messages**, **Message Types**, **On Call Schedules**, or **Daily On Call Record**.
- 5 Click **Notifications**. The Notification Definition panel appears.
- 6 Enter the Notification Definition name in the Name text box and click **Search**. For a list of all notifications shipped with this version of ServiceCenter, see *Default Notification Definitions* on page 167. Alternately, perform a true search to see a list of all notification records on your particular system.

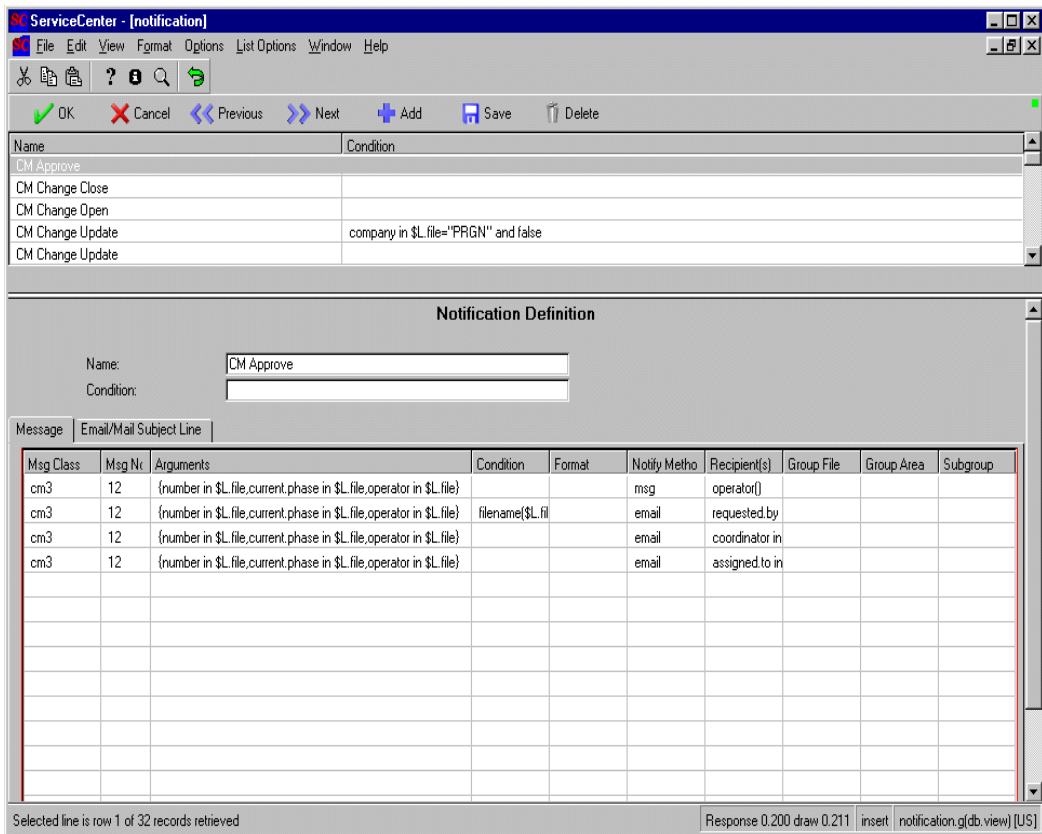


Figure 3-1: The Notification Definition Panel

The notifications shipped with ServiceCenter cover all system events. You can however add a condition to the notification that is unique, and which will be handled only when the condition is met. For example, you can add the condition

`company in $L.file="PRGN"`

to the CM Change Update notification record, add the new file to the system, resulting in multiple CM Change Update records that behave according to their conditions.

The first three fields on this panel, Msg Class, Msg No and Arguments, determine which message to bring up from the scmessage file.

Fields on the Notifications Definition Panel

Msg Class - The Msg Class relates to the application area. For example **cm3**, **pm**, **sm**, etc. The Msg Class does not have to match the application for which you are sending notifications, however. For instance, the Msg Class “us” does not relate to a specific application area, and contains general notifications.

Msg No - The Msg No or message number corresponds to the **scmessage** file. See *Messages* on page 154. The **Message Class**, **Message Number**, and **language** fields make up the unique key for this notification. If you add your own message to the **message** file, the combination of the **Msg Class**, **Msg No** and **Language** fields must not exist in the system already.

Arguments - The message arguments can range from none to many. The arguments correspond to the %S in the message text. If there is only one argument, enter the value directly. List multiple arguments in an array. For example (**<arg1>**,**<arg2>**, **<and so on>**). Elements of the array can be string literals or expressions. To reference a value in a record, enter: **fieldname in \$L.file**. Strings must be enclosed in double quotes

Condition - Enter the condition under which the message should be sent. Values can be True, False, or an expression that evaluates to either True or False. The default value is True.

Format - This field allows you to attach the contents of a record to a message. The format you specify here dictates which fields to include in the message. The record associated with the notification will be appended using the format indicated.

Notify Method - This field specifies the how the message is sent. Enter either a Message Type or Message Class in this field. Message types are available from the pull-down menu, while message classes are typed directly in the field. While using a **message type** is more direct, using a **message class** lets you send a notification in multiple ways; One event can trigger an on-screen notification, print to a file, write to a log and send an external email, if so defined.

If you specify a **message class** in the **Notify Method** field, then you must define an appropriate message class record.

For example, in the notification definition for **IM Action Open**, the **Notify Method** is a message class called **Problem Open**. Actions for message classes must be defined before they can be used. The message class associated with **Problem Open** is **On Screen**. Additional notification actions for the **Problem Open** message class must be defined before they can be used.

If you do not make a selection, the **Notify Method** will default to **msg**.

Recipients - Specify whom to send the message to. Enter an expression or string literal that references an individual user or group name.

The following fields are only necessary if the **Recipients** field contains a group name:

Group File - Enter the file that the group name is referencing. If you use this field with either **ocmgroups** or **cm3groups**, you can also specify members of those groups by using the **Group area** and **Sub group** fields. If you create an **On Call Group** from the **On Call Schedule** panel, say **PRGN Admin**, then the **Group Area** and **Sub Group** fields do not apply.

Group Area - Acceptable values for use with the **ocmgroups** file are: **All**, **Quotes**, **Line Items**, and **Orders**. Acceptable values for use with the **cm3groups** file are: **All**, **Changes**, and **Tasks**.

Subgroup - The subgroup field further specifies the user list. Values are **Members**, **Approvers** and **All**.

Email/Mail Subject Line tab

Email subject lines can be customized using the fields found on this tab. Think of the **Email/Mail Subject line** as a continuation of the message tab; the custom email subject must be on the same line as the notification on the message tab. In other words, if the message that you want to add a custom subject line for is found on the fourth line under the message tab, then you have to place the subject line information on the fourth line of the **Email/Mail Subject line** tab.

Msg Class - The message class relates to the application area. For example **cm3**, **pm**, **sm**, etc. The Message database lists all available messages. See *Messages* on page 154. The Msg Class does not have to match the application for which you are sending notifications, however. For instance, the Msg Class “us” does not relate to a specific application area, and contains general notifications. If left blank, the system uses the Msg Class defined on the message tab of the corresponding line.

Msg No. - The Msg No or message number corresponds to the **scmessage** file. See *Messages* on page 154.

Arguments - Add arguments to the subject line in this field. Certain messages require arguments, while others do not. You can determine which is the case when you select the Msg Class and Msg No.

Editing a Notification Record

Administrators can add to or change the fields in a Notification Record, including the recipients of a particular message, the conditions under which the message will be sent, and the arguments the message will use, by editing the record’s Notification Definition in the notification file.

To edit a Notification Record:

- 1 Open the notification file. See *Opening the Notifications File*: on page 144.
- 2 Enter the Notification Record’s Name and Condition in the search fields, if you know them, and click **Search**. The related Notification Records will appear. If you didn’t enter anything, all Notification Records will appear.
- 3 Select the desired Notification Record. The Notification Definition will appear at the bottom of the screen.

Note: The fields **Name** and **Condition** make up the unique key of a Notification Record.

- 4 Edit the **Message Class**, if desired. (cm3, pm etc.)
- 5 Edit the **Message Number**, if desired. (the message id)

The **Message Class**, **Message Number** and **Language** fields form the unique key that the program will use to determine which message in the **scmessage** file will be sent by this notification.

- 6 Enter or edit the **Arguments** the message should use, if desired.

Depending on the referenced message's text, as stored in the `scmessage` file, the number of arguments range from none to many. If there are no arguments in the referenced message, leave this line blank. If there is only one argument, enter that value directly. If there are several arguments, list them in an array, enclosed by braces and separated by commas (`{,}`). The format for referencing a value in the record is `fieldname in $L.file`. The format for string is just the string itself enclosed by double quotes.

- 7 Edit the **Conditions** under which this particular message should be sent, if desired.

The condition can be true or false, or any expression that evaluates to true or false.

- 8 Select the **Notify Method** from the drop down list, if desired.

If you do not make a selection, the Notify Method will default to `msg`.

- 9 Select the **Recipient(s)**: Enter the recipient of the message. This Recipient can reference more than one person if it is, for example, an email Group Name.

Enter only the **Recipient(s)** if it is one person, or both **Recipient(s)** and **Group File** if it's a Group Name.

Recipient(s), **Group File**, **Group Area**, and **Subgroup** work together to select the appropriate recipients from each group. **Recipient(s)** can be either a single person or a group name. If a group's name has been entered, the **Group File** specifies exactly which group it is.

Notifications can be sent to the people in a group who are on call, rather than to an entire group. To do this, select **oncall** as the **Group File** in the Notification Record's Notification definition. The group must be defined as an oncall group. See *On Call Scheduling* on page 157.

- 10 If the recipient is a Group Name, select the **Group File** that will receive the message from the drop down list.

The Group Files `cm3groups` and `ocmgroups` cannot be uniquely keyed by just the group name. For those two groups, select the **Group Area** from the drop down list and **Subgroup** from the drop down list.

For example, if you have `ASSET MANAGEMENT` as the **Recipient(s)**, `cm3groups` as the **Group File**, `All` as the **Group Area** and `Approvers` as the **Subgroup**, the message will go to the approvers of the Asset Management Change Management group.

- 11 Repeat step 4 through step 10 for each entry in the record you are changing.
- 12 Save the notification file

Adding new fields to a Notification Record

ServiceCenter can generate messages under your specified conditions by adding new rows to a record's Notification Definition in the notification file.

To add a new row to a record in the notification file:

- 1 Open the notification file. (See *Opening the Notifications File*: on page 144.) The Notification Definition window will appear.
- 2 Enter the Notification Record's Name and Condition in the search fields, if you know them, and click **Search**. The related Notification Records will appear. If you didn't enter anything, a list of all Notification Records will appear.
- 3 Select the desired Notification Record. The Notification Definition will appear at the bottom of the screen.
- 4 Make your entries in the next available row.

Note: The fields **Name** and **Condition** make up the unique key of a Notification Record. You can change the condition value, but it must not be identical to another condition with the same name.
- 5 Enter the **Message Class**.
- 6 Enter the **Message Number**.
- 7 Enter the **Arguments** the message should use, if necessary.

Depending on the referenced message's text, as stored in the `scmessage` file, the number of arguments range from none to many. If there are no arguments in the referenced message, leave this line blank. If there is only one argument, enter that value directly. If there are several arguments, list them in an array, enclosed by braces and separated by commas (`{,,}`). The format for referencing a value in the record is "`fieldname in $L.file`". The format for string is just the string itself enclosed by double quotes.
- 8 Enter the **Conditions** under which this particular message should be sent, if desired.

The condition can be true or false, or any expression that evaluates to true or false.
- 9 Select the **Notify Method** from the drop down list, if desired.

If you do not make a selection, the Notify Method will default to `msg`.
- 10 Select the **Recipient(s)**: Enter the recipient of the message. This Recipient can reference more than one person if it is, for example, an assignment Group Name.

Enter only the **Recipient(s)** if it is one person, or both **Recipient(s)** and **Group File** if it's a Group Name.

Recipient(s), **Group File**, **Group Area**, and **Subgroup** work together to select the appropriate recipients from each group. **Recipient(s)** can be either a single person or a group name. If a group name's been entered, the **Group File** specifies exactly which group it is.

- 11 If the recipient is a Group Name, select the **Group File** that will receive the message from the drop down list.

Note: The Group Files **cm3groups** and **ocmggroups** cannot be uniquely keyed by just the group name. For these two groups only, select the **Group Area** from the drop down list and **Subgroup** from the drop down list.

For example, if you have:

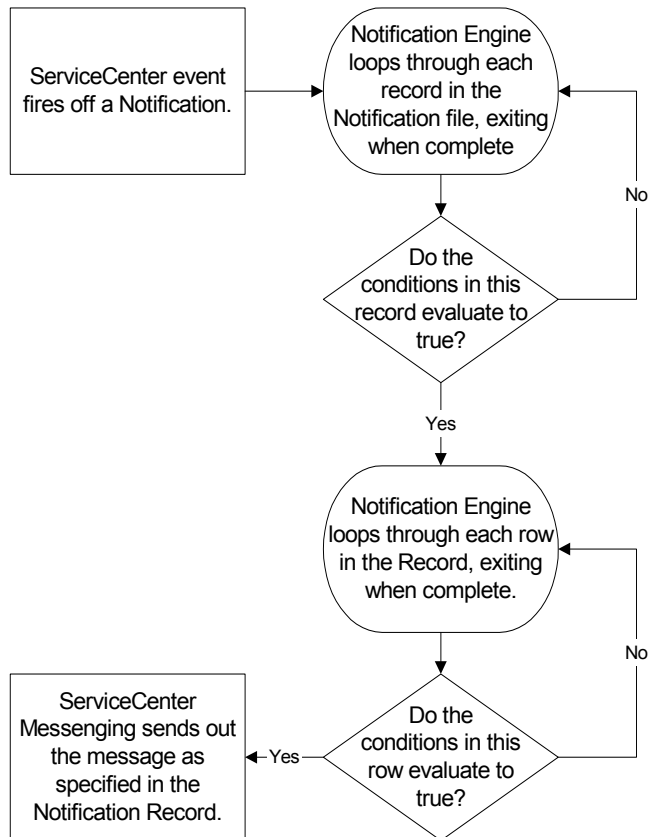
- Asset Management as the **Recipient**
- cmgroups as the **Group File**
- Approvers as the **Subgroup**
- Changes as the **Group Area**

the message will go to the members of the Asset Management Change Management group, who approve changes.

- 12 Repeat step 6 through step 11 for each entry in the record you are creating.
- 13 Save the notification file.

Notifications Flow Chart

Notification Engine



Calling Notification from Format Control

You can specify or call notifications for a format from Format Control. To do so simply requires that you call `us.notify` from the subroutine panel of the format you are working with.

There are three options for the parameters of `us.notify` under the Format Control subroutine:

Name - Enter the name of the notification record, for example CM Deny

Record - Enter the record variable, for example `$file`

Names - Enter an array of values to add to the message, if desired.

The screenshot shows the 'Format Control Maintenance - Subroutines' window. At the top, the title bar reads 'ServiceCenter - [Format Control cm3r.mac]'. Below the title bar is a menu bar with 'File', 'Edit', 'View', 'Format', 'Options', 'List Options', 'Window', and 'Help'. A toolbar contains icons for OK, Back, Previous, Next, Add, Save, and Delete. Below the toolbar are tabs for 'Views', 'Queries', 'Calculations', 'Validations', 'Subroutines', 'Add Options', and 'Privileges'. The main area is titled 'Format Control Maintenance - Subroutines' and shows 'Name: cm3r.mac' and 'View:'. Under 'Subroutines', there is a table with columns: Add, Upd, Del, Dis, Initial, Before, and Application. The table contains several rows, with the 'us.notify' row highlighted. The 'us.notify' row has 'true' in the 'Add' column, and its 'Application' column contains 'us.notify'. Below the table, there are several rows of input fields. The 'name' field is highlighted with a red border and contains 'CM Deny'. The 'record' field contains '\$file'. Other fields include 'false enc', 'false enc', 'not null(c', 'true', 'us.link', 'record', '\$file', 'prompt', 'fill', and 'us.notify'.

Figure 3-2: Calling Notifications from Format Control

See the Format Control section for more information on Format Control subroutines.

Messages

Administrators can edit the messages that are sent for ServiceCenter events, and add new customized messages by editing records in the `scmessage` record.

To view, edit or add a record in the message file:

- 1 Start a ServiceCenter Client and log in as an administrator.
- 2 Click the **Utilities** tab in the ServiceCenter main window.
- 3 Click the **Administration** button.
- 4 Click the **Notifications** tab.
- 5 Click the **Messages** button.
- 6 Open the appropriate `message` record.
 - a To search for a record, enter any identifying search information. (For example enter *us* or *cm3*, etc. in the **Class** field) and click **search**. The related records will appear.
 - b Click on a record to select it. The record's information will appear. The text of the message appears at the bottom of the entry in the box labeled **Text**. You can view multiple records by selecting **Record List** from the **ServiceCenter View** menu.

To see all `scmessage` records, perform a true search.

The message's **Language Code**, **Class**, and **Message Number** form the unique key that identify the message. Do not edit those unless you are creating a new message. If you create a new message, be sure to **Add** rather than **Save**.

- 7 Edit the message's **Severity**.
- 8 Edit the message's **Text**.
- 9 Add your **Comments**, if desired.
- 10 Click **Save** to save your changes, or click **Add** to add the record as a new message.

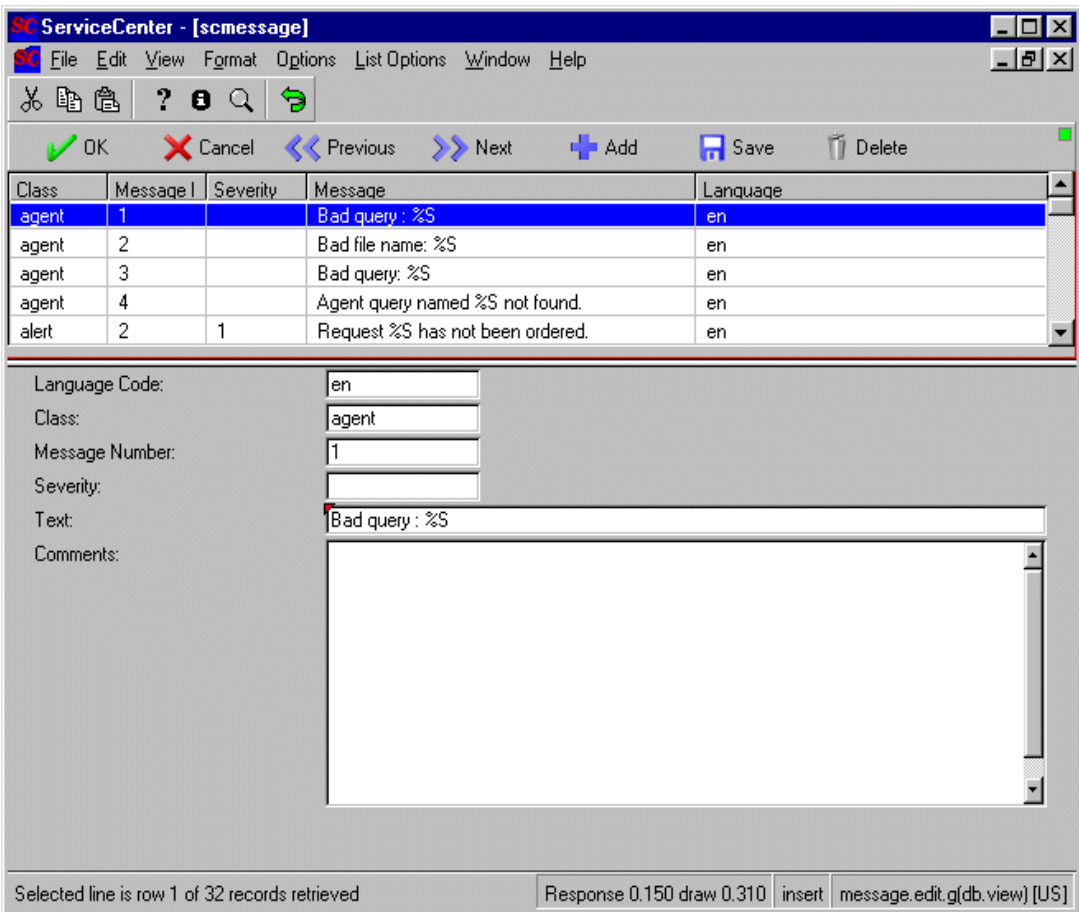


Figure 3-3: The scmessage panel

Fields on the message panel

Language Code - Defaults to the language in which ServiceCenter is currently running.

Class - The class refers to the ServiceCenter application to which the message pertains.

Message Number - The message ID.

Severity - Reserved for future use.

Text - Enter the message text in this field. Arguments are denoted as %S or %S[n] if there is more than one argument.

Comments - This field is optional, and is used to further describe the message.

Message Type

The message type defines how a message is sent (i.e. fax, page) and corresponds to the "notify method" in the notification definition. The **msgtype** file stores the RAD application to call when processing a message of a given type. There are currently eight predefined message types and message applications. Unlike notifications, additional message types can be created and incorporated into the engine by adding a record to the msgtype file and making sure the new type is available on the notification screen. The application can be taken from the canned system or created from scratch (by a system administrator in most cases).

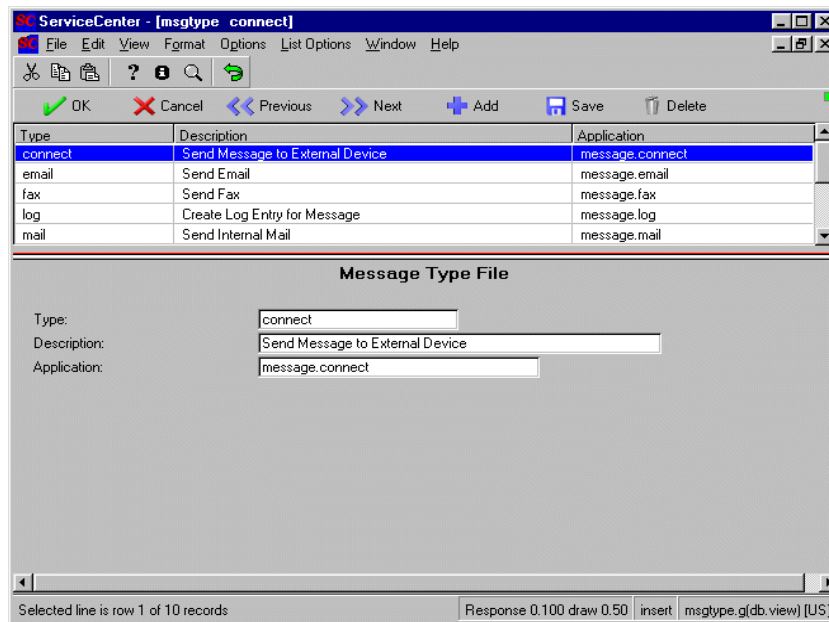


Figure 3-4: The Message Type Panel

Fields on the Message Type Panel

Type - Enter a unique name that can be called by the Notification engine.

Description - Enter a description of the Application and its function.

Application - Enter the application to call. This application must be defined in RAD code and exist on the system in order to work.

On Call Scheduling

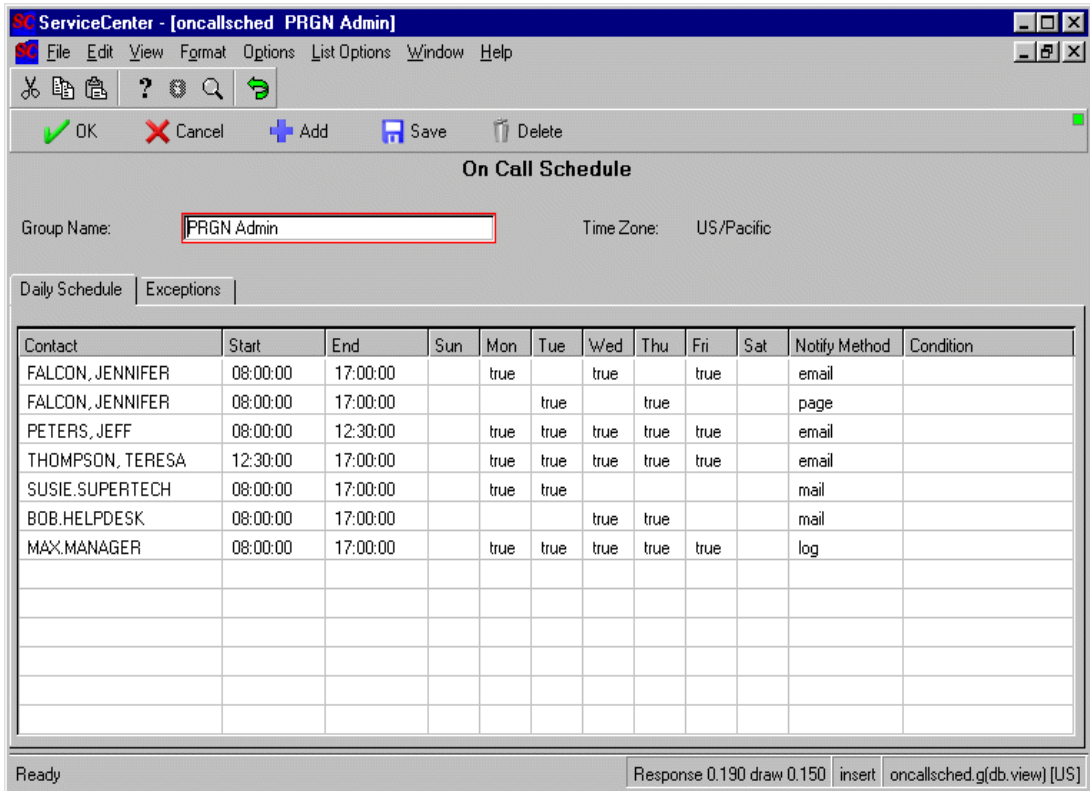
Notifications can be sent to the people in a group who happen to be on call, rather than to an entire group. To do this, select **oncall** as the group file in the notification record. For more information on editing Notification Records, see *Editing a Notification Record* on page 148.

In determining recipients, the Notification Engine refers to the **Group File**. If the **Group File** selected is *oncall*, then only the people scheduled as On Call for that day will receive the notification.

The members of the *oncall* group are redefined daily from the *oncallsched* file. The *oncall* **Group File** can be selected for use in a Notification Record. *oncall* record is created daily at 12 A.M. by default, using the info in the *oncallsched* database. *oncall* is not modifiable; changes in shifts and contacts must be made in the *oncallsched* file, which will then be reflected by the associated *oncall*.

The On Call Schedule for a day is created at midnight, and covers the next 25 hours, so that there is a one hour overlap. The On Call Schedule can be modified by an administrator to start at any time during the day, and run for 25 hours. Midnight is the default setting. The daily *oncall* **Group File** can be viewed, but not edited directly since it is auto generated. To edit the *oncall* schedule, click the On Call Schedules button found under the Notifications tab.

Fields on the OnCall Schedule Panel



Contact	Start	End	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Notify Method	Condition
FALCON, JENNIFER	08:00:00	17:00:00		true		true		true		email	
FALCON, JENNIFER	08:00:00	17:00:00			true		true			page	
PETERS, JEFF	08:00:00	12:30:00		true	true	true	true	true		email	
THOMPSON, TERESA	12:30:00	17:00:00		true	true	true	true	true		email	
SUSIE.SUPERTECH	08:00:00	17:00:00		true	true					mail	
BOB.HELPDESK	08:00:00	17:00:00				true	true			mail	
MAX.MANAGER	08:00:00	17:00:00		true	true	true	true	true		log	

Figure 3-5: The OnCall Schedule Panel

The Daily Schedule tab

Contact - Enter the name of the person receiving the notification. The individual can be either an operator or a contact.

Start - Enter the start hour, in 24 hour format. For example 08:00:00. The default is 00:00:00.

End - Enter the end hour, in 24 hour format. For example 17:00:00. The default is 23:59:59

Days of the week - Enter True if the contact is scheduled for that day, or false if not. The default is false.

Notify Method - Select the notification method from the drop-down list.

Condition - Enter either True or False, or an expression that evaluates to True or False, in order to process the record. The default is True.

The Exceptions tab

Contact - Enter the name of the person receiving the notification.

Start Date - Enter the start date, in standard ServiceCenter format.

End Date - Enter the end date, in standard ServiceCenter format.

Notify Method - Select the notification method from the drop-down list.

Condition - Enter either True or False, or an expression that evaluates to True or False, in order to process the record. The default is True.

Replace Daily - If this field is set to True, then the Exception fields will override the Daily Schedule fields, for the selected contact name.

If both Daily Schedule and Exceptions hours have been defined for a person, setting Replace Daily to false causes both sets of rules in the *oncall* record to be followed. If Replace Daily is set to true, only the exception hours will be listed.

Creating an oncall schedule

To create an *oncall* schedule:

- 1 From the System Administrators home menu, click Utilities tab > Administration > Notifications tab > On Call Schedule.
- 2 Enter the **Group Name** you want to determine the schedule for.
- 3 Select the Daily Schedule tab.
- 4 Enter the **Contact** name.
- 5 Enter the **Start** time.
- 6 Enter the **End** time.
- 7 Mark the days of the week as true or false. The default value is false.

- 8 Select the **Notify By** method from the drop down list.
- 9 Enter the **condition** that applies.
A valid entry is either true or false or any expression that evaluates to true or false.
- 10 Select the Exceptions tab.
- 11 Enter the **Contact** name.
- 12 Enter the **Start** date. You must enter at least the full date (mm/dd/yy).
If no time is entered, the time is defaulted to '00:00:00'.
- 13 Enter the **End** date. You must enter at least the full date (mm/dd/yy).
If no time is entered, the time is defaulted to '00:00:00'.
- 14 Select the **Notify By** method from the drop down list.
- 15 Enter the **Condition** that applies.
A valid entry is either true or false or any expressions that evaluate to true of false.
- 16 Select true or false from the **Replace Daily** drop down list.

Modifying an oncall schedule

When a contact's availability changes for a short period, the administrator can update the oncall schedule so that the changes will be taken into consideration by the notification engine. To update the oncall schedule, it will be necessary to make the schedule updates in the `oncallsched` file, and then run the oncall schedule.

Update a contact's oncall schedule:

- 1 From the System Administrators home menu, click Utilities tab > Administration > Notifications tab > On Call Schedule.
- 2 Enter the **Group Name** you want to determine the schedule for.

Run the oncall schedule after updating the schedule, to have updates included in the daily oncall file.

Update the schedule:

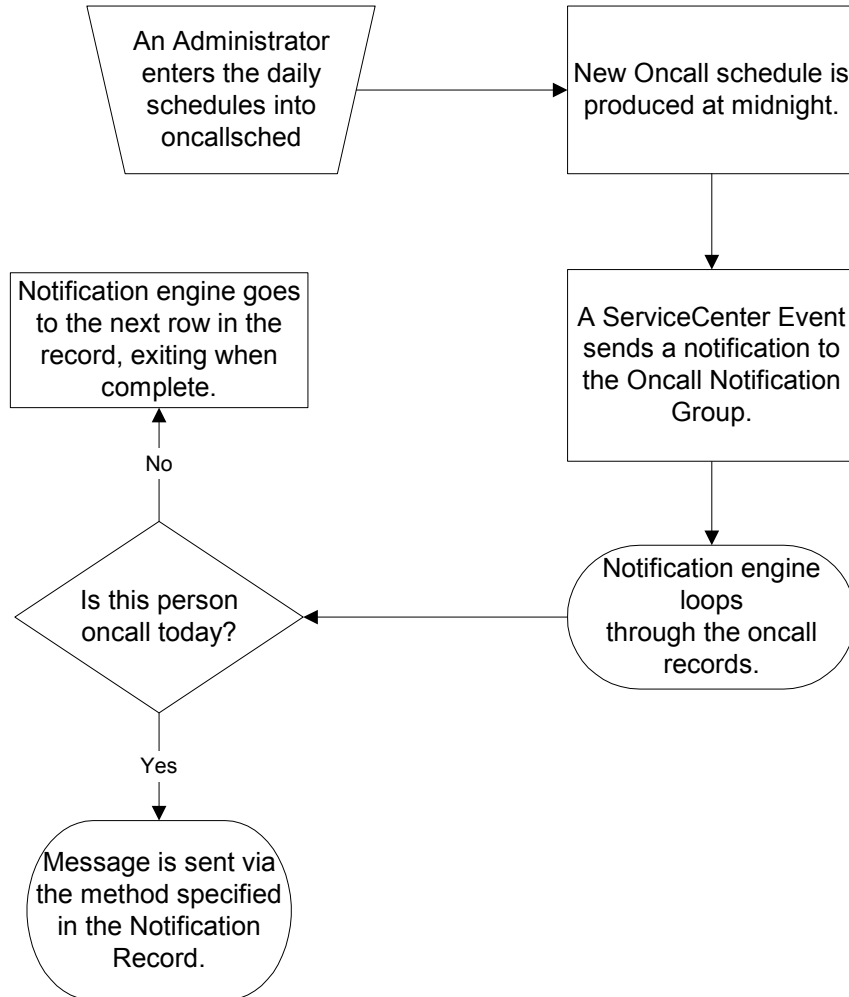
- 1 Make the appropriate changes to the `oncallsched` record.
- 2 Open the schedule file.
- 3 Select the Oncall Notification Processor record.

- 4 Change the Expiration to a time that has already passed.
- 5 Save the file and exit.

The next time incident scheduler runs, it will notice that Oncall Notification Processor expiration time has passed, and will cause the update to be run. The new expiration date will be set to the following midnight.

Oncall Flow

On Call Scheduling



Daily On Call Records

The Daily On Call Records panel displays an overview of the members of a group who are on call for a given day. The information refreshes based on problem scheduler.

Note: You cannot create an On Call Record from this panel.

To display an On Call Record for a given group:

- 1 Click **Utilities** on the system administrator's home menu.
- 2 Click the **Administration** button.
- 3 Click the **Notifications** tab
- 4 Click **Daily On Call Records**. A blank On Call Record displays.
- 5 Enter a group name and press enter, or just click search to bring up a list of all group names on your system. The system ships with one group defined, PRGN Admin. Selecting a group displays the schedule for the members of that group.

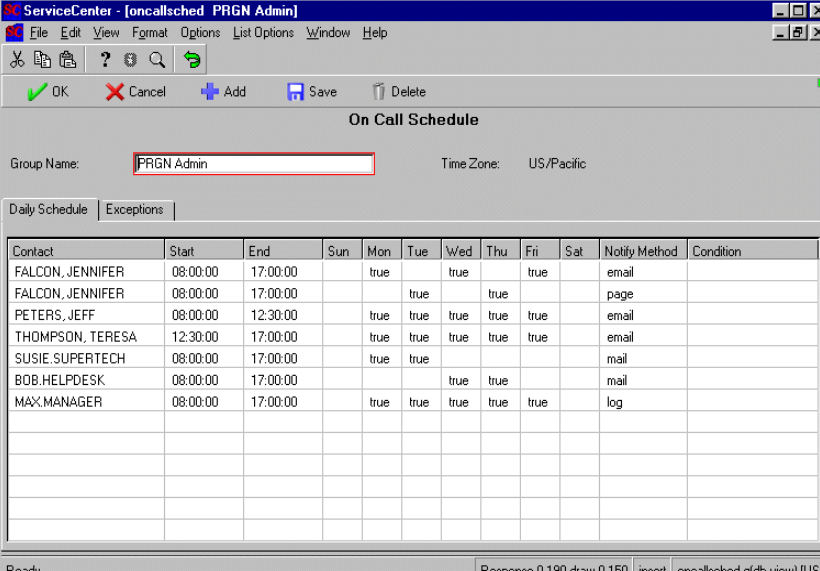
Contact	Start Time	End Time	Notify Method	Condition
FALCON, JENNIFER	12/13/01 08:00:00	12/13/01 17:00:00	page	
PETERS, JEFF	12/13/01 08:00:00	12/13/01 12:30:00	email	
THOMPSON, TERESA	12/13/01 12:30:00	12/13/01 17:00:00	email	
BOB.HELPPDESK	12/13/01 08:00:00	12/13/01 17:00:00	mail	
MAX.MANAGER	12/13/01 08:00:00	12/13/01 17:00:00	log	

Ready Response 0.130 draw 0.20 insert oncall.g[db.view] [US]

Figure 3-6: On Call Schedule

Example: On Call Schedule

The following panels ship with ServiceCenter. For example purposes, a group named PRGN Admin has been created. You can access this example and modify it for your own use by clicking search from a blank On Call Schedule.



ServiceCenter - [oncallsched PRGN Admin]

File Edit View Format Options List Options Window Help

OK Cancel Add Save Delete

On Call Schedule

Group Name: PRGN Admin Time Zone: US/Pacific

Daily Schedule Exceptions

Contact	Start	End	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Notify Method	Condition
FALCON, JENNIFER	08:00:00	17:00:00		true		true				email	
FALCON, JENNIFER	08:00:00	17:00:00			true		true			page	
PETERS, JEFF	08:00:00	12:30:00		true	true	true	true	true		email	
THOMPSON, TERESA	12:30:00	17:00:00		true	true	true	true	true		email	
SUSIE.SUPERTECH	08:00:00	17:00:00		true	true					mail	
BOB.HELPPDESK	08:00:00	17:00:00				true	true			mail	
MAX.MANAGER	08:00:00	17:00:00		true	true	true	true	true		log	

Ready Response 0.190 draw 0.150 insert oncallsched.g(db.view) [US]

Figure 3-7: Example On Call Schedule

The members of the group have had their regular, daily schedules defined, as per the field definitions for this panel. Note FALCON's schedule is irregular and that she has two definitions, one for Monday/Wednesday/ Friday, the other for Tuesday and Thursday. Note also the different notification methods for each of the two lines: one email, one page, based on the assumption that when FALCON is off site, she will not be able to access email.

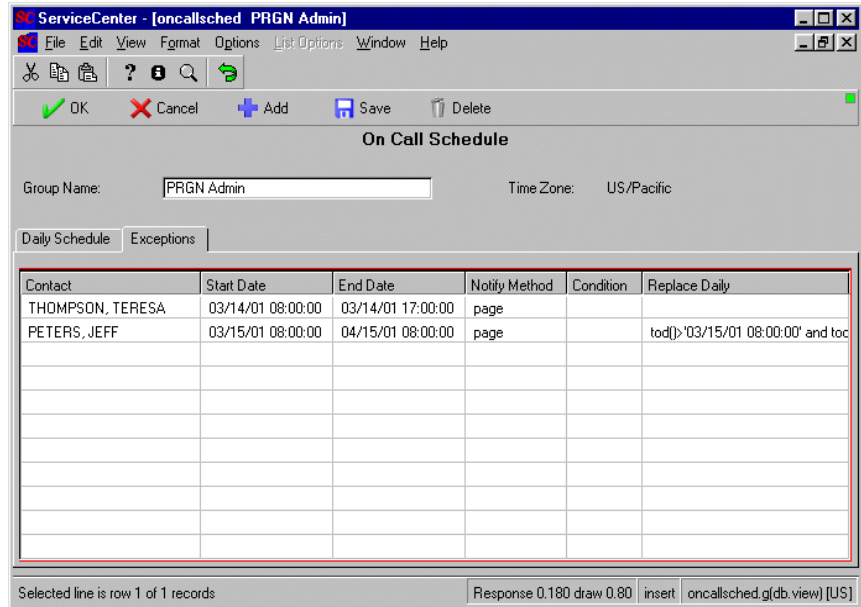


Figure 3-8: Example On Call Exceptions

Should members of your group be travelling, on vacation, or deviating from their defined On Call Schedule for whatever reason, then you must use the Exceptions tab to define the exceptions. Note the exception for Thompson and Peters that overrides their On Call schedule for the period specified.

ServiceCenter - [oncall PRGN Admin]

File Edit View Format Options List Options Window Help

Back

On Call Schedule

Group Name: Delete?

Contact	Start Time	End Time	Notify Method	Condition
FALCON, JENNIFER	12/13/01 08:00:00	12/13/01 17:00:00	page	
PETERS, JEFF	12/13/01 08:00:00	12/13/01 12:30:00	email	
THOMPSON, TERESA	12/13/01 12:30:00	12/13/01 17:00:00	email	
BOB.HELPDESK	12/13/01 08:00:00	12/13/01 17:00:00	mail	
MAX.MANAGER	12/13/01 08:00:00	12/13/01 17:00:00	log	

Ready Response 0.130 draw 0.20 insert oncall.g(db.view) [US]

Figure 3-9: Example Daily On Call Schedule

The Daily On Call Record panel takes the information entered in the On Call Schedule panels and displays the members of group, daily.

You cannot modify the information being displayed from the Daily On Call Record panel; this panel displays the current definitions only. You can, however, take the information and definitions from the PRGN Admin group and modify them to make them your own. Just be sure to save it out to a different group name.

The information on this panel refreshes daily, or whenever the incident scheduler is run.

Exceptions

Daily On Call Schedule covers a 24 hour period. When using the exceptions tab and entering dates that span across multiple days, such as from 4/02 to 4/03, the On Call schedule will reflect the exception for the upcoming 24 hours only. Therefore, if you schedule a Daily On Call exception that spans multiple days, the system will not reflect the full length of the exception. As the scheduler runs, exceptions will be updated and will reflect the time remaining on the exception.

For example, if MAX.MANAGER adds a 24 hour exception to his schedule, from 0800 4 April to 0800 5 April, the system will reflect the exception in two stages:

- stage one - the scheduler runs at 1 am and updates the Daily On Call exception for the next 24 hour period. Exceptions that fall outside that period or straddle the 24 hour cycle will not show the full length of the exception on the display panel.
- stage two - the scheduler runs the following day at 1 am and updates the exceptions panel for the next 24 hour period.

After the scheduler runs at 1 AM, the exception panel will reflect the last part of MAX.MANAGER's exception.

Default Notification Definitions

The following table lists the Notification Records and their basic function:

Notification Record	Action
CM Approve	sent when a user approves a change phase
CM Change Close	sent when a user closes a change phase
CM Change Open	sent when a user opens a change phase
CM Change Update	sent when a user updates a change phase or opens one that's been deferred
CM Deny	sent when a user disapproves a change
Final Approve	
CM Task Close	sent when a user closes a task phase
CM Task Open	sent when a user opens a task phase

Notification Record	Action
CM Task Update	sent when a user updates a task phase or opens one that's been deferred
CM Retract	sent when a user unapproves a change phase
CM Retract All	sent when a user unapproves all levels of a change phase
CM Retract One	sent when a user unapproves one level of a change phase
Clone Relation	sent when a user copies associations of a cloned incident or call
ICM Attribute Add	sent when a user adds an attribute
ICM Attribute Update	sent when a user updates an attribute
ICM Children Update	sent when a user updates a device's children
ICM Delete	sent when a user deletes a device type
ICM Device Add	sent when a user adds a device
ICM Device Update	sent when a user updates a device
IM Action Alert	sent when a user updates an incident's alert status
IM Action Close	sent when a user closes an incident
IM Action Open	sent when a user opens an incident
IM Action Reopen	sent when a user reopens an incident
IM Action Update	sent when a user updates an incident
IM Alert Reassign	sent when an incident exceeds the reassignment limit
IM Edit Close	sent when a user closes an incident
IM Edit Close Cascade	sent when a user closes an incident and its related calls
IM Edit Close Linked	sent when a user closes an incident but not its related calls
IM Edit Open	sent when a user opens an incident
IM Edit Reopen	sent when a user reopens an incident
IM Edit Resolve	sent when a user resolves an incident
IM Edit Update	sent when a user updates an incident
IM First Open	sent when a user opens an incident
IM First Close	sent when a user closes an incident
IM First Resolve	sent when a user resolves an incident

Notification Record	Action
IM Save Relation	sent when a user opens a related incident for a call and ends up linking the call to an existing incident instead
RM Approval	sent when any approval action is taken against the request or order
RM Denial	sent when any denial action is taken against the request or order
RM Final Approval	sent when the request is completely approved
RM Final Denial	sent when the request is completely denied
RM Line Item Change Category	sent when a user manually changes the category of a line item
RM Line Item Close	sent when a user manually closes a line item
RM Line Item Drop Avail	sent when a user marks a line item as unavailable
RM Line Item Mark Avail	sent when a user marks a line item as available to order
RM Line Item Open	sent when a user opens a new line item
RM Line Item Reopen	sent when a line item is reopened
RM Line Item Update	sent when a line item is updated
RM Order Change Category	sent when a user manually changes the category of the order
RM Order Close	sent when an order is closed
RM Order Open	sent when an order is opened
RM Order Update	sent when an order is updated
RM Request Change Category	sent when a user manually changes the request category
RM Request Close	sent when an order is closed
RM Request Deferred	sent when a request is deferred
RM Request Open	sent when a request is opened
RM Request Open Next Phase	sent when the quote moves from one phase to the next
RM Request Phase Change	sent when a user manually changes the phase of the request
RM Request Reopen	sent when a request is reopened
RM Request Update	sent when a request is updated

Notification Record	Action
Request Late Notice	sent after a specified due date
Request Not Approved	sent when a request is denied or not approved
Request Not Ordered	sent if the request is not in the ordering phase within one day of the date that the first line item is ordered.
SM Add	sent when a call is opened in an active state
SM Close	sent when a call is closed
SM Save	sent when a call is opened in an inactive state
SM Update	sent when a call is updated
Save Association	sent when linking two associated records
Save Relation	sent when opening a related record
due date notice	sends the message that corresponds to the due date notice alert
late notice	sent when an item reaches due date
Password.changed	sent when a user successfully changes their password.

4

Global Lists / Global Initer

CHAPTER

There are two tools that work together to generate lists for the ServiceCenter system: The Global Initer and Global lists.

Global lists are stored in the system and are available to any application. For instance, you may wish to add a combo box to a form that displays a list of values, operators for instance, based on a table in your database. For example, set your combo box **DisplayList** value to **\$G.time.zones**, and the system displays a drop-down list of time zones from the *tzfile* file.

The Global Initer is an efficient method of building lists of records from the database. Generally, these lists are used to fill the drop-down lists in combo boxes on display forms. Examples of common lists include a list of all operators in the system, all Incident categories in the system, or all assignment groups on file.

The Global Initer consists of two parts; a server side element which generates and packages these lists according to a user-defined schedule, and a client side requester which queries lists from the server and places them into client side memory.

Starting the Server Side Component of the Global Initer:

The server side component of the Global Initer consists of a routine called `apm.server.initer`. A background scheduler called `lister` has been defined to run `apm.server.initer`. To start `lister`, if `lister` has not started with the system, go to the system status utility, choose start background scheduler, and select `lister`. Then, every sixty seconds, `lister` will check and refresh any global lists that have passed their reset time.

Moving a Global List to a client:

To move a global list into client side memory, you call a routine called `apm.global.initer`. This routine accepts two parameters, an array containing the name of the lists that you want, and a keyword, either `create` or `refresh`. When `apm.global.initer` is invoked with the `refresh` parameter, it will collect the newest version of its list from the server whether or not that list already exists in the client's memory space. If invoked with the `create` parameter, it will only collect a list from the server if a copy does not already exist.

If you are programming in RAD, you can invoke `apm.global.initer` directly from your code. If, as is more common, you are using the Global Initer to fill combo boxes for display on a form, you can invoke the Global Initer as a subroutine using Format Control for the format in question.

Global Initer Considerations:

- A global list cannot contain more than 2800 individual elements.
- The total size of the `globallist` record, which contains the full lists, cannot exceed the logical file limit set for the system.
- The sort field must be a key on the table which is used to generate the list.
- No matter what you set as the reset time, the server initer can only refresh the lists once every time its scheduler wakes up. Thus, if you have a list with reset time = 5 seconds and the scheduler runs every 60 seconds then the list will actually refresh every 60 seconds.

Lister

In order to build a dynamic list, you must first generate a *list definition block*. The background scheduler, (lister) wakes up every sixty seconds to search for obsolete lists and uses list definition blocks to generate new lists, which are then stored in the list repository.

Once lister has built a list on the server, multiple clients can request and *share* the same list without each client having to build its own copy. This method of building lists improves system speed and helps server-side performance.

Note: Lister variables can be defined for *scalar* type fields only and not for *array* fields.

List records

To access a list record:

- 1 Select the **Utilities** tab in the System Administrator's Home Menu.
- 2 Click **Tools**.
- 3 Click Global Lists. A blank global list form is displayed.
- 4 Enter data and click **Add** to create a new list, or Search for one included in the basic system.

The following examples use the `time.zones` list, which builds a list of all time zones recognized by the system.

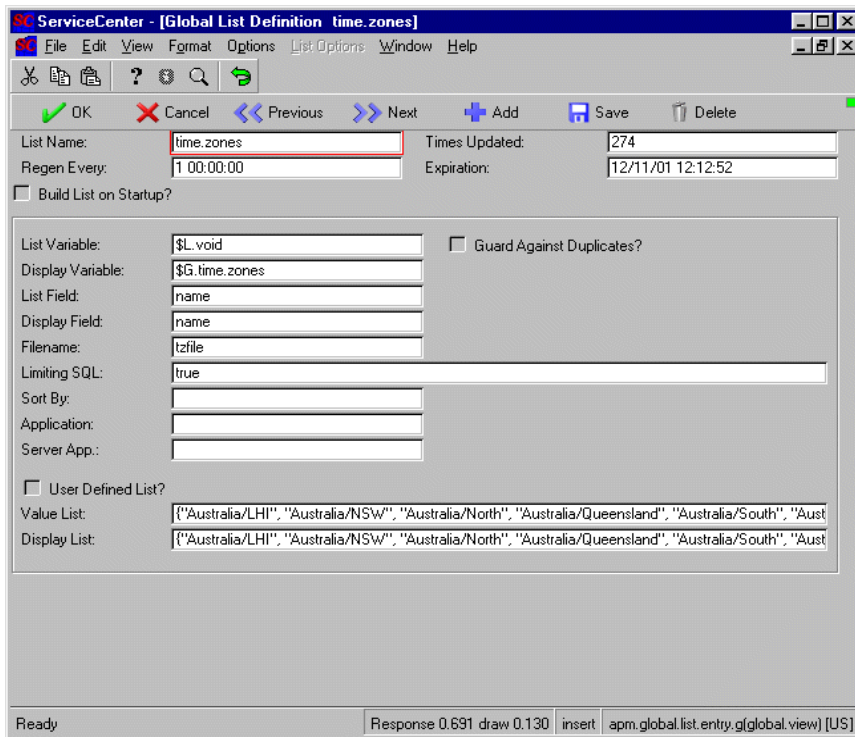


Figure 4-1: Time Zones Global List Record

When the `time.zones` list is generated on the server, the system:

- Queries all records in the system from the `tzfile` file which match a `true` query.
- Builds a display list by adding the `name` field from each returned record to an array.
- Builds a value list by adding the `name` field from each returned record to a different array.
- Saves both lists.
- When a client requests this list by name, the following occurs:
 - The client creates a variable called `$G.time.zones` in which to store the contents of the server's `DisplayList`.
 - The client creates a variable called `$L.void` in which to store the contents of the server's `ValueList`.

Field Definitions

List Name — The unique name of this list. Choose list names which relate to their contents, e.g., active operators if a list is to include all operators currently active on the system.

Regen Every — How often should the list be rebuilt? If the data in the table is highly volatile, pick a quick regen time. If the data rarely changes, pick a larger regen interval so the system does not waste resources regenerating unnecessary lists.

Times Updated — The number of times this list has been regenerated.

Expiration — The next point in time at which the list is to be regenerated. This value defaults to the current date and time.

Build List on Startup? — Click the check box if you want this list to be rebuilt at login.

List Variable — The name of the variable in which to store the ValueList value when this list is brought to your client.

Display Variable — The name of the variable in which to store the DisplayList value when this list is brought to your client.

List Field — The name of the field in the target file to be stored in the ValueList.

Display Field — The name of the field in the target file to be stored in the DisplayList.

Note: It is possible for your DisplayList and ValueList properties to be the same. For example, to store precisely the same thing the user sees in the combo box, enter the same values for each list.

Filename — The name of the file from which this list is to be built.

Limiting SQL — A valid query which determines which records in the target file will be included in the list.

Advanced users may take advantage of the flexibility of global variables (variables that begins with \$G.) so the lists can be shared between multiple threads.

Advanced fields

Guard Against Duplicates? — If this evaluates to *true*, the system will check to ensure that every element in the DisplayList and ValueList is unique. Even if the file from which you query has ten identical records, only one value will be stored in the DisplayList and ValueList properties.

Setting this field to *true* is unnecessary and slows the system slightly if your DisplayList and ValueList values are guaranteed to be unique in any event (e.g., they were unique keys on the table in question).

Sort By — This will force the system to sort by a particular key when building a list. By default, the lists will be sorted in primary key order.

Application — Enter the name of a RAD application to be called when a client requests a copy of the list. This application will build the list from the client side. This feature allows you to bypass the processing order of the lister.

Server App. — Enter the name of a RAD application the server side should use to build its lists rather than using the default processing.

User Defined List? — If this field evaluates to *true*, the system will use the literal values stored in ValueList and DisplayList to create a list rather than generate a list through table lookup.

Value List — This field contains a fixed list of values to be used as the Value List. Data must appear in a syntactically correct format (e.g., comma delimited and enclosed by braces, {"alpha","beta","kappa"}).

Display List — This field contains a fixed list of values to be used as the Display List. Data must appear in a syntactically correct format (e.g. comma delimited and enclosed by braces, {1,2,3} and *not* 1,2,3).

Global Lists

Certain files inside ServiceCenter have been tagged as containing global list data. Whenever relevant adds, updates, or deletes are performed against these files, ServiceCenter automatically *expires* these lists. Once a list has expired, it is regenerated the next time lister wakes up. Consequently, these *tagged* files can be associated with global lists that have long regen intervals and still maintain current data. This is possible because lister is notified that it needs to regenerate a list whenever an associated file is updated.

Tips for working with global lists:

- Do not decrease the regen intervals for any tagged lists. This will slow down your system.
- If you build a new list based upon a tagged file, you can pick a long regen interval and still be confident in the accuracy of your list.

To see a complete list of all the Global lists on your system:

- 1 Click the **Utilities** tab.
- 2 Click the **Tools** button.
- 3 Click the **Global Lists** button.
- 4 Open the Global lists definition form.
- 5 Click **Search**. A list of all Global lists on your system appears.

Binding a List to a Form

Lists are bound to the forms they serve with global variables applied in Forms Designer.

To bind your list to a form:

- 1 Select the **Toolkit** tab in the System Administrator's Home Menu.
- 2 Click **Forms Designer**.
- 3 Enter the name of the form you wish to modify in the **Form** field of the Forms Designer dialog box.

For example, enter `contact.detail.subform`. This is the name of the *subformat* used by `problem.template.open.g`, `problem.template.update.g`, and `problem.template.close.g` to display contact information.

- 4 Press **Enter** or click **Search**.
- 5 Click **Design** to edit the displayed form.
- 6 Add a combo box with label called **Time Zones** to the form, using the following values:

Field	Value
Input	time.zones
ValueList	\$G.time.zones
DisplayList	\$G.time.zones

- 7 Wait for lister to refresh the **time.zones** Global List, or go to the list definition panel (**Utilities > Tools > Global Lists**) and reset the refresh time.
Note: You may choose to leave the DisplayList value blank. If you do so, the contents of the ValueList will be used for both lists.

The completed form is shown in Figure 4-2 on page 179.

Returning a List to Your Client

Return a particular list, or set of lists, to a ServiceCenter client using either of the following methods:

- Building lists on startup
- Configuring with Format Control

Building lists on startup

This method moves the list to every client machine, whether the user requires it or not. The globallists record is set to build on startup using the following procedure.

To build lists on startup:

- 1 Select the **Toolkit** tab in the System Administrator's Home Menu.
- 2 Click **Database Manager**.
- 3 Type globallists in the **File** field of the Database Manager dialog box, and click **Search**.

- 4 Type the name of the list you want to view, time.zones in this example, and click **Search**. The time.zones file is displayed.
- 5 Click the Build List on Startup? check box.
- 6 Click Save.
- 7 Open an incident ticket to view the new field.

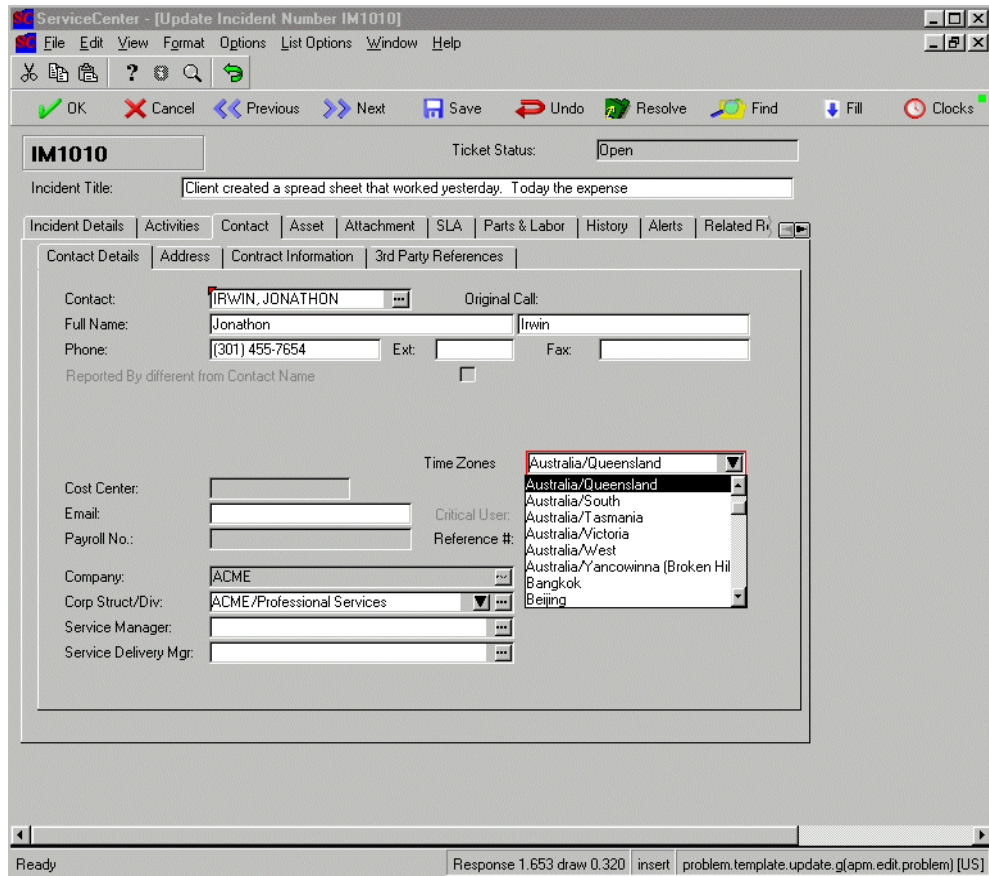


Figure 4-2: New Field in problem.template.update subform

Configuring with Format Control

Use Format Control to ask for a list only when a specific form is displayed. This procedure is more efficient, particularly if the form defined is not displayed frequently.

To configure a form:

- 1 Determine the *base* name of the form to which you added your list.
- 2 Select the **Utilities** tab in the System Administrator's Home Menu.
- 3 Click **Tools**. The Tools menu will open.
- 4 Click **Format Control**. A blank format control record will open.
- 5 Enter the base name of the form in the **Name** field in the blank Format Control Maintenance form. For this example, create a new record for the form **problem.template.update**.

Important: If you add a combo box to a *subformat*, attach the Format Control record to any *base* form using that subformat and *not* to the subformat itself.

- 6 Press **Enter** or click **Search**.
- 7 If a Format Control record does not exist for your form, create one.
- 8 Select **Subroutines** from the Options menu.
- 9 Select Show Expanded Form from the Options menu of the Subroutine form.
- 10 Scroll to the first empty section and set up a Subroutine call using the following values:

Parameter	Value
Application name	apm.global.initer
Names/Values	See next table
Before	true
Display	true

Names	Value(s)	Definition
names	{"time.zones"}	Array of lists to be moved
name	create	Action to perform (create lists)

- 11 Click Add to add a new record, or click Save to save your changes.

Note: This Format Control record returns the **time.zones** list to all incident tickets displaying the **Contact Detail** tab, regardless of mode.

Calling the `apm.global.initer` several times from Format Control will not slow the system appreciably. The system automatically keeps track of what lists are *already* on the client and will not recover them unnecessarily; however, if you want the *Global Initer* to reassemble the list regardless, invoke it with the **refresh** parameter (rather than the **create** value listed in the above). When called with the **refresh** parameter, the Global Initer returns all the lists passed to the **names** array, even if they are already in client memory.

Troubleshooting Lister

To troubleshoot lister, you must:

- Verify lister status and configuration.
- Regenerate obsolete lists.
- Rebuild *every* list in the system.

Note: This procedure assumes you have properly modified and linked your forms.

Verify Lister Status and Configuration

Verify the status of the process in the system and whether or not you have configured lister properly.

Is lister running?

To determine if lister is running:

- 1 Click the **System Status** button in the System Administrator's Home Menu.
- 2 Check to see if **lister** appears in the list of processes running on your system.

The screenshot shows the ServiceCenter System Status screen. The window title is "ServiceCenter - [** select option **]". The menu bar includes File, Edit, View, Format, Options, List Options, Window, and Help. The toolbar contains icons for Back, Refresh, Start Scheduler, Broadcast, Show Locks, Display Options, System Monitor, Command List, Summary, and Execute Commands. The main area displays "TOTAL USERS: 7 - use Refresh Display to refresh statistics". Below this is a table of running processes. The 'lister' process is circled in red.

Commr	User Name	PID	Device ID	Login Time	Idle Tim
	CLIENT-12670	832	SYSTEM	12/07/01 12:10:32	00:34:0
	CLIENT-12680	2008	SYSTEM	12/07/01 12:10:33	00:54:0
	spool	1744	SYSTEM	12/07/01 12:10:36	00:02:0
	report	988	SYSTEM	12/07/01 12:10:38	00:00:3
	problem	944	SYSTEM	12/07/01 12:10:39	00:00:5
	change	916	SYSTEM	12/07/01 12:10:40	00:00:2
	sla	904	SYSTEM	12/07/01 12:10:41	00:00:4
	agent	1696	SYSTEM	12/07/01 12:10:42	00:00:1
	marquee	1888	SYSTEM	12/07/01 12:10:44	00:00:0
	lister	932	SYSTEM	12/07/01 12:10:45	00:00:0
	linker	928	SYSTEM	12/07/01 12:10:46	00:00:3
	event	1908	SYSTEM	12/07/01 12:10:47	00:00:5
	availability	2116	SYSTEM	12/07/01 12:10:49	00:00:2
	contract	888	SYSTEM	12/07/01 12:10:51	00:00:0
	ocm	740	SYSTEM	12/07/01 12:10:52	00:00:2
	alert	2264	SYSTEM	12/07/01 12:10:53	00:00:3
	falcon	592	Java	12/07/01 12:12:51	3 22:13
	FALCON	2048	Java	12/07/01 12:14:50	3 22:11
	FALCON	804	Express-Windows NT	12/07/01 14:31:28	3 19:18

The status bar at the bottom shows "Ready" and "Response 0.300 draw 0.50 insert system.status.list.g [US]"

Figure 4-3: System Status Screen

- 3 If Lister is not on the list, click **Start Scheduler**.

Note: The option to start a scheduler is only available from an express login.

- 4 Double-click on `lister.startup` in the Startup Record QBE list to start the program.

The screenshot shows a window titled "Select startup record" with a toolbar and a table of records. The table has two columns: "Name" and "Description". The record "lister.startup" is highlighted in blue.

Name	Description
SLA	SLA background agent
agent	query/chart agent
alert.startup	IM 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 11 of 23 records insert info.qbe.g [US]

Figure 4-4: Startup Record QBE List

Does a scheduler record exit?

To find and open a scheduler record:

- 1 Select the **Toolkit** tab in the System Administrator's Home Menu.
- 2 Click **Database Manager**.
- 3 Enter `schedule` in the **Form** field of the Database Manager dialog box.
- 4 Press **Enter** or click **Search**.
- 5 Open `schedule` from the QBE list (double-click the record or click **Enter**). A blank Schedule File record is displayed.
- 6 Enter `List Runner` in the Name Text Box and click **Search**. If a record is found, it will open for viewing and editing.

Is the regen cycle realistic?

To determine the regen cycle:

- 1 Select the **Toolkit** tab in the System Administrator's Home Menu.
- 2 Click **Database Manager**.
- 3 Enter `apm.global.list.entry` in the **Form** field of the Database Manager dialog box.
- 4 Click **Search**. A blank record from the `globallists` file is displayed.
- 5 Click **Search**.
- 6 Select the list you want to regenerate from the Record list.

The screenshot shows the ServiceCenter Database Manager dialog box for the 'Calendars' record. The 'Regen Every' field is highlighted with a red circle. The dialog box contains the following information:

List Name	File Name	Expiration	Build on Startup
Aristocratic Titles	saphraristocratictitle	12/11/01 12:12:45	false
Assignment Groups	assignment	12/11/01 12:12:46	true
Availability Maps	availabilitymap	12/11/01 12:12:46	true
Calendars	calduityhours	12/11/01 12:12:46	true

Below the table, the following fields are visible:

- List Name: Calendars
- Times Updated: 278
- Regen Every: 1 00:00:00 (highlighted with a red circle)
- Expiration: 12/11/01 12:12:46
- Build List on Startup?
- List Variable: \$L.void
- Display Variable: \$G.calendars
- List Field: name
- Display Field: name
- Filename: calduityhours
- Limiting SQL: true
- Sort By: name
- Application:
- Server App.:
- Guard Against Duplicates?
- User Defined List?
- Value List: {"Long", "Short", "day shift", "day shift 2", "graveyard shift", "managers", "ops graveyard", "second sh
- Display List: {"Long", "Short", "day shift", "day shift 2", "graveyard shift", "managers", "ops graveyard", "second sh

At the bottom, the status bar shows: Selected line is row 4 of 32 records retrieved. Response 0.130 draw 0.110 insert apm.global.list.entry.g[db.view] [US].

Figure 4-5: Global List Record — Regen Field

- 7 Check the value in the **Regen Every** field.

Use a short time interval for lists that change frequently and a longer interval for lists that change only occasionally.

Has your list been regenerated recently?

To determine when a list was last regenerated:

- 1 Select the **Toolkit** tab in the System Administrator's Home Menu.
- 2 Click **Database Manager**.
- 3 Enter `apm.global.list.entry` in the **Form** field of the Database Manager dialog box.
- 4 Click **Search**. A blank record from the `globallists` file is displayed.
- 5 Click **Search**.
- 6 Select the list you want to regenerate from the Record list displayed.
- 7 Check the value in the **Expiration** field to see when the list was last regenerated.

Have recent changes been added to the list?

To view recent changes:

- 1 Select the **Toolkit** tab in the System Administrator's Home Menu.
- 2 Click **Database Manager**.
- 3 Enter `globallists` in the **File** field of the Database Manager dialog box. Click **Search**. A blank record from the `globallists` file is displayed.
- 4 Click **Search**.
- 5 Select the list you want to view from the Record list.
- 6 Check that the changes appear.

Regenerate Obsolete Lists

If you have performed all the steps above and the system still does not display the most current version of your list, regenerate all the obsolete lists in the system. For a large system with numerous obsolete lists, this procedure may take a few minutes.

You can change the expiration date of a list to a date in the past on prior to regenerating, in order to force regeneration.

Note: The expiration date of your list has not passed, the list will not be flagged as obsolete, and therefore not regenerated.

To regenerate a list:

- 1 Enter `*aapm.server.initer` in a command line.
Running this application will force the system to regenerate all obsolete lists.
- 2 Log out of ServiceCenter and log in again.

Regenerate All Lists

If none of the steps to this point have succeeded in regenerating your list, you should regenerate *all* the lists in the system. This procedure makes all the lists in the system obsolete and allows you to update the entire file.

To regenerate all lists:

- 1 Select the **Toolkit tab** in the System Administrator's Home Menu.
- 2 Click **Database Manager**. The Database Manager dialog box will open.
- 3 Mark the Administrative Mode check box.
- 4 Enter `apm.global.list.entry` in the **Form** field, and click search.
- 5 A blank record from the `globallists` file is displayed.
- 6 Click **search** to display a list of lists.
- 7 Select **mass update** from the List Options menu. A blank update screen is displayed. This form is identical in appearance to the lister record, but contains different option buttons.

8 Set the date in the **Expiration** field to any date in the past (e.g., 01/01/90).

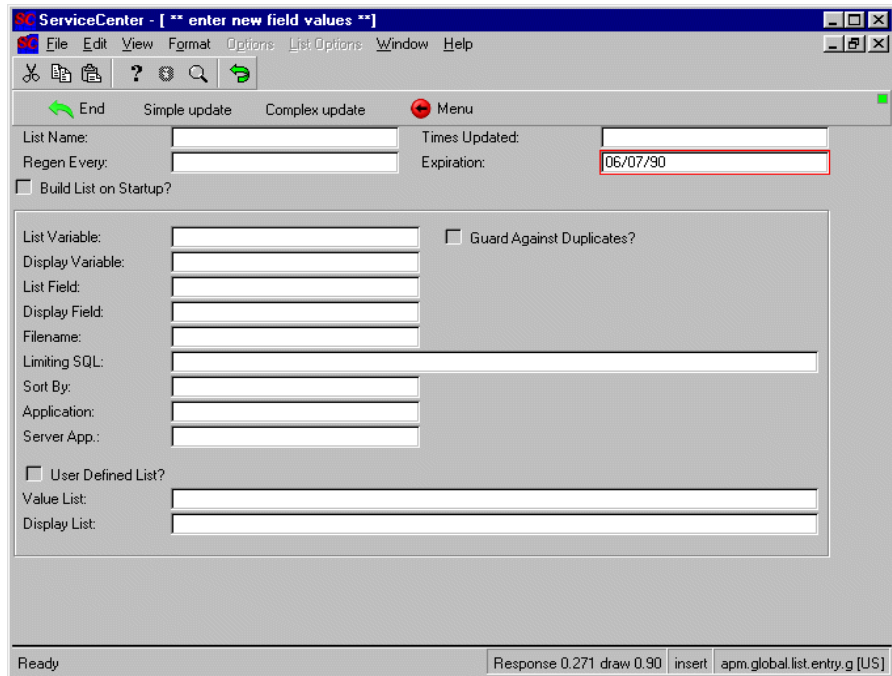


Figure 4-6: Mass Update of All Lists

9 Click Simple update.

The expiration date of all the lists in the globallists file is reset.

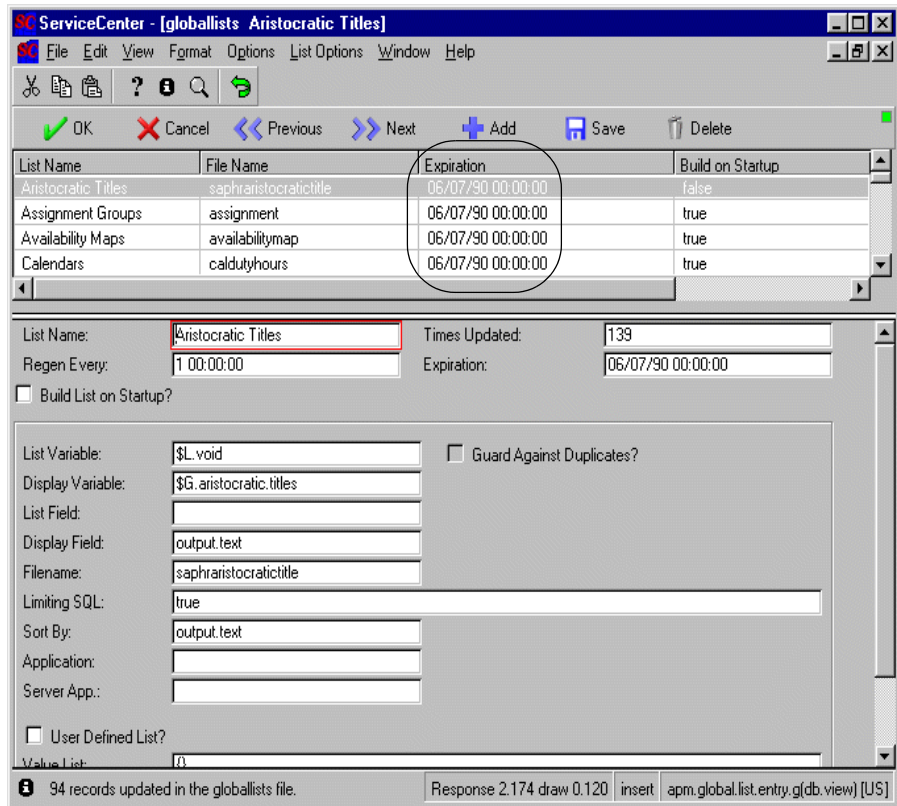


Figure 4-7: Updated Lists

10 Return to the home menu.

- 11 Enter `*aapm.server.initer` on the command line.

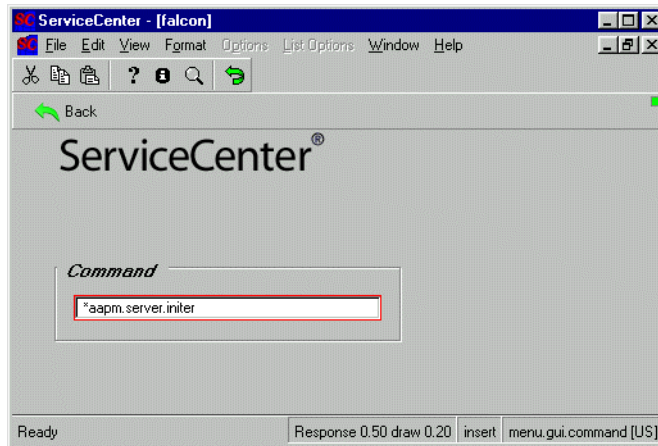


Figure 4-8: enter `*aapm.server.initer` on the command line

- 12 Press Enter.
- 13 Log out of ServiceCenter and re-login.

All the lists in the system are regenerated whether or not they are actually obsolete.

5 Online Help

CHAPTER

The Help Utility provides access to online help for all users of ServiceCenter. Peregrine Systems has provided help for some of the applications. Because of ServiceCenter's considerable flexibility and the desire of many customers to tailor its features for their own needs, the help contained in the standard system is intended as a starting point only. This document will assist you to create online help specific to your site requirements or language needs.

All users of ServiceCenter can access the online help you create for your system. Only users with the proper system capabilities can create, update or delete help records.

The topics covered in this chapter include:

- *Viewing Online Help* on page 192—Procedures for accessing online help and help record field definitions
- *Accessing the help File* on page 198—Procedures used to access the help file
- *Types of Help* on page 201—Discussion of the different types of help that can be created: file, field, format, and term
- *Creating Online Help* on page 206—Steps for creating online help
- *Related Information* on page 208—Keyword searches for related information

Viewing Online Help

Online help can be accessed from the Help menu in the menu bar at the top of the screen or by clicking the question mark button in the toolbar.

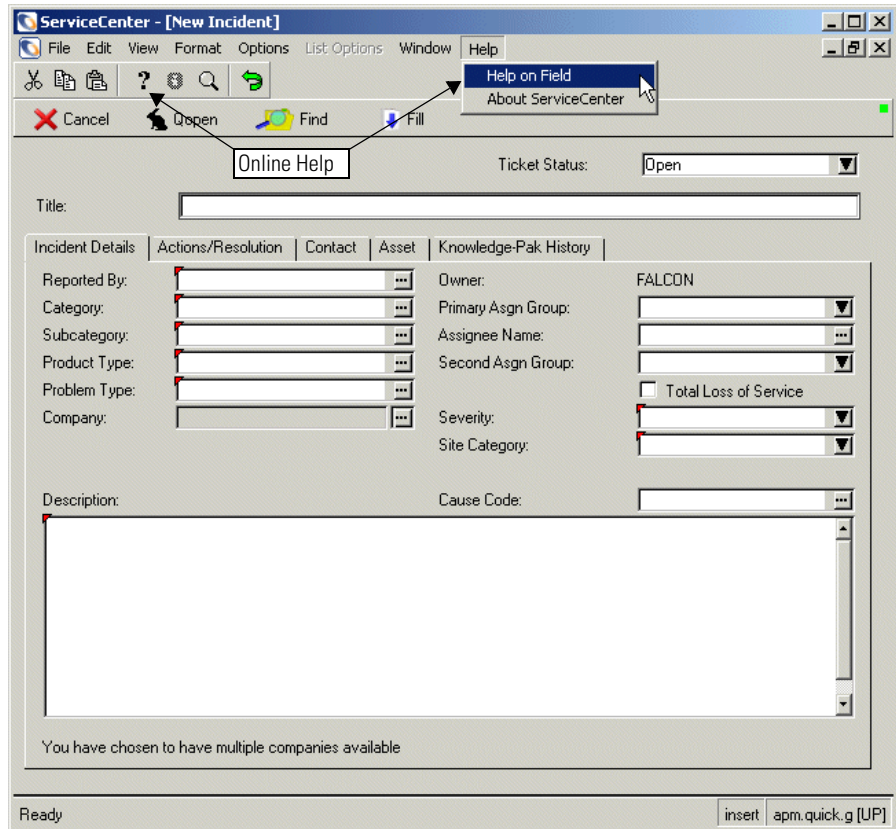


Figure 5-1: Online help access points

The forms that appear when you access online help depend upon your system capabilities. Users with **SysAdmin** or **help** capability words in their operator records are presented with editable help records, which they can use to add, update, or delete online help.

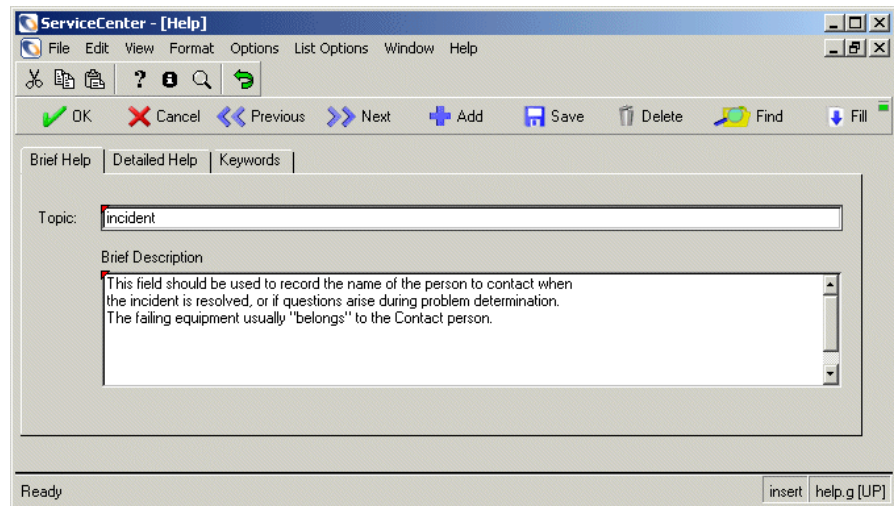


Figure 5-2: Editable help form for system administrators

Help Record Fields

Brief Help Tab

Field	Description
Topic	An optional description of the subject matter of the help entry. The system can search on the words in this field to assemble lists of appropriate help topics.
Brief Description	A brief, concise description of the field, menu, or form.

Detailed Help Tab

Use the Detailed Help tab to offer the user more detail about the field. In this example, the detailed description of the **category** field discusses such things as data type and key structure.

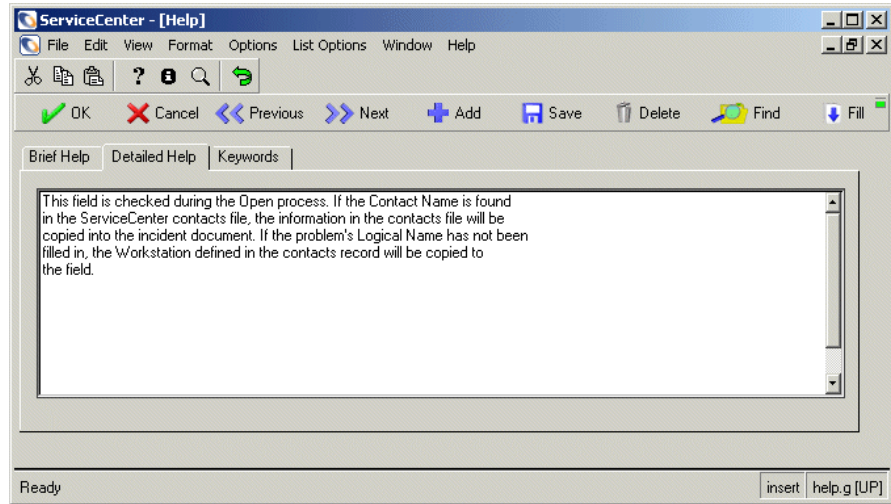


Figure 5-3: Detailed Help tab in the help record

Keywords Tab

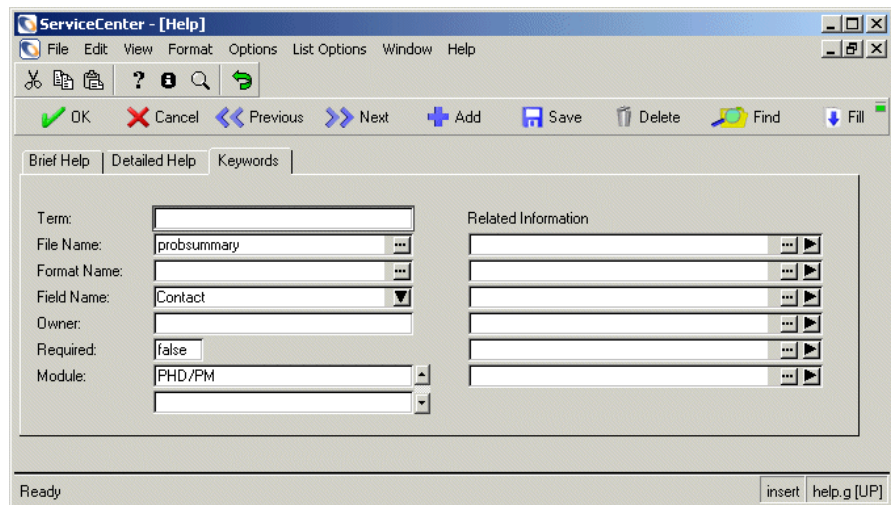


Figure 5-4: Keywords tab in the help record

Field	Description
Term	<p>Optional field describing terms which are used in the system, such as activity descriptions and menu entries. If the cursor is placed on a field that contains information when help is requested, the Help Utility searches the <i>help</i> file for records where the term matches the field contents. If a match is found, this term record can be viewed.</p> <p>The Term field also contains the value linked to the Related Information array on a separate help record.</p>
File Name	<p>Contains a valid ServiceCenter file name (Database Dictionary name). The value entered here is the name of the file to which the help record is related. Click the ellipsis button to fill in the name of the file associated with the field selected for the help request.</p> <p>The file name in the help record coincides with the field name. If the cursor is placed on a field within a file when help is requested, the Help Utility will search the help file for a record that matches with the field name and the file name. If a match is found, the record can be viewed for help information.</p> <p>When the file name is entered without a field name on the help record, the help record can only be accessed through the Database Dictionary.</p> <p>Note: You must include a file name when creating field help.</p>
Format Name	<p>Contains a valid ServiceCenter form name. This field is only required if the help record is intended to describe a form. Click the ellipsis button to fill in the name of the form in which the cursor is located when the help request is made.</p> <p>If the cursor is placed on a field within a file when help is requested, the Help Utility will search for a record within the help file that matches the field name and the format name. If a match is found, the record can be viewed for help information. This is helpful when there are several like-named fields for separate formats which require different help text.</p>

Field	Description
Field Name	<p>Contains a valid ServiceCenter field name (both on forms and within the Database Dictionary). Click the ellipsis button to fill in the name of the field in which the cursor is located when the help request is made.</p> <p>If the cursor is located on a field within a database file when help is requested, the Help Utility will search the help database for records matching the field name, file name, or the format name. If a match is found, this help record can be viewed for help information.</p>
Owner	<p>Optional field containing the name of the operator adding or updating the help record.</p> <p>The purpose of this field is to allow the user to show ownership of help records for maintenance purposes if desired. The value in this field may be anything the user chooses.</p>
Required	<p>Logical field indicating whether or not a value in this field is required. This is an optional field.</p>
Module	<p>Indicates the module to which this help record applies, for example, IM (Incident Management), SM (Service Management), ICM (Inventory and Configuration Management). This is a reference field only and is not linked to any other file.</p>
Related Information	<p>The Related Information array may contain any value or list of values. It's purpose is to allow the user to link to other help records containing information that is related to the current help record. These linked records may elaborate on the subject discussed in the current help record or introduce a new topic.</p> <p>Any entry made into the Related Information array must have a separate help record with that value in the Term field.</p>

User views

Users without the capability to create or edit help see help records without the editing buttons that are available to the system administrator. A Find button allows users to search keywords for related information.

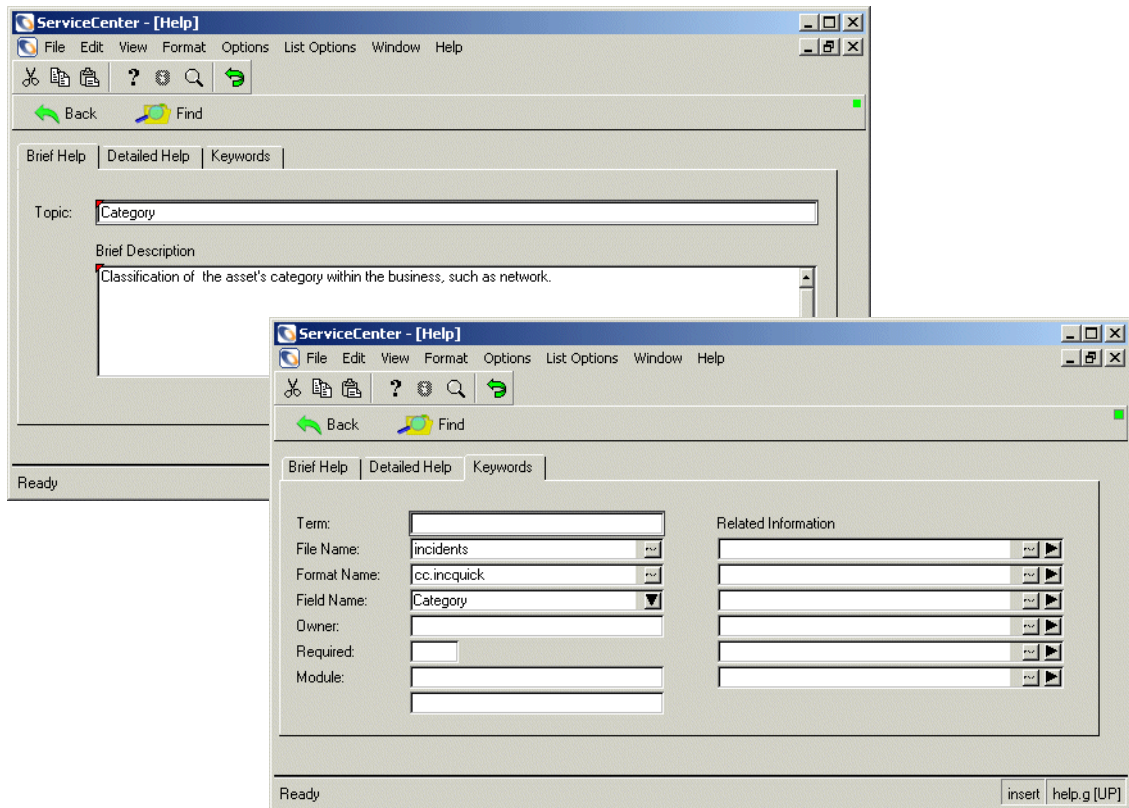


Figure 5-5: Read-only help record for regular users

Querying the Help File

If no help exists for the requested element, a search screen is displayed allowing the user to enter a plain-text query for related topics. The system automatically fills in the names of the, field, file, and form to assist you in your search.

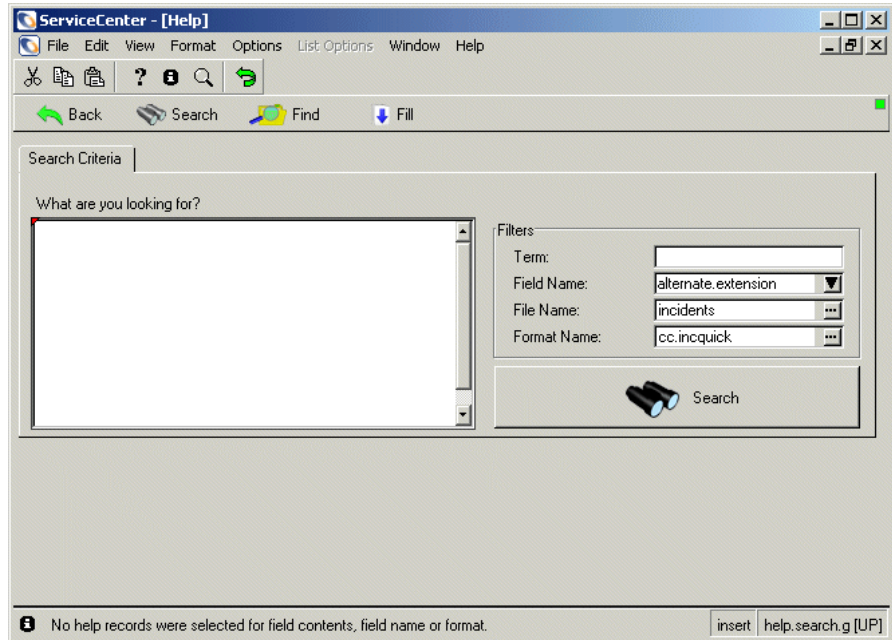


Figure 5-6: Help search form

Accessing the *help* File

System administrators and other users with the appropriate system capabilities can access the help file in the edit mode by using any of the following methods:

- Database Manager
- Command line
- Toolbar button
- Help menu.

Database Manager

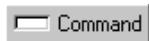
Use Database Manager access to search for records, edit online help, or export records to another system.

To access the *help* file from the Database Manager:

- 1 Click **Database Manager** in the Toolkit tab of the system administrator's home menu.
- 2 Type **help** in the **Form** field of the Database Manager dialog box.
- 3 Click Search or press Enter.
- 4 Select either **help.input.g** or **help.g** from the QBE list of forms.
- 5 Enter your search criteria.
- 6 Click Search or press Enter.

Command Line

To access the *help* file from a command line:



- 1 Click **Command** in the system administrator's home menu.
- 2 Type **help** in the command line displayed.
- 3 Press Enter.

The help.input.g form is displayed. This input form contains the same information as the help.g form, without the tabs.

Figure 5-7: Help input form

- 4 Enter your search criteria.
- 5 Click Search or press Enter.

Toolbar Button

Click the question mark button in the toolbar at the top of your screen to access online help.



Help Menu

The Help menu appears on all forms and menu in ServiceCenter and is accessible to all users. When accessed from a form, the Help Utility displays existing help for that form and provides Fill functionality from that form.

To access the help file from the Help menu, select **Help > Help on Field** in any ServiceCenter form or menu. If online help for that form already exists, the record (or a QBE list of records) is displayed. If no help exists, a blank help record is displayed.

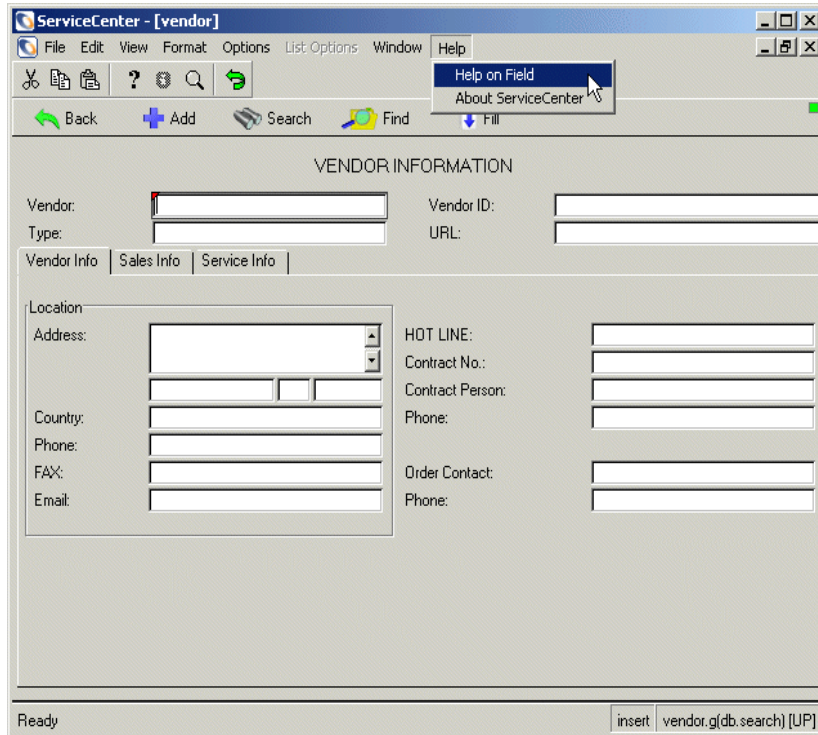


Figure 5-8: The Help on Field menu option

Types of Help

Users with the proper system capabilities (**SysAdmin** or **help**) can create or maintain records for the following types of online help:

- Field—file specific
- Field—format specific
- Field contents

- Forms/Menus
- Terms

In addition, some help records are linked to other help records that may relate to or expand on the initial help you requested.

All appropriate types of help for the current field, form, or file are displayed together in one QBE list when the **Help on Field** option is selected.

Field—File Specific

This type of help is useful for a field whose usage remains constant regardless of the form in which it appears. An example of this type of field would be the **assignment** field in the problem file.

The fields required for this type of help are:

- File Name
- Field Name

The screenshot shows the ServiceCenter - [Help] window. The menu bar includes File, Edit, View, Format, Options, List Options, Window, and Help. The toolbar contains icons for OK, Cancel, Add, Save, Delete, Find, and Fill. Below the toolbar are tabs for Brief Help, Detailed Help, and Keywords. The main area contains a search form with the following fields:

Term:	<input type="text"/>	Related Information	<input type="text"/>
File Name:	problem	<input type="text"/>	<input type="text"/>
Format Name:		<input type="text"/>	<input type="text"/>
Field Name:	assignment	<input type="text"/>	<input type="text"/>
Owner:		<input type="text"/>	<input type="text"/>
Required:	true	<input type="text"/>	<input type="text"/>
Module:	appm	<input type="text"/>	<input type="text"/>

The status bar at the bottom shows 'Ready' and 'insert help.g [UP]'.

Field—Form Specific

This type of help is useful for a field whose usage varies depending upon the form in which it appears.

The fields required for this type of help are:

- Format Name
- Field Name

The screenshot shows a window titled "ServiceCenter - [Help]" with a menu bar (File, Edit, View, Format, Options, List Options, Window, Help) and a toolbar with icons for Cut, Copy, Paste, Help, Find, and Refresh. Below the toolbar is a status bar with buttons for OK, Cancel, Add, Save, Delete, Find, and Fill. The main area has three tabs: "Brief Help", "Detailed Help", and "Keywords". The "Detailed Help" tab is active, showing a form with the following fields:

- Term: [Empty text box]
- File Name: [problem] [...]
- Format Name: [problem] [...]
- Field Name: [Empty dropdown menu]
- Owner: [Empty text box]
- Required: [true] [checkbox]
- Module: [appm] [^] [v]

On the right side, there is a "Related Information" section with a list of items:

- Document Management [...]
- Incident Management [...]
- [Empty list item] [...]
- [Empty list item] [...]
- [Empty list item] [...]

The status bar at the bottom shows "Ready" and "insert help.g [UP]".

Field Contents

This type of help is used to define the contents of drop-down lists or system generated field values. An example of this would be the value of **Open** in the **status** field of an incident ticket. The system automatically assigns a status of **Open** to all new tickets. Content help can be defined for **Open** or for any other available status in the drop-down list. The content of the field that you want to define must appear in the **Term** field of the help form.

When you select the **Help on Field** option, the Help Utility displays all the types of help associated with that field and form. For the **status** field, you might have a QBE list that displays a form help record, a field help record, and a dozen field content records.

The fields required for this type of help are:

- Term
- File Name
- Field Name

The screenshot shows the 'ServiceCenter - [Help]' dialog box. It has a menu bar (File, Edit, View, Format, Options, List Options, Window, Help) and a toolbar with icons for Cut, Copy, Paste, Help, Find, and Refresh. Below the toolbar are buttons for OK, Cancel, Previous, Next, Add, Save, Delete, Find, and Fill. The dialog is divided into three tabs: Brief Help, Detailed Help, and Keywords. The 'Detailed Help' tab is active. It contains two columns of fields. The left column has labels and input fields: Term (text box with 'Open'), File Name (text box with 'probsummary' and a browse button), Format Name (text box with a browse button), Field Name (dropdown menu with 'Problem Status'), Owner (text box with 'pign'), Required (checkbox checked), and Module (text box). The right column is titled 'Related Information' and contains five empty text boxes, each with a browse button. The status bar at the bottom shows 'Ready' and 'insert help.g [UP]'.

Form

This type of help is useful for identifying the general function of a form or a menu.

The fields required for this type of help are:

- Format Name

The screenshot shows a window titled "ServiceCenter - [Help]". The menu bar includes File, Edit, View, Format, Options, List Options, Window, and Help. The toolbar contains icons for OK, Cancel, Previous, Next, Add, Save, Delete, Find, and Fill. Below the toolbar are tabs for Brief Help, Detailed Help, and Keywords. The main area contains a form with the following fields:

- Term: [Text Field]
- File Name: [Text Field] ...
- Format Name: link.structure ...
- Field Name: [Text Field] ▾
- Owner: [Text Field]
- Required: [Text Field]
- Module: [Text Field]

To the right of these fields is a "Related Information" section with five empty text fields, each followed by a right-pointing arrow icon. The status bar at the bottom shows "Ready" on the left and "insert help.g [UP]" on the right.

Term

This type of help is used to link similar topics together and make them easily accessible from within the help file. This help is linked to entries in the keywords in the Related Information array and is intended to provide additional information to the initial help record.

The fields required for this type of help are:

- Term

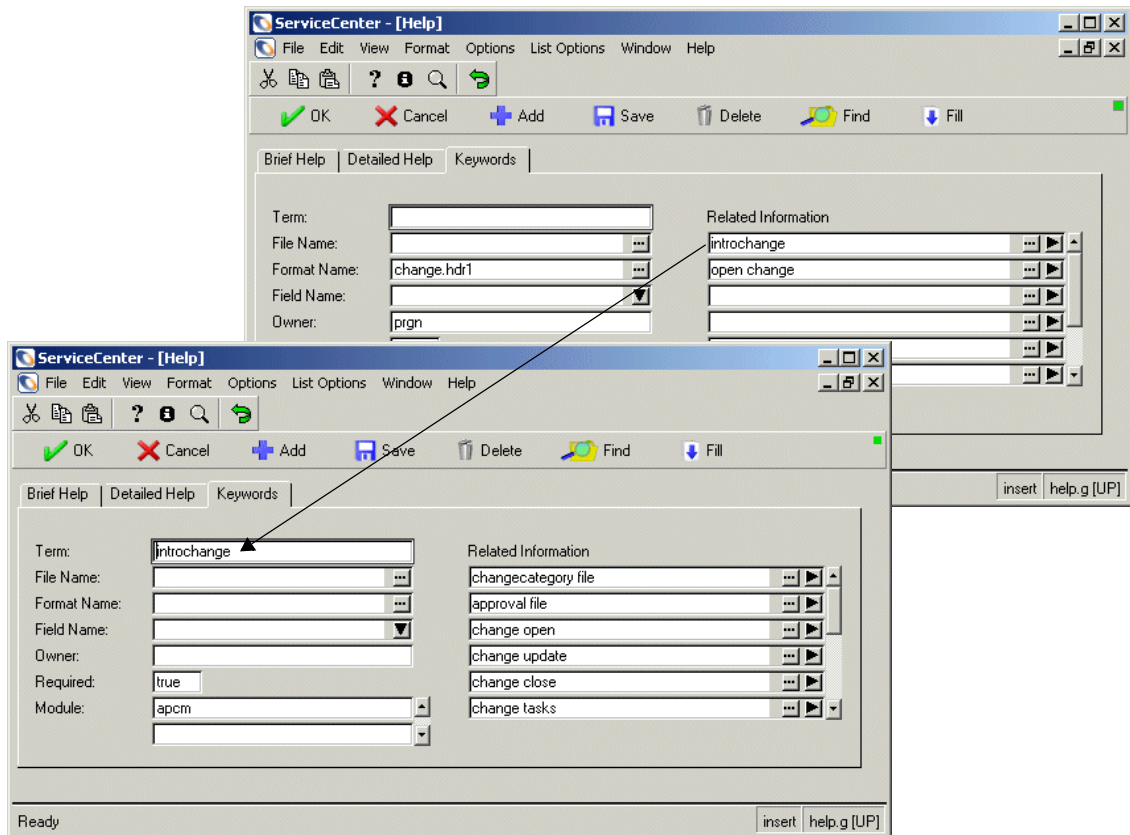


Figure 5-9: Term record for related information

Creating Online Help

Only users with appropriate capabilities can edit or add help records.

To create online help:

- 1 Open the form for which you want to create online help.
- 2 Select **Help > Help on Field**.

If no help exists for the current form, a blank help record (*help.g*) is displayed.

3 Fill in the form with information appropriate to the type of help desired.

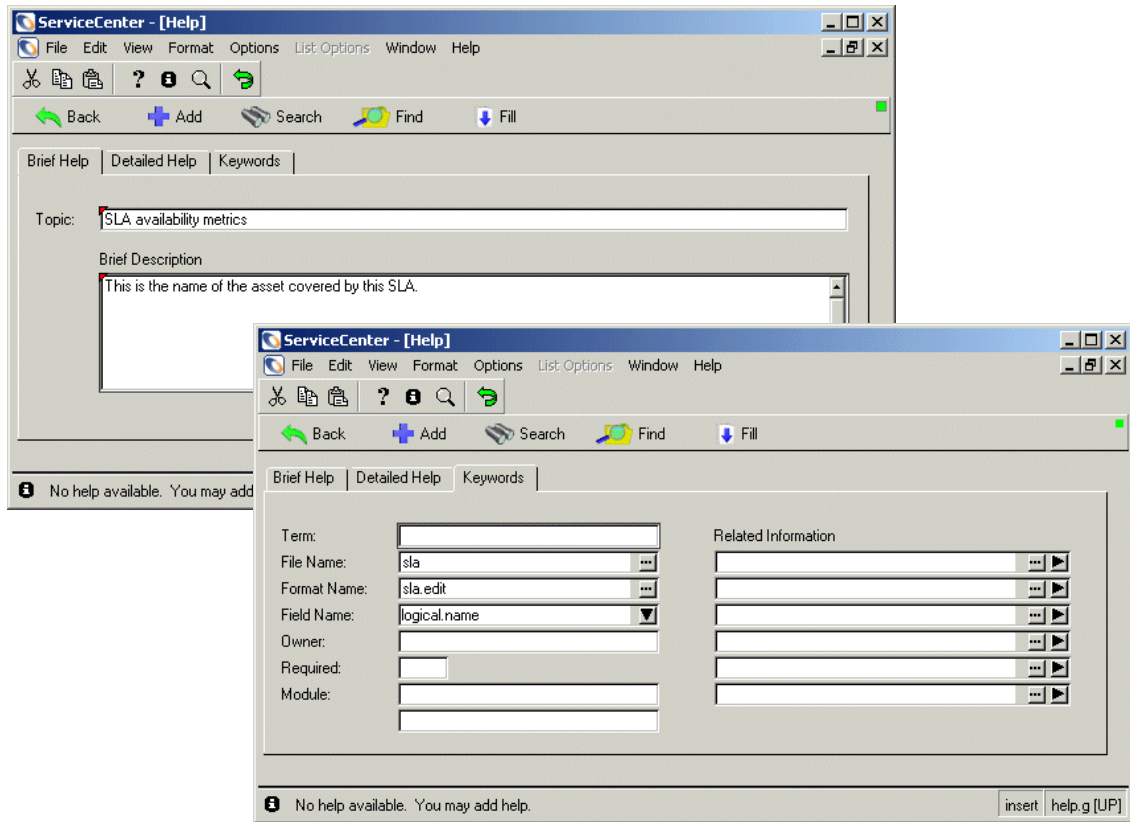


Figure 5-10: Help Utility—creating a new help record

4 Click Add to save your new help record.

The buttons in the toolbar change, and the following message appears in the status bar: *Help record added.*

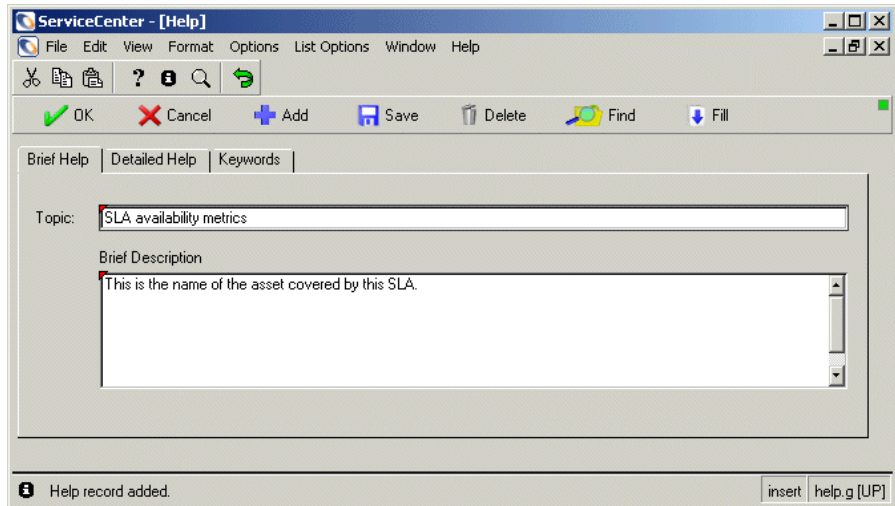


Figure 5-11: Help Utility—maintenance controls

- 5 Click **OK** to display a blank help record.
Use this procedure to create help records for keywords you have defined (See *Creating Keyword Records* on page 209).
- 6 Click **Cancel** to return to the record for which the help has been created.
Note: You may create any type of help using this procedure.

Related Information

The **Related Information** array contains *keywords* that are linked directly to field or form help records that contain information on related topics. Keywords do not appear on QBE lists of help associated with fields or forms and can only be displayed from a direct query of the help file.

If you make an entry in the **Related Information** array of a help record, you must create a separate help record containing the same keyword value in the **Term** field.

Creating Keyword Records

To create keyword records:

- 1 Access the help file from the Help menu.
The help.g form is displayed.
- 2 Create a new help record or search for an existing record.

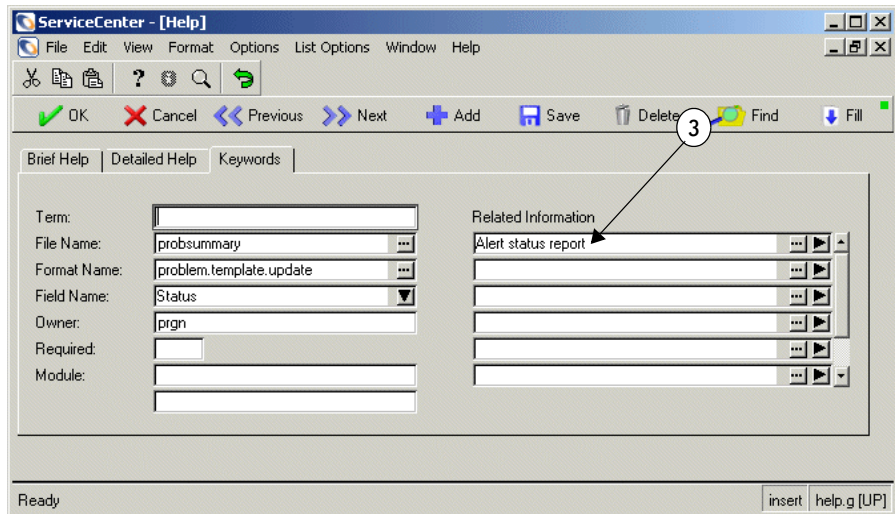
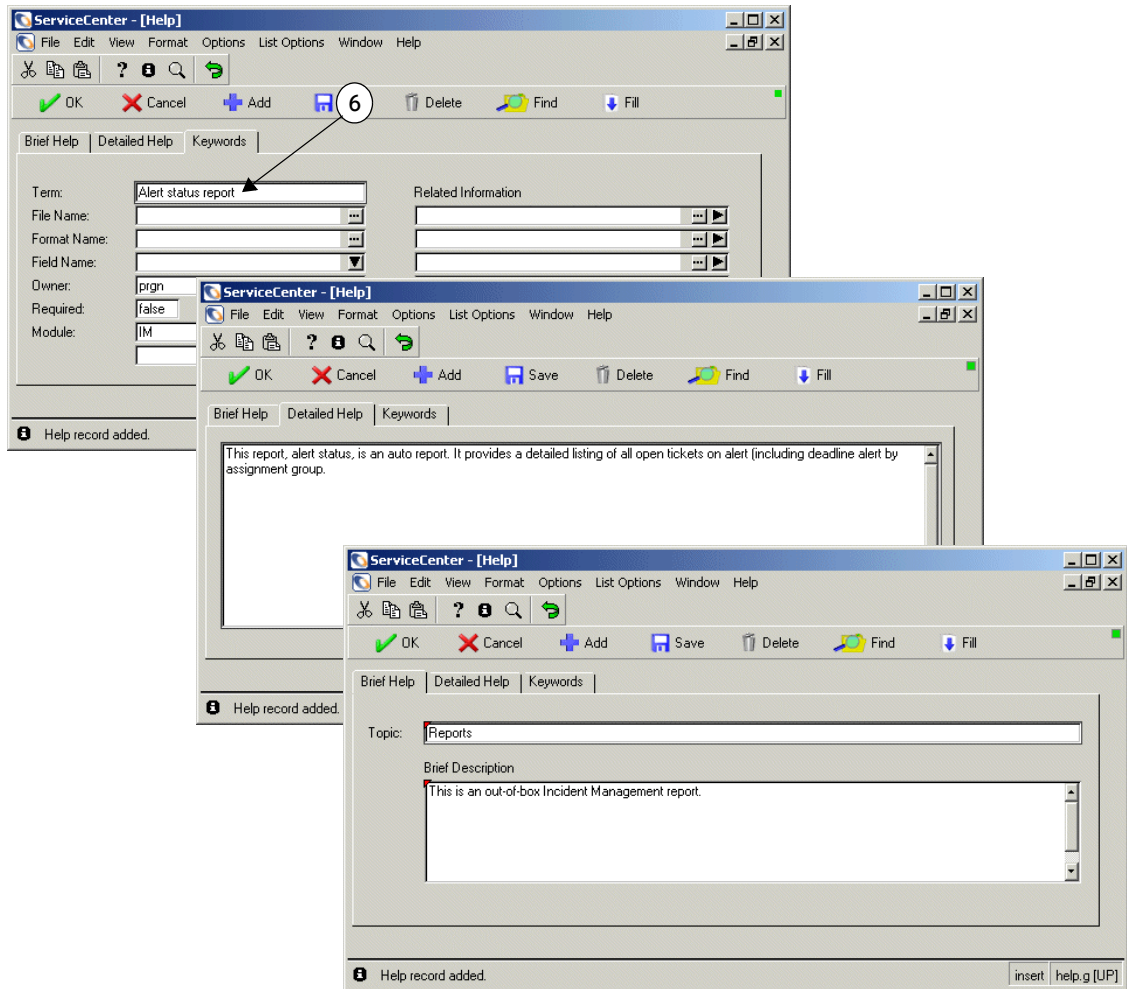


Figure 5-12: Expanded help form for related information

- 3 Type the descriptive title of a related topic in the **Related Information** array.
- 4 Click **Save**.
The following message appears in the status bar: *Help record updated.*
- 5 Click **OK**.
- 6 Click **Back** if a QBE list of help records is displayed.
A blank help record is displayed.
- 7 Type the same value in the **Term** field that you entered in the **Related Information** array in the previous record.
Remember that the **Term** field is the only required element in the **Keywords** tab for this type of help.

8 Fill in the necessary descriptions and additional keywords for related topics.



5-13: Related information record

- 9 Click Add.
- 10 Click OK to display a blank help record.
- 11 Create additional related information records as needed for all keywords.

Displaying Related Information

To display related information:

- 1 Open a field or form help record containing a keyword entry for related information.
- 2 Place the cursor in the **Related Information** array containing the term record you want displayed.

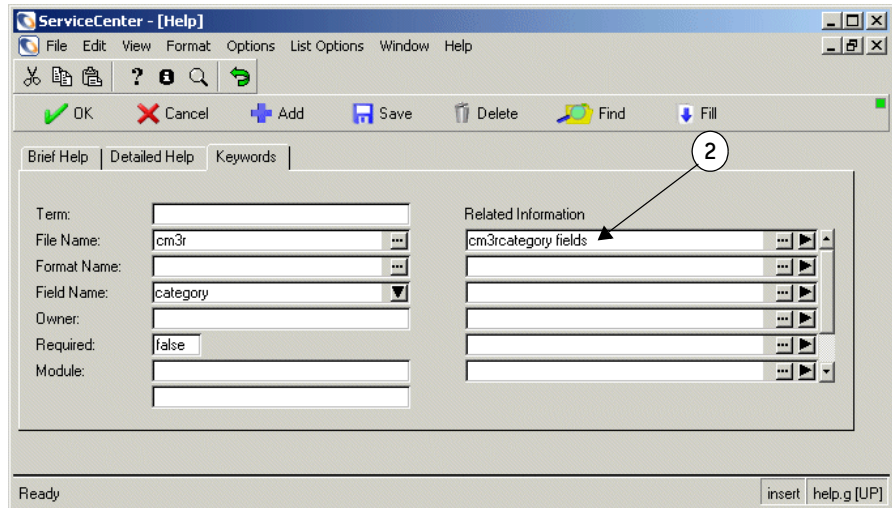


Figure 5-14: Help record—searching keywords

- 3 Click Find or press F8.

The selected related information record is displayed. If there are more than one record, a QBE list of records is displayed.

Topic	Field Name	File Name	Format Name	Term
Change categories	phases	cm3rcategory		cm3rcategory fields
Change categories	name	cm3rcategory		cm3rcategory fields
Change categories	description	cm3rcategory		cm3rcategory fields
Change categories	avail.cond	cm3rcategory		cm3rcategory fields
Change categories	assign.number	cm3rcategory		cm3rcategory fields

* Cannot find related information in help using query: format.name#"cm3rcategory fields" insert help.qbe.g [UP]

Figure 5-15: QBE list of related records

Note: In this help record, one keyword contains five related records, each referencing a different field.

- 4 Double-click on a record to view it.
- 5 Click **Cancel** to return to the QBE list.
- 6 Click **Back** to return to the principal help record.
- 7 Select another keyword to view or click **OK** to return to the record from which you selected help.

6 The Cascade Update Utility

CHAPTER

Cascade Updates

The Cascade Update utility allows Administrators to maintain database integrity and consistency by altering the data in one or more dependent files to match changes made to data in a source file.

For example, SYSBLOB records are linked to the contacts table by the contact name (contacts, contact.name = SYSBLOB, topic). If someone were to edit the contact record and change contact.name, the associated SYSBLOB records would lose their link and become orphaned. A Cascade Update could be set up to monitor contacts.contact.name and change the topic field in related SYSBLOB records to match accordingly.

Creating a Cascade Update

Cascade Updates are created in two steps:

- 1 Create a trigger record to monitor the source table.
- 2 Create a Cascade Update Configuration Record to set parameters for the update.

Creating a Trigger Record

Triggers monitor files for certain types of activity (Adds, Updates, Deletes) and launch a specified RAD application when the appropriate conditions are met.

To create a trigger record:

- 1 From the Database Manager utility, search for the file triggers.



Figure 6-1: The Trigger Record

The triggers record is displayed.

- 2 Create a trigger with the following parameters:

Parameter	Description
Trigger Name:	Any unique, meaningful name
Table Name:	Select the file to be monitored by this trigger.

Parameter	Description
Trigger Type:	<p>Select the specific activity to react to:</p> <p>1- Before Add - If the user is adding a new record to this file, launch the application prior to committing the add.</p> <p>2- After Add - If the user is adding a new record to this file, launch the application after committing the add.</p> <p>3- Before Update - If the user is modifying a record in this file, launch the application prior to committing the change.</p> <p>4 - After Update - If the user is modifying a record in this file, launch the application after committing the change.</p> <p>5 - Before Delete - If the user is deleting a record in this file, launch the application after performing the deletion.</p> <p>Note: Only triggers of type 4 - After Update, and type 6 - After Delete are supported by Cascade Updates.</p>
Application:	<p>The RAD application to launch.</p> <p>Note: The application cannot require parameters. It will have access only to \$L.old (the record prior to the change), and \$L.new (the record post-change).</p> <p>For most Cascade Updates, set this to <code>cascade.update.wrapper</code>. The only exception to this would be if the file you will be monitoring is the device file or any related device attribute file (joincomputer, etc.). In that case, set this to <code>am.cascade.update.wrapper</code>.</p>

3 Click Add to add the new trigger.

Create a Cascade Update Configuration Record

- 1 From the Main menu, select the Utilities tab.
- 2 Select Tools.
- 3 Click Cascade Updates.

Cascade Update Config Record

Name: Type:

Execute Condition: Display Scope of File Variables:

Configuration Rules

Update File:

Execute Condition:

Description:
(Example: field name=field name in \$L.old)

Adv Query:

Source Key: Target Key:
Source/Target Key fields will be used to retrieve records

Source Fields: Target Fields:

Expressions:

Figure 6-2: The Cascade Update Configuration Record

Field Descriptions

Name - The unique name of the record. This must coincide with the name of the file defined in the trigger record **Table Name** field. For example, if you wanted to monitor the contacts table for changes to records, you would enter contacts.

Note: If the file you will be monitoring is the device file or any related device attribute file (joincomputer, etc.), simply enter *device* (rather than any specific device join names).

Type - This must coincide with the **Trigger Type** field defined in the *trigger* record.

Note: The configuration file will only be processed off of a trigger using a the 4 - After Update and/or 6-After Delete trigger types.

After Update - This cascade will be triggered by modifications to an existing record.

After Delete - This cascade will be triggered by deleting an existing record.

Execute Condition - A logical expression defining the condition that must be met before any of the configuration rules for this file should be processed. If nothing is entered the default value is 'true'.

For example, to run cascade updates only if the logical.name field in the record was modified, enter the following:

```
logical.name in $L.old~=logical.name in $L.new
```

File variables in scope:

- \$L.old - The state of the main record before the update.
- \$L.new - The state of the main record after the update.

Display Scope of File Variables - This check box provides a quick reference for an Administrator. when this is checked, the file variables within scope for each expression are displayed. (See Figure).

Configuration Rules - Configuration Rules is an array of structures containing the information for each cascade update triggered by this file. Each field listed here is repeated for every rule.

Update File: The file to be modified by this rule.

Execution Condition: A logical expression defining the condition that must be met before this specific configuration rule should be processed. If nothing is entered the default value is 'true'.

For example, to process this rule only if the istatus field of the record was modified, enter the following:

```
istatus in $L.old~=istatus in $L.new
```

File variables in scope:

- \$L.old - The state of the main record before the update.
- \$L.new - The state of the main record after the update.

Description - Brief description of what this configuration rule will do.

Advanced Query - This query is used to retrieve the records from the update file that will be affected by this rule.

Note: If the Advanced Query field is left blank, the Source Key/Target Key fields will be used to select records instead. If anything is in this field, the Source Key/Target Key fields will be ignored. Either Advanced Query or Source Key/Target Key must be entered.

For example, to affect only records from the update file where the **id** field matches the **logical.name** field, and the **status.proration** field matches the **istatus** field in the record being changed, enter the following:

```
id=logical.name in $L.old and status.proration=istatus in $L.old
```

File variables in scope:

- \$L.old - The state of the main record before the update.
- \$L.new -The state of the main record after the update.

Source Key/Target Key - If **Advanced Query** is left blank, these fields will be used to build the query that retrieves the records to be affected by this rule.

Note: If any text is entered in the **Advanced Query** field, the **Source Key/Target Key** fields will be ignored. Either **Advanced Query** or **Source Key/Target Key** must be entered.

For example, if the **Source Key** is **logical.name** and the **Target Key** is **id**, the query processed against the target file will be:

```
id=logical.name in $l.old
```

Expressions:

An array of expressions which are executed after each record is updated.

File variables in scope:

- \$L.old - The state of the main record before the update.
- \$L.new -The state of the main record after the update.
- \$L.target - The record in the new file to which you want to make changes.

Note: Only changes to the fields in \$L.target can be made

- 1 Click **Add** to add your new Cascade Update record.

Example

ACME HQ has moved and now needs to update all of their contact records to reflect the new location “ACME CORPORATE”. Use the Cascade Update Utility to automatically update all ACME HQ *contact* records when the *location* has been updated in the ACME HQ *location* record.

There are two phases to make this happen:

- Add a trigger record.
- Add a Cascade Update Configuration record.

First, set a trigger to monitor the location file and call the RAD application, *cascade.update.wrapper*, when the *location* record is updated (After Update) in the *location* file:

- 1 From the **Toolkit** tab, select **Database Manager**.
- 2 In the **Form** field, type **triggers** and press ENTER.

The trigger record is displayed.

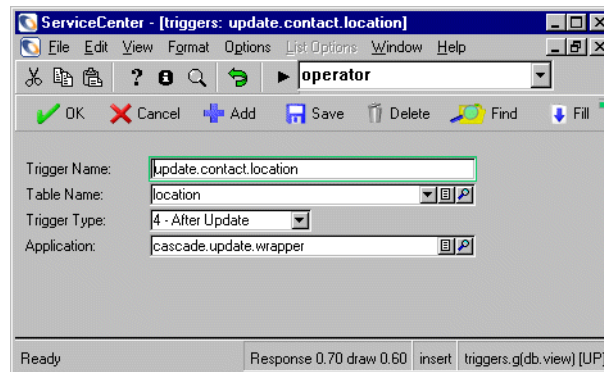


Figure 6-3: Completed Trigger Record

- 3 Type the following information in the *triggers* record:
Trigger Name: update.contact.location
Table Name: location

Trigger Type: 4 - After Update

Application: cascade.update.wrapper

Note: If you are configuring a cascade update for device records, type `am.cascade.update.wrapper` in the **Application** field. This wrapper works with the device joindef files.

4 Click Add.

The message line indicates that the trigger record was added.

Next, configure a Cascade Update Configure Record to update the *location* in all *contact* records associated with that same location:

1 From the Utilities tab, select Tools > Cascade Updates.

The Cascade Update Config Record is displayed.

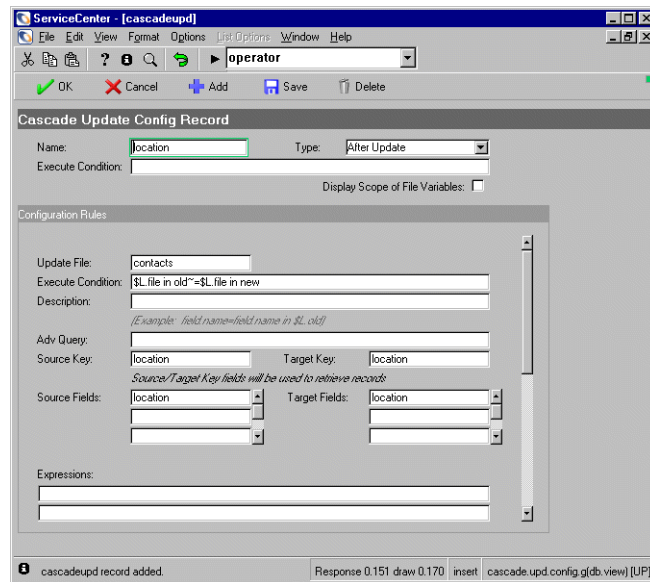


Figure 6-4: Completed Cascade Update Config Record

2 Type the following information in the *Cascade Update Config Record*:

Name: location

Type: After Update

(Configuration Rules)

Update File: Contacts

Execute Condition: location in \$L.new~=location in \$L.old

Source Key: location

Target Key: location

Source Fields: location

Target Fields: location

3 Click Add.

Test:

- 1** From the **Toolkit** tab, select **Database Manager**.
- 2** In the **Form** field, type contacts and press **Enter**.
- 3** From the **Options** menu, select **Modify columns**.
- 4** Use the drop-down arrow to change the “company” field to “location”.
- 5** Click **Proceed**.
- 6** Click in the heading of the location column to sort the list so that all contacts with the same location are shown together.
- 7** Open a second session of ServiceCenter.
- 8** From the **Toolkit** tab, select **Database Manger**.
- 9** Click **Search**.
- 10** Double-click on an ACME HQ record in the record list.
The ACME HQ location file is displayed.
- 11** Change the location by typing ACME CORPORATE in the location field.
- 12** Click **Save**.
- 13** Revert to the first session of ServiceCenter, which is still showing the record list of contact records.
- 14** Click **Refresh**.
- 15** All of the ACME HQ locations are now ACME CORPORATE.

7 Display Options

CHAPTER

This chapter addresses the advanced issues related to display options, including the following topics:

- Creating options
- Testing options
- Balloon help

Creating Options

The basic procedures for adding an option to a ServiceCenter form are as follows:

- 1 Decide what type of option (button or menu item) to add or modify.
- 2 Locate the appropriate screen ID.
- 3 Modify an existing option record or create a new one.
- 4 Make appropriate changes (if necessary) to the form using Forms Designer.

Sorting a column

In this example, you will create a new column called “Assignment” and will create a menu option for the column in the ServiceManagement Call Queue (apm.manage.incident.g). The option will sort calls alphabetically by assignment when a user clicks on the option.

Displayoption records currently exist for the other columns in the Service Management Call Queue.

Identify the Screen ID

Identifying the Display screen ID for a particular form begins by noting the name of the form, as shown in the following procedure:

- 1 Click **Service Management** on the Services tab of the home menu.
- 2 Click **Call Queue** on the Service Management menu.
- 3 Switch inboxes to **All Open Calls**.

The Service Management Call Queue appears.

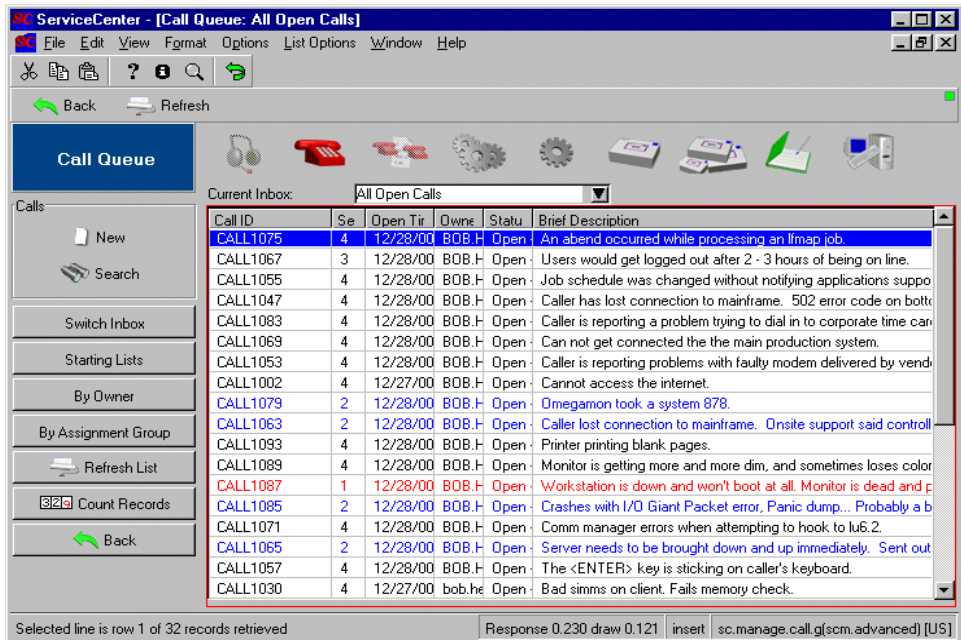


Figure 7-1: Service Management Call Queue

- 4 Click **View** on the menu bar and select the **Form Name** and **Screen Name** options. The form and screen name appear in the format <form name>(<screen name>). In this case, the form is **sc.manage.call.g**, the screen is **scm.advanced**.
- 5 Return to the home menu.

Modify the Form

You must add the assignment column to the form using the **Forms Design utility**.

- 1 Select the **Toolkit Tab** from the home menu.
- 2 Click **Forms Designer**.
- 3 Open the form **sc.manage.call.g**
- 4 Click **Design** to edit the form.
- 5 Click on the bottom of the table to edit it's properties in the **Object Properties** dialog.
- 6 Highlight the **Columns** property and click **Edit List**.
- 7 7. Add a new item called "Assignment" and click **OK**.
- 8 8. Click on the header for the new Assignment column to edit it's properties in the **Object Properties** dialog.
- 9 9. Highlight the **Input** property. Write in the value "assignment" and click **Y** to apply.
- 10 10. Save and close the form

Create the *displayoption* record

To create a new option record for the Assignment column, you will copy and modify a record from one of the other columns, then add the modified record to the database under a new name.

- 1 Select the **Utilities** tab from the home menu.
- 2 Click **Tools** to open the GUI tools menu.
- 3 Click **Display Options** to open the **displayoption** form.
- 4 In the form, enter the screen ID for the desired form, **scm.advanced** in this case and press **Enter**. The **display option** screen displays, with the **scm.advanced** options in a qbe list.

The screenshot shows the ServiceCenter application window titled "ServiceCenter - [displayoption]". The window contains a menu bar (File, Edit, View, Format, Options, List Options, Window, Help) and a toolbar with icons for Cut, Copy, Paste, Help, Search, and Refresh. Below the toolbar is a navigation bar with buttons for OK, Cancel, Previous, Next, Add, Save, and Delete.

A table displays a list of display options:

GUI	Bank	Option	Screen ID	Default Label	Text Label
2	1	11	scm.advanced	Refresh	
3	1	3	scm.advanced	Back	
51	5	12	scm.advanced		Get Inbox
52	5	11	scm.advanced		Get Sub Inbox
500	3	1	scm.advanced	Select Queue	
501	3	2	scm.advanced	Switch Queue>Inventory Ma	ICM List

Below the table is the "Display Application Option Definition" form. The fields are as follows:

- Screen ID: scm.advanced
- Language: ENG
- Action: refresh
- Unique ID: 3150
- Modifies Record (back, close, and more are special)
- GUI option: 2
- Balloon Help (If Option < 200): Refresh
- Text Option: 11
- Default Label: Refresh
- Bank: 1
- Text Alternative:
- Condition: true
- User Condition:

Below the form is the "RAD" section with tabs for "RAD" and "Comments". The "RAD" tab contains:

- Expressions evaluated after option is selected, but before RAD call
- RAD Application: (empty)
- Names: (empty)
- Separate Thread?: (empty)
- Values: (empty)
- Post RAD Expressions: (empty)

The status bar at the bottom shows "Ready", "Response 0.150 draw 0.150 insert", and "displayoption.qbe.g [US]".

Figure 7-2: Displayscreen Record

- 5 Select an already assigned option in the 800 range. (for example, Option 870, the By Owner option).

The screenshot shows the 'Display Application Option Definition' dialog box. The 'RAD' tab is selected, showing the following fields and values:

- Screen ID: scm.advanced
- Language: ENG
- Action: newfile
- Unique ID: 3171
- Modifies Record:
- Balloon Help (If Option < 200): Incidents by Owner
- GUI option: 870
- Text Option: 1
- Default Label: By Owner
- Bank: 5
- Text Alternative: By Owner
- Condition: \$L.filename="probsummary"
- User Condition: (empty)

The 'RAD' section contains the following expressions:

- Expressions evaluated after option is selected, but before RAD call:
 - \$L.temp="\$L.name"
 - \$L.owner.title=scmsg(489, "us")
- RAD Application: apm.pop.dialog
- Separate Thread?: false
- Names: name, query
- Values: apm.choose.operator, \$L.temp
- Post RAD Expressions:
 - \$L.query="flag = true and ticket.owner = \"\"+\$L.name+\"\",\$L.query.name=scmsg(101, "sm", {\$L.name});if {\$L.name=NI
 - if {\$L.name=NULL or \$L.name="%None%"} then {\$L.query="flag=true and ticket.owner=NULL"}"

The status bar at the bottom indicates: Selected line is row 28 of 32 records retrieved. Response 0.150 draw 0.171 insert displayoption.qbe.g [US]

Figure 7-3: Creating a new displayoption Record

- 6 Modify the following fields to read:

Field	Value	Explanation
Balloon Help	Sort by Assignment	
Default Label	Sort by Assignment	
GUI Option	904	GUI Option = any unique ID greater than 200.

Field	Value	Explanation
Text Option	2	Text option = the F-key used for this option in the text-only client (in this case, F3).
Text Alternative	Sort by Assignment	
Bank	7	Bank = the F-key bank to use (in cases where more than 12 F-key options are needed).
Expressions...	<code>\$L.messages=insert(\$L.messages, 1, 1, "sort:0{\"assignment,1\"}")</code>	sort:0 for ascending sort:1 for descending

Note: Assigning a number greater than 200 (e.g., 904) to an option places that option in the **Options** drop-down menu. Options with values less than 200 appear as buttons in the system tool tray.

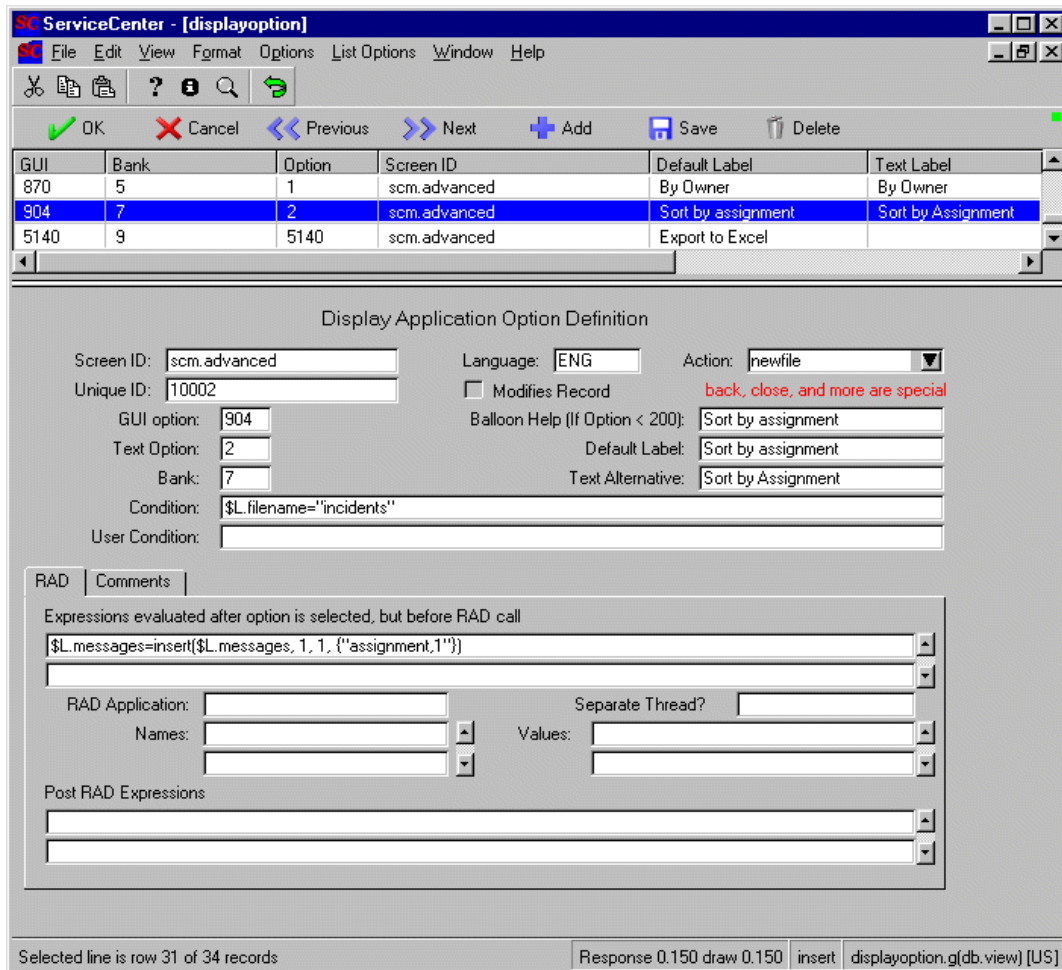


Figure 7-4: The new display option

- 7 Click Add to commit the record to the displayoption file.
- 8 Exit the Display Options utility.
- 9 Cycle the client by logging out then logging back in.

Testing your new option

Attempting to use the new Display control is the next step in creating display options.

- 1 Click Service Management on the home menu.

- 2 Click **Call Queue** on the Service Management menu.
The `sc.manage.call.g` form is displayed.
- 3 Pull down the **Options** menu.
- 4 Select the **Sort by Assignment** option to sort the queue in descending, alphabetical order, based on the call assignment group.

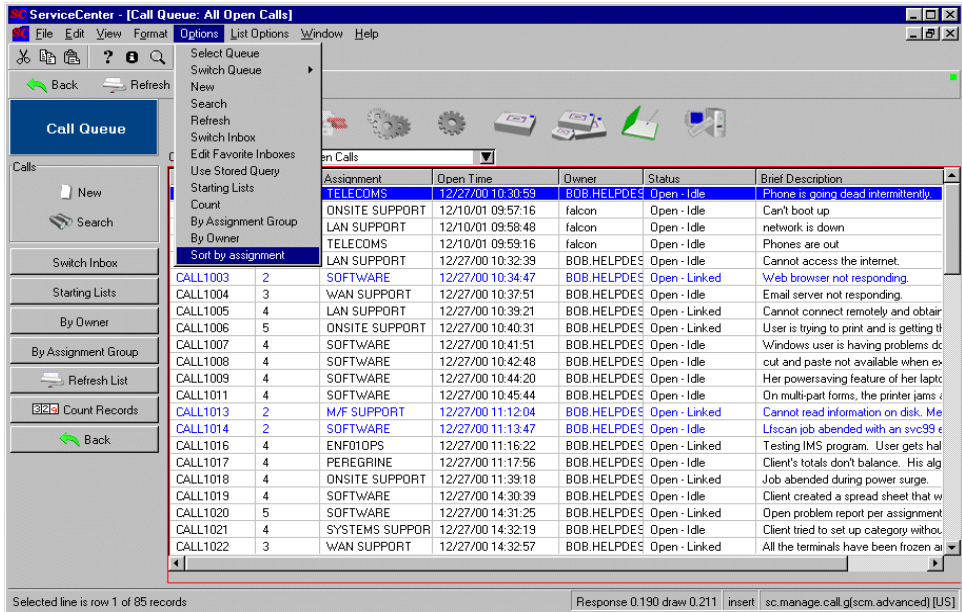


Figure 7-5: Incidents Sorted by Assignment

The results in the queue resort, descending alphabetically, according to the name of the assignment group for each Service Management report.

- You can also click on the Assignment column header to resort the column.

Opening an external application

In this example, you will create an option in the Service Management Call Queue that will call the `message.fc` application and open a Microsoft Word document. This process also can be used to open NotePad or other Windows program, as well as display images created in a graphics program.

The elements of this exercise are arranged in the following order:

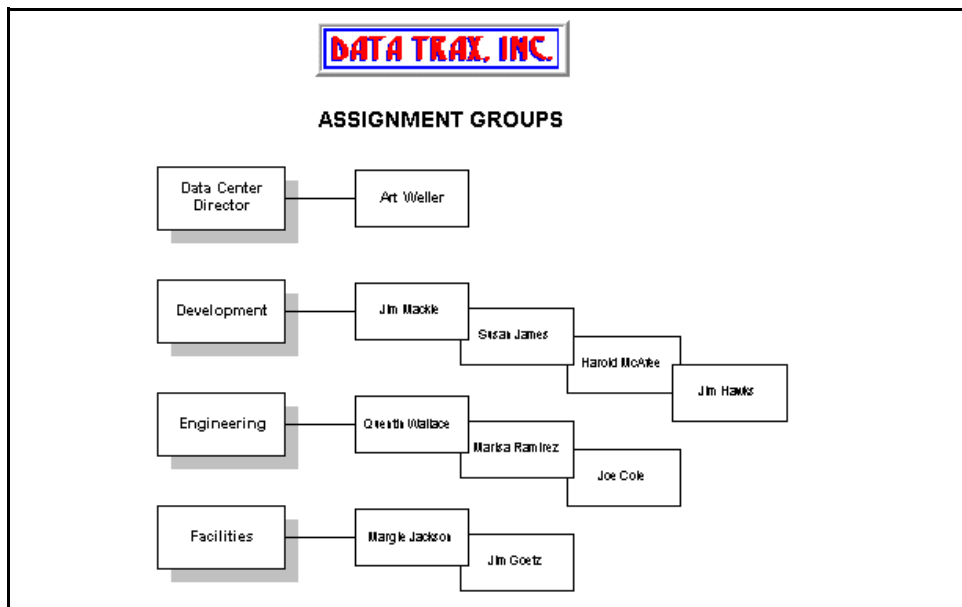
- Create the external file to call.
- Create the `displayoption` record.

Note: Use the same screen ID as you did in the previous example.

Create the reference file

Create a file in Microsoft Word or Notepad and save it to your local drive. This file does not need to be saved in the ServiceCenter directory. You may reference both text documents and graphic images.

For this example, we created an organizational chart of assignment groups and saved it in Microsoft Word.



Note: Write down the path of the file you plan to display from the option; you must define this path in the `displayoption` record in order for the `message.fc` application to locate the file.

Create the *displayoption* record

- 1 Open the System Tools menu by clicking **Tools** on the Utilities tab at the home menu.
- 2 Click **Display option** and open the record for `scm.advanced`.

The screenshot shows the ServiceCenter application window titled "ServiceCenter - [displayoption]". The window contains a menu bar (File, Edit, View, Format, Options, List Options, Window, Help) and a toolbar with icons for OK, Cancel, Previous, Next, Add, Save, and Delete. Below the toolbar is a table with the following data:

qui.option	txt.bar	txt.op	screen.id	labels.default	labels.txt.alterr	action	condition	labels.balloon
2	1	11	scm.advanced	Refresh		refresh	true	Refresh
3	1	3	scm.advanced	Back		exit	true	Go Back
51	5	12	scm.advanced		Get Inbox	getinbo	gui()	Get New Inbox
52	5	11	scm.advanced		Get Sub Inbox	getinbo	gui()	Get Sub Inbox

Below the table is the "Display Application Option Definition" form. The form contains the following fields and controls:

- Screen ID: scm.advanced
- Unique ID: 3145
- GUI option: 3
- Text Option: 3
- Bank: 1
- Condition: true
- User Condition: (empty)
- Language: ENG
- Action: exit
- Modifies Record
- Balloon Help (If Option < 200): Go Back
- Default Label: Back
- Text Alternative: (empty)
- RAD Application: (empty)
- Names: (empty)
- Values: (empty)
- Separate Thread?: (empty)

At the bottom of the form, there is a status bar that reads: "Selected line is row 2 of 32 records retrieved" and "Response 0.301 draw 0.220 insert displayoption.g(db.view) [US]".

Figure 7-6: scm.advanced Displayoption Record

- 3 Select the **back** option (see Figure 7-3 on page 227) and modify the fields according to the following table:

Field	Value
Action	do nothing
Balloon Help	Display Org Chart
Default Label	Display Org Chart
GUI Option	4
Text Option	3
Bank	1
Condition	true
RAD Application	message.fc
Separate Thread	true
Names	text
Values	WINEXEC;c:\<path to filename>

- 4 Click **Add** to add the record to the `displayoption` file.
- 5 Exit the `displayoption` form, and cycle client.

Test the new option

- 1 Click Service Management on the home menu.
- 2 Click Call Queue on the Service Management menu.

The `sc.manage.call.g` form is displayed with the option you have created on the menu bar.

The screenshot shows the ServiceCenter application window titled "ServiceCenter - [Call Queue: All Open Calls]". The interface includes a menu bar (File, Edit, View, Format, Options, List Options, Window, Help) and a toolbar with icons for Back, Refresh, and Org Chart. A red arrow points to the "Org Chart" button in the toolbar. The main display area shows a "Call Queue" sidebar with options like New, Search, Switch Inbox, Starting Lists, By Owner, By Assignment Group, Refresh List, Count Records, and Back. The main area displays a table of call records with columns: Call ID, Severity, Assignment, Open Time, Owner, and Status. The status of each call is "Open - Idle".

Call ID	Severity	Assignment	Open Time	Owner	Status
CALL1075	4	DEFAULT	12/28/00 14:59:16	BOB.HELPDES	Open - Idle
CALL1067	3	DUTYMANAGER	12/28/00 14:22:19	BOB.HELPDES	Open - Idle
CALL1055	4	ENF010PS	12/28/00 13:29:09	BOB.HELPDES	Open - Idle
CALL1047	4	FIELD ENG.	12/28/00 12:43:25	BOB.HELPDES	Open - Idle
CALL1083	4	LAN SUPPORT	12/28/00 15:11:14	BOB.HELPDES	Open - Idle
CALL1069	4	LAN SUPPORT	12/28/00 14:43:38	BOB.HELPDES	Open - Idle
CALL1053	4	LAN SUPPORT	12/28/00 13:27:20	BOB.HELPDES	Open - Idle
CALL1002	4	LAN SUPPORT	12/27/00 10:32:39	BOB.HELPDES	Open - Idle
CALL10011	4	LAN SUPPORT	12/10/01 09:58:48	falcon	Open - Idle
CALL1079	2	M/F SUPPORT	12/28/00 15:07:07	BOB.HELPDES	Open - Idle
CALL1063	2	M/F SUPPORT	12/28/00 13:51:22	BOB.HELPDES	Open - Idle
CALL1093	4	ONSITE SUPPORT	12/28/00 15:30:49	BOB.HELPDES	Open - Idle
CALL1089	4	ONSITE SUPPORT	12/28/00 15:28:55	BOB.HELPDES	Open - Idle
CALL1087	1	ONSITE SUPPORT	12/28/00 15:22:48	BOB.HELPDES	Open - Idle
CALL1085	2	ONSITE SUPPORT	12/28/00 15:17:31	BOB.HELPDES	Open - Idle
CALL1071	4	ONSITE SUPPORT	12/28/00 14:46:57	BOB.HELPDES	Open - Idle
CALL1065	2	ONSITE SUPPORT	12/28/00 13:54:15	BOB.HELPDES	Open - Idle
CALL1057	4	ONSITE SUPPORT	12/28/00 13:42:21	BOB.HELPDES	Open - Idle
CALL1030	4	ONSITE SUPPORT	12/27/00 16:01:37	bob.helpdesk	Open - Idle
CALL1026	4	ONSITE SUPPORT	12/27/00 14:36:34	BOB.HELPDES	Open - Idle
CALL10010	4	ONSITE SUPPORT	12/10/01 09:57:16	falcon	Open - Idle

Selected line is row 1 of 32 records retrieved Response 0.170 draw 0.121 insert sc.manage.call.g(sc.manage.call.g) [US]

Figure 7-7: New Display Option

- 3 Click the new option button to display the specified document/image. In this example, the **Org Chart** button launches Microsoft Word and opens the file **Orgchart.doc**.

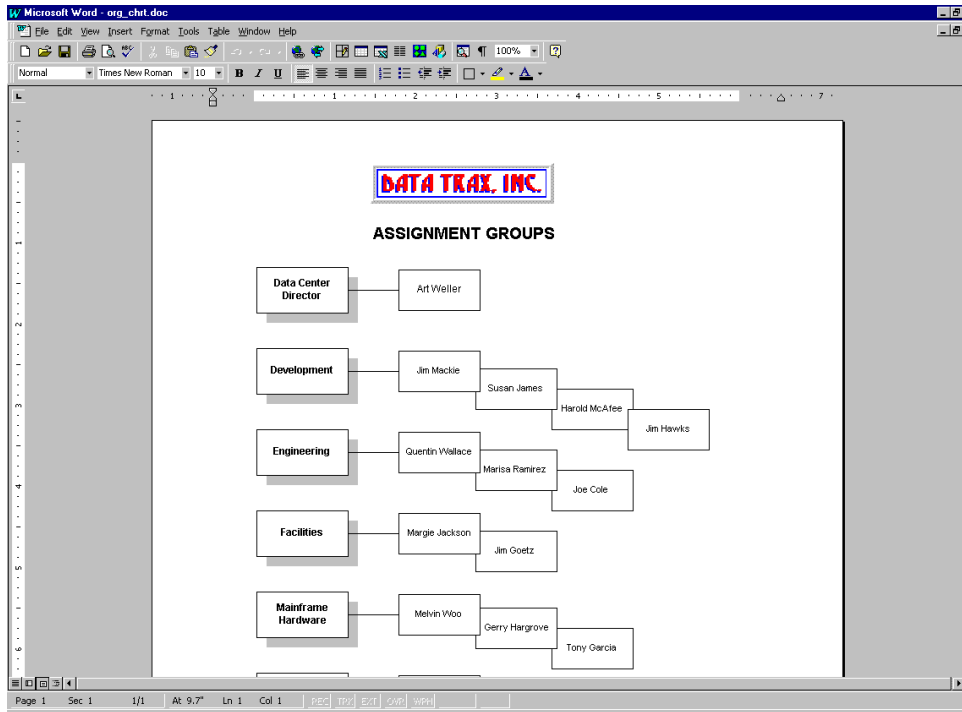


Figure 7-8: Displaying a Document from Outside the System

Balloon Help

The balloon help feature in the **displayoption** form allows a system administrator to create an instructional balloon for a newly created option. The message appears when the cursor is placed over a button and appears in a single line. Balloon help is limited in length only by its ability to fit on your screen; however, a brief, clear message is desirable.

Create balloon help from the displayoption panel, in the Balloon help field of the option you are creating the message for. See Figure 7-6 on page 232.

Note: Balloon help works for button options only—those with an option number of 200 or less.

8 The Display Application

CHAPTER

Display allows System Administrators to customize certain features of their system without altering RAD code.

Data files within **Display** contain the individual records, or *screens*, in which options, events, and window controls are defined. Each screen is attached to a particular application and controls the features appearing on the forms associated with that application. The information from these screens is stored in data tables and is not embedded in the RAD code. A system administrator has ready access to all **Display** features and may edit them freely.

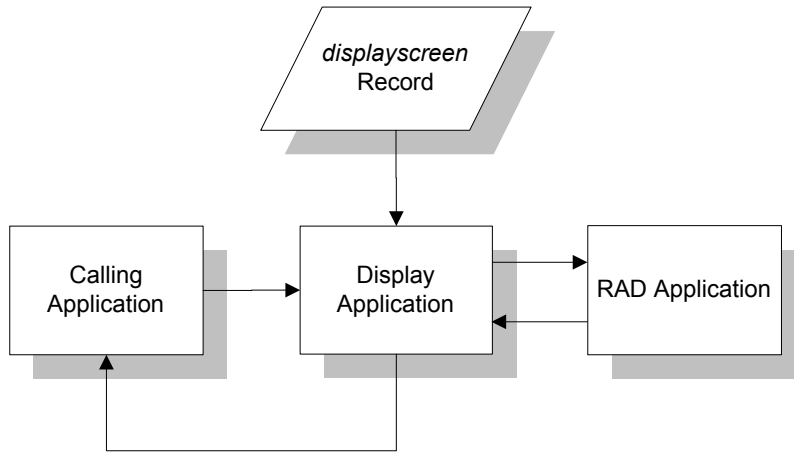


Figure 8-1: Display application workflow

Note: Because custom features are independent of RAD programming, Display options are protected from conflict during the upgrade process.

Accessing Display Records

The Display application reads data from four data files:

- displaymaster
- displayscreen
- displayoptions
- displayevents

Use the following procedure to access these data files:

- 1 Select the **Toolkit** tab in the home menu.
- 2 Click **Database Manager**.
- 3 Type **display** in the **Form** field of the Database Manager dialogue box.
- 4 Press **Enter**.

A QBE list of data file records appears.

Format Name	File Name	Last Update	Updated By	Language
display.cv.initial	applicati	12/20/96	rmari	en
display.cv.initial.g	applicati	02/08/00	lisa	en
displaycache	displayc	06/26/98	pcasey	en
displayevent	displaye	03/29/98	pcasey	en
displaymaster	displaym	12/18/96	rmari	en
displaymaster.g	displaym	01/14/00	Phil	en
displayoption	displayo	03/29/98	pcasey	en
displayscreen	displays	02/19/99	pbudic	en

Figure 8-2: QBE List of *display* Datafiles

- 5 To configure the Display application, open the **displaymaster** file first. Or select a data file from the record list, and double click the name to open it.

Displaymaster File

The displaymaster file contains the general controls for Display. There is one record per supported language. The default is ENG for English.

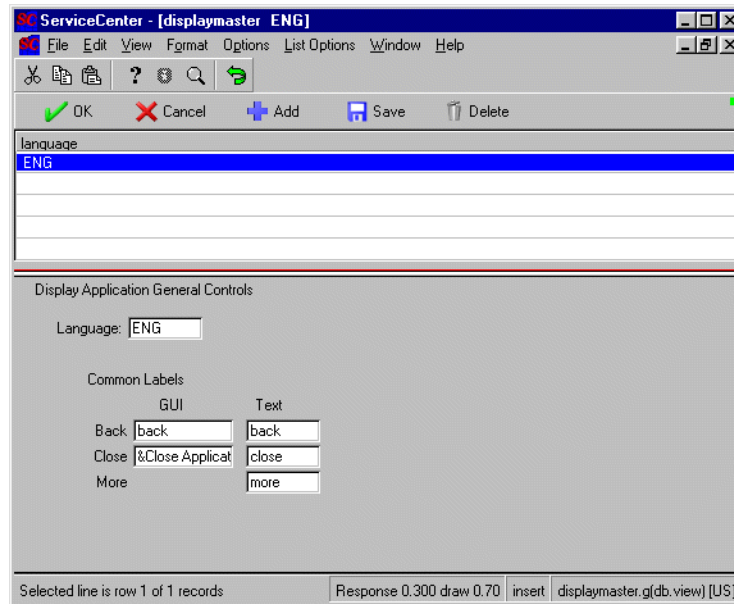


Figure 8-3: The Displaymaster Form

The following are the field definitions of the fields on the *displaymaster* form.

Language—selects the display language. The default is English (ENG).

Back—enables the button option (GUI) or the F-key option (Text) that allows the user to move back one screen.

Close—returns to the close/menu exit of the calling application.

More—builds a new set of options in text mode based on the next text bank. If no bank exists, it builds the first bank (text bank = 1).

Displayscreen File

A `displayscreen` record defines the attributes of a screen and provides the user with access to the individual records for options and events. Use the vertical scroll bar to view the entire form.

All ServiceCenter applications using `rio` or `fdisp` panels have been converted with the `display.cv` utility before being shipped to you. Displayscreen records have been created and hard-coded into RAD applications, which search for the appropriate Screen ID when called. Only RAD programmers who write custom applications for their particular enterprise will ever need to create new *displayscreen* records.

The existing *displayscreen* records are powerful tools for controlling the display attributes of certain forms within your system. Knowing which Screen ID is attached to a form is vital if you are to take full advantage of the features offered by the **Display** application.

Showing/Hiding Display Screen Names

Display screen names (and format names) are toggled on and off from the View menu. Select **View>Form Name** to see form names in the lower right of the GUI and **View>Screen Name** to see screen names.

A *screen* is not the same as a *form*. In **Display**, screens are individual records identified by a unique screen ID.

Main structure

This structure defines various conditions of the *displayscreen* record. Users should not edit fields in this structure, since changes to the record will alter when and under what conditions the screen ID is used by the calling application

The screenshot shows the ServiceCenter application window with a table of screen definitions and a detailed form for the selected record.

Screen ID	Enter Key	Format	Title
QBE.display.bad.add	do nothing	\$.format	** Correct or Skip Record **
QBE.display.bad.delete	do nothing	\$.format	** Correct or Skip Record **
QBE.display.gui	do nothing	\$.format	Database

The detailed form for the selected record (QBE.display.bad.add) includes the following fields:

- Screen ID: QBE.display.bad.add
- Language: ENG
- On option 0: do nothing
- Title: ** Correct or Skip Record **
- Format: \$.format
- I/O (If RIQ): true
- Time (If FDISP):
- Window when text and more? (checkbox)
- User Options:
- Initialization Expressions (Once per call to display)
- Initial Message?
- Text of msg:
- Window:
 - Open Window? (checkbox)
 - Parent Name:
 - Window Name:
 - Position: left
 - Percentage:
 - OR
 - # of Lines:

Figure 8-4: Detail of Screen Definition Form

The following are the field definitions for the fields on the *displayscreen* form.

Screen ID—a unique name, identifying the record whose display characteristics are being defined. The names of screen id's generally reflect their function and closely resemble the names of the forms to which they are attached.

Language—display language of the form. The default is English (ENG).

On option 0—determines what action to take when the user presses **Enter** (Option 0). Possible **actions** are:

- **redraw** (refresh) screen
- **do nothing**—returns the user to the current form in the same condition. This option currently works in Text mode only.
- **return to application**—This option forces the **Display** application to return to the calling application. The option is set to **0**, and the action is set to **enter**.

Title—title of the form. The value in this field can be a variable or an expression.

Format—form to display for this screen. The value in this field can be a variable or an expression.

I/O (If RIO)—logical field determining the user’s privileges for modifying the data displayed. This field is used only when the **display** panel is replacing an **rio** panel. The value in this field can be an expression.

Time (If FDISP)—date/time field determining the length of time a user may search the database. This field is used only when the **display** panel is replacing an **fdisp** panel. The value in this field can be an expression.

Window when text and more?—text mode control for displaying additional options. If this box is checked (*true*), and the More option is selected, the system displays all the options in the remaining banks in a single screen.

User Options—flag used by RAD developers to trigger functionality in the Display application. This is an advanced option and should not be changed in the production system.

Initialization tab

Initialization Expressions—sets initial values for variables and fields. These expressions are evaluated before the screen is displayed.

Initial Message?—logical field determining whether or not a message will be displayed to the user. If this field evaluates to *true*, a message is sent just before the screen is displayed.

Text of msg—message as it will appear to the user. The message can be an expression.

Window structure

Control *threading* features in the Window structure.

Open Window?-if *true* (checked), a window is opened. The window is a child of the parent window defined in the **Parent Name** field.

Note: When the **Open Window** option is selected, both the parent and child windows will be visible; however, only the child window will be active for updating.

Parent Name-name of the parent for the new window from the drop-down list. Values for this field can be **NULL** (blank), **Main Window**, **current.window()**, or the name of another window.

Window Name-names the new window. If the field is **NULL** (blank), a window is opened using the name in the **Screen ID** field.

Position-position in which the new window will be displayed. The choices are: *top*, *left*, *right*, and *bottom*.

Percentage-defines the size of the new window based on a percentage of the parent window.

of Lines-sets the number of lines or columns (determined by the value in the **Position** field) in the new window.

Options tab

The Options tab structure provides the user with access to the *displayoption* file.

Click the **Ellipses** button to display a record list of options defined for the current screen ID.


Initializations		Options		Events	
Options					
GUI	Bnk/Opt	Label	Action		
1	1 1	Retry	retry	true	
2	1 2	Skip Record	skiprecord	true	
3	1 3	Cancel	cancel	true	

Figure 8-5: Options Structure of Screen Definition Form

GUI-button control for opening the `displayoptions` record for that option. (see page 247)

Bnk/Opt-indicates the *bank* of options to which the indicated option has been assigned in text mode. (Each bank can contain up to 11 options.)

Label-name of the option as it appears in the form (in the Options menu or as a Service Tray button in GUI mode and as an F-key in Text mode).

Action—token passed back to the main (calling) application telling it how to respond to an action the **Display** application has taken.

Note: *Actions* are specific to each application. Users will only define actions for their own, custom applications or when creating an option that calls their own application.

Condition—logical value from the **Condition** field in the Option Definition file. Options evaluating to *true* are active. This value can be an expression.

Events tab

The Events structure provides the user with access to the *displayevent* file.

Click on the **Ellipses** button to display a list of events defined for the current screen ID.

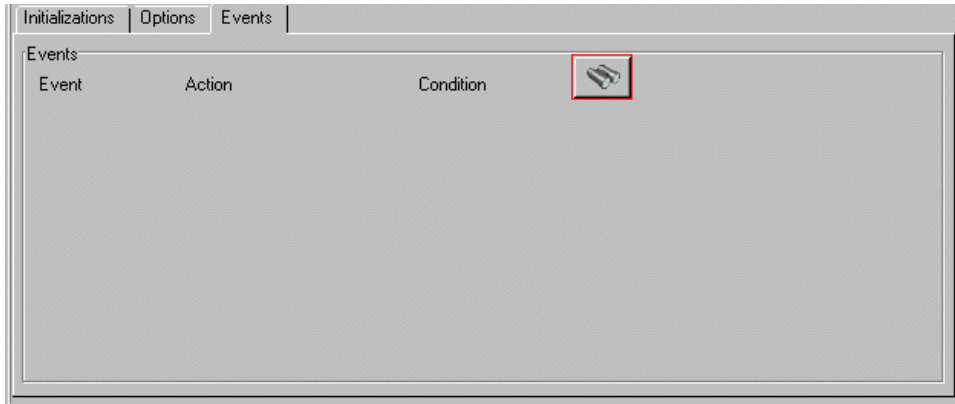


Figure 8-6: Events Structure of Screen Definition Form

Event-buttons that open specific event records.

Action-oken passed back to the main (calling) application telling it how to respond to an action the **Display** application has taken.

Note: *Actions* are specific to each application. Users will only define actions for their own, custom applications or when creating an option that calls their own application.

Condition-logical value from the **Condition** field in the Option Definition file. Options evaluating to *true* are active. This value can be an expression.

Displayoption File

Set the various display options in the **displayoption** file. Display options can appear in the Options menu or as System Tray buttons in GUI mode and as F-keys in Text mode. Option numbers less than (<) 200 in GUI mode appear as buttons. Option numbers greater than (>) 200 appear in the Options menu. RAD applications can also be called from Options Definition records

Access this form from any **displayscreen** record by clicking on the Ellipses button for a record list of all options or by clicking on a numbered option button for individual **displayoption** records (See Figure 8-5 on page 245).

The screenshot shows a window titled "ServiceCenter - [displayoption]" with a menu bar (File, Edit, View, Format, Options, List Options, Window, Help) and a toolbar. Below the toolbar is a table with the following data:

gui.option	txt.bank	txt.option	screen.id	labels.default	labels.txt.alter	action	condition	labels.balloon
1	1	1	QBE.display.b	Retry		retry	true	
2	1	2	QBE.display.b	Skip Record		skiprecord	true	
3	1	3	QBE.display.b	Cancel		cancel	true	

Below the table is the "Display Application Option Definition" form. The form contains the following fields and values:

- Screen ID: QBE.display.bad.add
- Language: ENG
- Action: retry
- Unique ID: 1
- Modifies Record
- Balloon Help (If Option < 200):
- Default Label: Retry
- GUI option: 1
- Text Option: 1
- Bank: 1
- Text Alternative:
- Condition: true
- User Condition:

Below the form is the "RAD" section with "Comments" and "Expressions evaluated after option is selected, but before RAD call" fields. There are also fields for "RAD Application:", "Names:", "Values:", and "Separate Thread?".

At the bottom of the window, the status bar shows: "Selected line is row 1 of 32 records retrieved" and "Response 0.220 draw 0.491 insert displayoption.g(db.view) [US]"

Figure 8-7: Display Options Record

Fields

The following are the definitions of the fields on the *displayoption* form

Screen ID-unique name, identifying the Screen Definition record whose display characteristics are being defined.

Language-display language of the record. The default is English (ENG)

Action-sets the display action.

- **do nothing**: Display does not return to the calling application, but returns to the **rio** or **fdisp** panels, leaving the screen intact. If you have scrolled down in a data field, your position is maintained, and the cursor remains in the field.
- **redraw**: Refreshes the screen and returns to the **rio** or **fdisp** panels. The cursor remains in the same field, but scrolling is reset to the top of the field.
- **back**: Returns to the back exit of the calling application.
- **close**: Returns to the close/menu exit of the calling application.
- **more**: In Text mode, this action builds a new set of options based on the next text bank. If no bank exists, it builds the first bank (text bank = 1)
- In GUI mode, selecting this action has the same effect as selecting **do nothing**.
- **NULL**: If the action field is left blank, the system assigns a value of NULL to the **Action** field.
- **UNKNOWN**: If no option can be found with a true condition, the system assigns a value of UNKNOWN to the **Action** field.

This only occurs if the condition for the option has changed to *false* as a result of the user's input on the screen actions.

Unique ID-The unique ID is a system generated unique identifier. You can assign a number, but if it is not unique the system will override it and assign a new number.

Modifies Record-This field signifies that using this display option may make changes to the record. The display routine will recognize this, and lock the record if necessary before running the option.

GUI Option-number of this option in the GUI mode, taken from the Screen Definition form. Options with numbers greater than 200 appear in the Options menu. Options with numbers less than 200 appear as buttons in the System Tray.

Text Option-number of this option in the Text mode, taken from the Screen Definition form.

Bank-determines the set of function keys with which this option is grouped. This field is required only when 11 or more options appear on a form. Option 12 is reserved for the **More** button.

Default Label-defines the default label for the option being created. This is a required field for GUI mode.

Balloon Help-provides balloon help for the option button being defined (GUI mode). Instructions may be given in any language.

Text Alternative-if not NULL, determines the label used for the F-key in a Text form.

Condition-logical field defining the condition for the option. If the field evaluates to *true*, the option appears to the user. An expression may also be used.

User Condition—logical field defining a user-specific condition for the option. If the field evaluates to *true*, the option appears to the user. An expression may also be used to further tailor the expression used in the previous field for user-specific access control.

RAD tab

Expressions...-expressions evaluated when this option is selected by the user, but before the RAD application is called.

RAD Application-names a RAD application to be called when the option is selected.

Separate Thread-determines whether or not the RAD application is started in its own *thread*.

Note: *Threading* allows for several active windows to be open for editing at once.

Names-identifies the parameters being passed to the RAD application.

Values-defines the value of each parameter being passed to the RAD application.

Post RAD Expressions-expressions evaluated when this option is selected by the user. This is done after a RAD application is called.

Comments tab

Comments-allows developers to provide additional commentary on this displayoption for later development.

Database Dictionary keys

These keys control how the data is placed and retrieved from the file. Use values based on these keys when issuing queries on the displayoption file.

Field	Key	Description
unique.id	unique	Ensures record numbers are unique
screen.id language gui.option gui.sig	unique	This key ensures that no two GUI options can have the same screen ID, language, or condition.
screen.id language txt.bank txt.option txt.sig	unique	This key ensures that no two text options can have the same screen ID, language, text bank, or condition.
action screen.id language gui.option	nulls&duplicates	Useful when querying by action.

Displayevent File

The `displayevent` file defines the *events* a screen will handle. Access the form by clicking an Event button in a *displayscreen* record. (See *Events tab* on page 246).

The screenshot shows the 'ServiceCenter - [displayevent]' application window. At the top, there is a menu bar (File, Edit, View, Format, Options, List Options, Window, Help) and a toolbar with icons for OK, Cancel, Previous, Next, Add, Save, and Delete. Below the toolbar is a table with columns: Screen ID, Event, Actions, and Condition. The table contains three rows, with the second row highlighted in blue.

Screen ID	Event	Actions	Condition
cm3t.update	32103	eventreturn	\$.L.event.edit.flag
icm.display.joinfile	32103		\$.L.mode="display" and evaluate(update in \$.L.env) and \$.L.addflag=false and index(type in \$.L.dev, \$.icm.type)
cc.first	32767	newcall	event.name()!="transfer" "conference"
cc.first	32767	transfer	event.name()="transfer"

Below the table is the 'Display Application Event Definition' form. It contains the following fields and values:

- Unique ID: 8
- Screen ID: icm.display.joinfile
- Language: ENG
- Event Name/Number: OnFormModified
- Condition: \$.L.mode="display" and evaluate(update in \$.L.env) and \$.L.addflag=false and index(type in \$.L.dev,
- User Condition:
- Expressions evaluated after event but before RAD call:
- RAD Application: icm.event.lock
- Separate Thread?: false
- Names: file
- Values: \$.L.dev
- Post RAD Expressions:


```
$.L.ds.action=$EVENT
if ($.L.ds.action=("eventreturn", "reselect")) then (1 in $.L.messages="locked";$.L.mode="update")
```

At the bottom of the window, it says 'Selected line is row 5 of 11 records' and 'Response 0.161 draw 0.100 insert displayevent.g(db.view) [US]'.

Figure 8-8: Event Definition Screen

Fields

The following are the field definitions for fields on the `displayevent` form:

Unique ID-unique number identifying this record from others which may also be associated with the same screen id.

Screen ID-unique name, identifying the record whose display characteristics are being defined.

Language-display language of the record. The default is English (ENG)

Actions-indicates what kind of *action* to perform. *Redraw* is a reserved action. If this action is requested, **Display** does not return to the calling application, but to the **rio** or **fdisp** panel instead.

Conditions-logical field controlling the action specified. If this field evaluates to *true*, the action will be available to the user.

Event Name/Number-the Event Name/Number fields are used to define when events will occur. Note that the **OnFocusIn**, **OnFocusOut**, and **OnNewValue** fields mean that an event will trigger any time an action occurs on the associated format. This could affect system performance.

- **OnFocusIn** – Each time an object (combo box, radio button, and so on) receives focus this event will be sent. The **OnFocusIn** field can be used with the `cursor.field` function for example to trigger an event when a field receives focus.
- **OnFocusOut** – When focus leaves an object (combo box, radio button, and so on) this event will be sent.
- **OnFormModified** – When a user modifies any field in a record, the system knows that a record is being updated. For example, this definition can be used to place a lock on a record once the form is modified.
- **OnNewValue** – If a value in a specified field changes, then this event will trigger.

Condition-string equivalent of the condition expression. This field is required to allow the unique key to work properly. Because the condition is evaluated by the form, the contents must be converted back to a string before adding the field to the key.

User Condition-condition statement that overrides the value in the **Condition** field. Use this field to modify the conditions under which the events will be handled rather than editing the statement in the **Condition** field.

Expressions evaluated after event but before RAD call—expressions performing calculations or reassigning a global variable to a local variable. These expressions are evaluated after the event has occurred and before a RAD application is called.

RAD Application—name of the RAD application called when the defined event is selected. This occurs after the expressions are evaluated.

Separate Thread?—logical field controlling the appearance of a threaded screen. If this field evaluates to *true*, the calling application is started in its own thread. A condition can also be used here for text-mode operations.

Names—identifies the parameters being passed to the RAD application.

Values—defines the value of each parameter being passed to the RAD application.

Database Dictionary keys

Field	Key	Description
unique.id	unique	Ensures the record numbers are unique
screen.id language event event.sig	unique	This key ensures that no two Events records can have the same screen ID, language, event.name, object.name, or condition.
actions screen.id language event	nulls&duplicates	Useful when querying by action.

Creating a displayscreen record

Displayscreen records have been created and hard-coded into RAD applications, which search for the appropriate Screen ID when called. Only RAD programmers who write custom applications for their particular enterprise will ever need to create new *displayscreen* records.

The displayscreen record is the control point for applying options to ServiceCenter forms using the Display application. You must create the displayscreen record first, then create the individual options in the *displayoption* file.

To create a displayscreen record:

- 1 Access the *displayscreen* file using one of the following methods:
 - Type `ds` in the ServiceCenter command line and press Enter.
 - Type `displayscreen` in the **File** field of the Database Manager dialog box and click Search or press Enter.
 - Click the Display Screens button in the Tools menu of the Utilities tab.
- 2 Enter a unique name in the **Screen ID** field of the blank displayscreen record form.

This name should identify the file involved and describe the process to which the options are attached. For example, a displayscreen record created for a script that opens incident tickets might be called *pm.script.open*.
- 3 In the **Format** field, enter the name of the form on which the new options should appear or use the appropriate format variable (see *Format variable* on page 271) if you want the display options to appear on more than one form.
- 4 Complete any other appropriate fields.

For definitions of display screen fields, refer to *Displayscreen File* on page 241.
- 5 Click Add.

Procedures for creating displayoptions

To create a displayoption record:

- 1 Access the *displayoption* file using one of the following methods:
 - Type `do` in the ServiceCenter command line and press Enter.
 - Type `displayoption` in the **File** field of the Database Manager dialog box and click Search or press Enter.

- Click the Display Options button in the Tools menu of the Utilities tab.
- 2 Enter the name of the screen ID you created for the displayscreen record in the **Screen ID** field of the blank displayoption record form.
 - 3 Enter the appropriate action for your option in the **Action** field (*Display actions* on page 272).

Complete the displayoption record using the procedures found in *Displayoption File* on page 247.

Important: If you create a displayoption by copying an existing record, you must change the screen ID and leave the Unique ID field blank; the system will assign a unique ID to your option.

If you are defining a Fill option, be sure that the value in the **GUI option** field matches the **Button ID** value of the field in the form.

- 4 Click **Add** to add the record to the database.
- Open the displayscreen record for your screen ID and select the **Options** tab to make sure your new option appears there.

Restricting Access

As a system administrator, you may wish to restrict users' access to certain display options based on their system rights. This is accomplished by creating an expression in a **displayoption** record that determines who may use the option. This expression is constructed with *capability words* found in a user's operator record. You may use existing capability words or create new ones that more narrowly define the restrictions.

To restrict access to a display option, you must:

- Identify the option and the group to whom you want to deny access.
- Define a capability word unique to the option.
- Add your capability word to the operator records of all users to whom you wish to *grant* access to the option.
- Create a condition expression in the displayoption record using your new capability word.

Selecting the option

Select an option that performs an important function in your system.

- Is this option part of a broader functionality?
- Should all users with rights to this functionality have access to this option?

For example, should all users with access to Incident Tickets in Incident Management be able to close a ticket?

By using a condition expression in a displayoption record, a system administrator can limit a user's right to close tickets without restricting rights to the other features of Incident Management or incident ticket options.

In the following example, you restrict access to the **Close** button in the system tray of Incident Management incident tickets to all users except those with *SecClose* capabilities.

Users with *SysAdmin* capability will be able to view, update and save existing incident tickets, but will not be allowed to close tickets.

Figure 8-9: Incident Ticket Update form

Defining a capability word

Select capability words that reflect the access rights they define. For the purpose of this example use `SecClose` to grant access to the `Close` button in the incident ticket (`apm.<category>.update.g`) system tray.

Use the following procedure to define the capability words:

- 1 Select the **Utilities** tab in the home menu.
- 2 Click **Administration**.
- 3 Click **Capability Words** in the *Security* structure.
An empty capability definition form (`capability.g`) is displayed.
- 4 Enter `SecClose` in the **Capability** field of the form.
- 5 Enter a description of the capability's function in the **Description** box.
- 6 Click **Add**.

Display conditions

Access to an option is controlled by the **Condition** field in a displayoption record. This field may evaluate to *true* or contain an expression defining a condition for display.

- 1 Return to the **Utilities** tab.
- 2 Click **Tools**.
- 3 Click **Display Options**.

The displayoption form appears.

- 4 Enter the screen ID of the update form, *apm.edit.problem*, and press **Enter**.
A record list of all display options assigned to this form is displayed.
- 5 Locate the **Close** option from the record list.

Note: The **User Condition** field is empty by default. The User Condition is processed after the **Conditions** field and operates at the user-specific level.

The screenshot shows the ServiceCenter application window with a table of application options and a detailed form for the 'Close' option.

GUI	Bank	Option	Screen ID	Default Label	Text Label	Action	Condition
1	1	1	apm.edit.problem	Reopen		reopen	evaluate(reopen in \$G.pm.e
2	1	2	apm.edit.problem	OK		ok	gui() and evaluate(update in
3	1	3	apm.edit.problem	Cancel		back	true
4	1	4	apm.edit.problem	Save		save	evaluate(update in \$G.pm.e
5	1	5	apm.edit.problem	Undo		reselect	true
6	1	7	apm.edit.problem	Close		close	status in \$L.filed~="closed"
6	1	7	apm.edit.problem	Resolve		resolve	status in \$L.filed~="close"

Display Application Option Definition

Screen ID: Language: Action:

Unique ID: Modifies Record back, close, and more are special

GUI option: Balloon Help (If Option < 200):

Text Option: Default Label:

Bank: Text Alternative:

Condition:

User Condition:

RAD | Comments

Expressions evaluated after option is selected, but before RAD call

RAD Application: Separate Thread?

Names: Values:

Post RAD Expressions

Selected line is row 6 of 32 records retrieved Response 0.140 draw 0.150 insert displayoption.g(db.view) [US]

Figure 8-10: Default appearance of Close option definition on incident update form.

- 6 Add the following condition statement to the User Conditions field:

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

Where *SecClose* is the name of the capability word you created in the previous section.

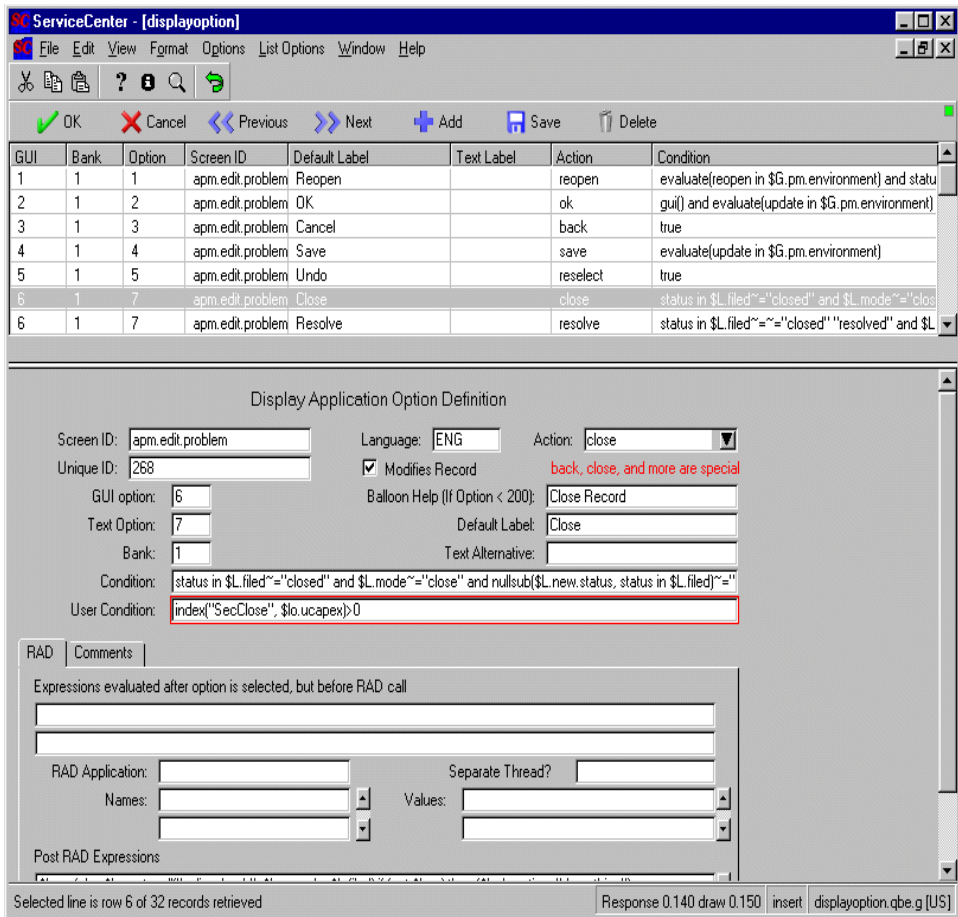


Figure 8-11: Modified Close option definition requiring SecClose capability word

- 7 Click **Save** to record the modification to the record.
- 8 Click **OK** to back out, and return to the home menu.

Adding operator capability

- 1 Return to the **Utilities** tab on the home menu.
- 2 Click **Administration**.
- 3 Click **Operators** in the *Security* structure.
An empty operator definition form, *operator.g*, opens.
- 4 Enter the name of an operator. *FALCON* is used in this example.

Note: Remember, not all operators have Incident Management access.

5 Press Enter.

The selected operator record appears.

6 Select the *Startup* tab.

7 Add the new capability word in the next available slot in the **Execute Capabilities** array.

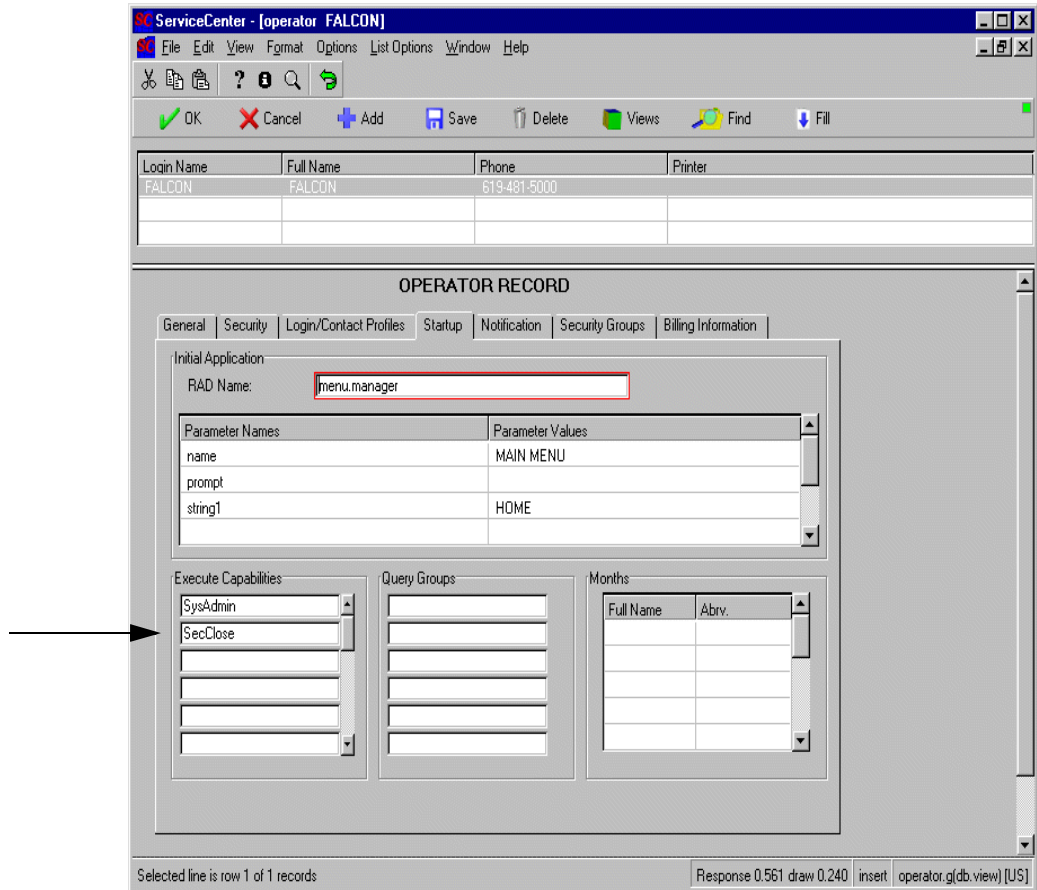


Figure 8-12: Adding a capability word to an operator record

8 Click Save to record this modification.

9 Click OK to exit and return to the home menu.

10 Log the client out of ServiceCenter.

Testing the display option

- 1 Log back into ServiceCenter as the operator to whose record you added the capability word *SecClose*.
- 2 Open the Incident Management Queue and view an incident ticket.
The Close button should appear.

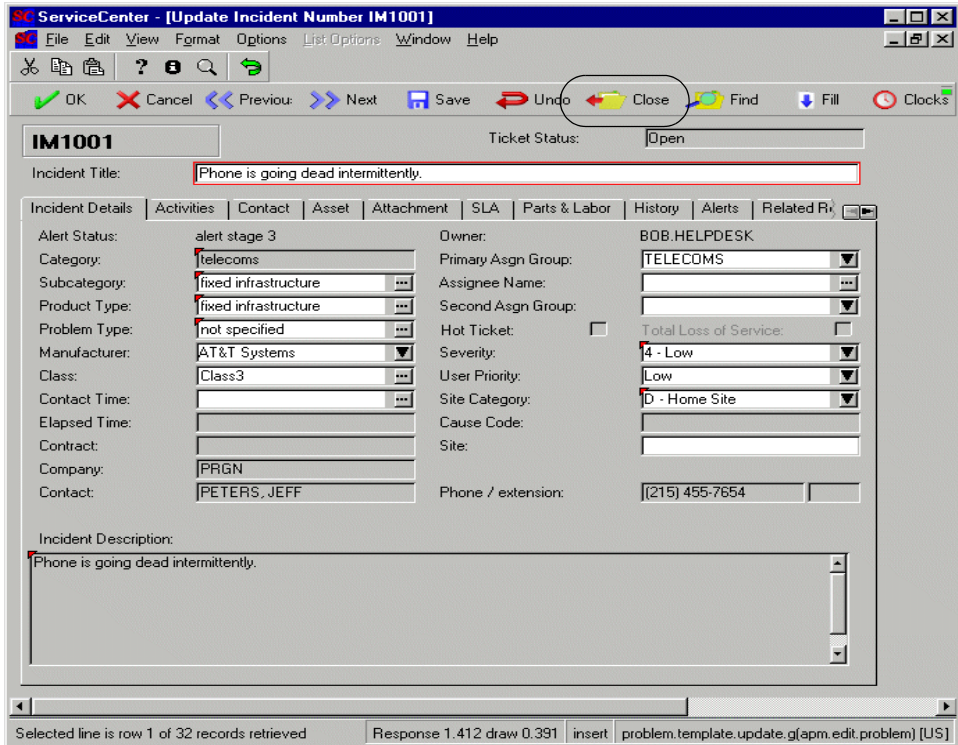
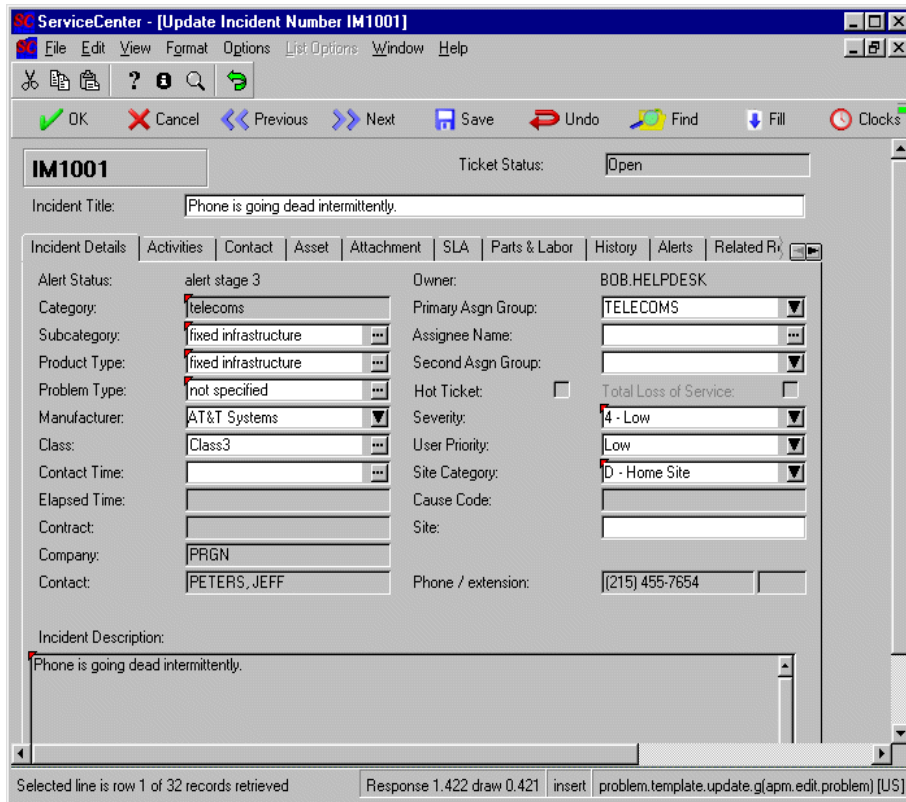


Figure 8-13: Option that appears to operator with SecClose capability

- 3 Open a new client as an operator who does not have the capability word defined in his or her operator record.
- 4 Go to the Incident Management Queue and view the same ticket as you did above.

The Close button should not appear for the operator whose record did not contain the capability word.



Close button does not appear for this user.

Figure 8-14: System tray for operator without SecClose capability

Calling an Application

Display panels allow the system administrator to set up a series of actions that otherwise would have to be hard-coded into each individual application. Among the added functionality is the ability to call separate applications from within the current application.

An example of this is calling the **build.list** application from a combo box on the *problem.initial* form.

The call to **build.list** recreates a list of viable subcategory, location, or other options, based on the information already appearing on the *problem.initial* form.

When calling an application, several options are available, including threaded windows and scripting.

Threading

Display allows you to control the appearance of other forms in separate *threads*—individual sequences of screens independent from the forms that launched them. Threading makes it possible to edit within different independent screens quickly and without excessive navigation.

Threading within Display is controlled by the logical field, **Separate Thread?**, in the **displayoption** record. When this field evaluates to *true*, the application launched from that record appears in a separate sequence of windows.

In the following example, when the **apm.show.cost** application is called, it appears in a threaded window from the *apm.edit.problem* screen.

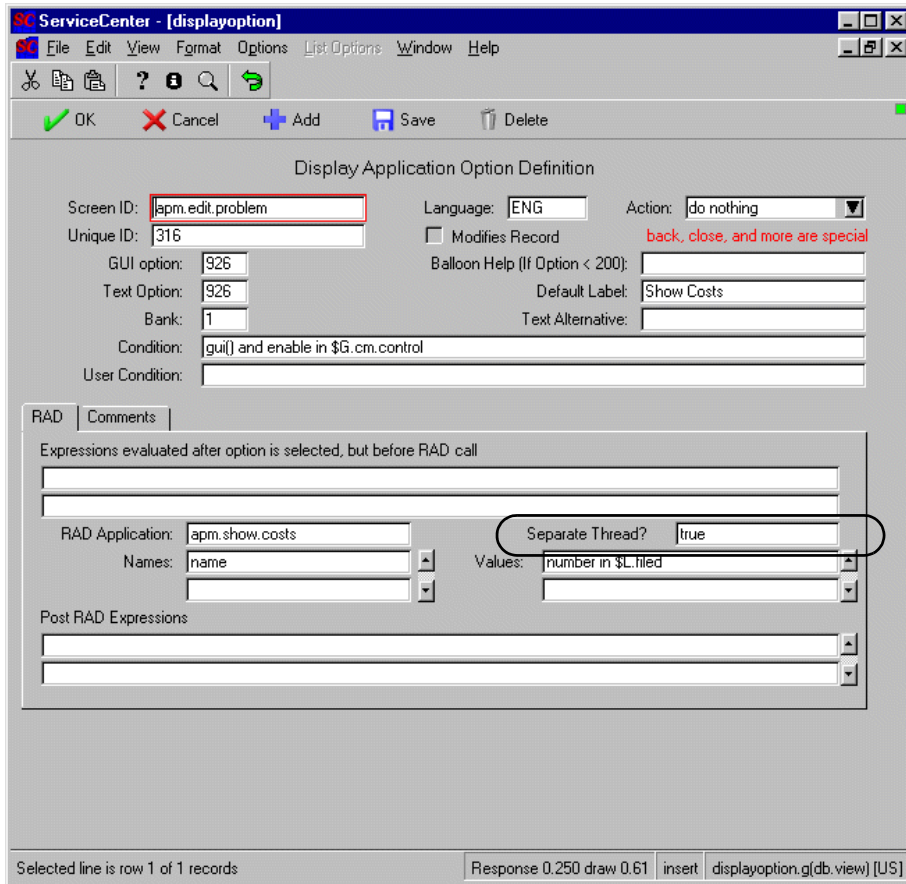


Figure 8-15: Example of opening an application in a threaded window

When **Back** is clicked or the `apm.show.cost` application is closed from this threaded window, the `apm.edit.problem` screen is redisplayed in the same frame.

Threading is often used on conjunction with other Display actions, e.g., calling external applications (as shown above) and scripts.

Scripts

A script is a series of instructions run from an option. These may include a progression of screens, requesting information or user-specific responses, or instructions to perform certain computations with existing record data. Since the script call is setup and performed at the displayoptions application level, it does not need to be hard-coded into the RAD of the module. This allows for easier maintenance and modification.

Script display calls often employ the `script.execute` application, specifying the progression of forms to appear for the user. An example of this form of display call is seen in ServiceCenter Request Management, where the **New** button option forces the user through a script at open time. The script calls a preliminary information screen which requests certain data that is later populated into a new Request Management record. The screen displayed serves as an organized and user-specific entrance point into the application.

ServiceCenter - [displayoption]

File Edit View Format Options List Options Window Help

OK Cancel Add Save Delete

GUI	Bank	Option	Screen ID	Default Label	Text Label	Action	Condition
604	4	4	apm.edit.proble	Service Evaluation		evaluation	false

Display Application Option Definition

Screen ID: Language: Action:

Unique ID: Modifies Record back, close, and more are special

GUI option: Balloon Help (If Option < 200):

Text Option: Default Label:

Bank: Text Alternative:

Condition:

User Condition:

RAD Comments

Expressions evaluated after option is selected, but before RAD call

RAD Application: Separate Thread?

Names: Values:

Post RAD Expressions

Selected line is row 1 of 1 records Response 0.290 draw 0.80 insert displayoption.g(db.view) [US]

Script display calls often employ the **script.execute** application.

Figure 8-16: Calling a script of instructions from a display option

Note: In the example shown in Figure 8-16 on page 267, the called script appears through a threaded window, as indicated by the *true* value in the **Separate Thread?** field.

Creating Displayscreen Records: Advanced Topics

Your ServiceCenter system contains a number of displayscreen records that have been predefined in the RAD code. Some of these records bind their options to a specific form, while others bind the options to local variables for use with multiple forms. If you edit an option in a displayscreen record used by more than one form, every form that uses that Screen ID is affected. If you need option combinations that do not exist in the displayscreen records provided with the standard system and do not wish to edit any of the existing Screen IDs, you must create your own unique displayscreen record. You might choose to do this when:

- Writing custom RAD applications
- Creating script forms

Note: Creating displayscreen records is an advanced procedure requiring access to the RAD editor and a knowledge of RAD.

Custom RAD

Determine whether your display screen record should apply to a single form or be available to all the forms within your application. If applying to a single form, consider using Format Control. If you want to apply the display options to a single form, enter the name of that form in the **Format** field in the displayscreen record. If you want the same display options to be available on multiple forms, you must bind the **Format** field to the appropriate local variable. This is the variable you have defined in the **Array of local variables** field of the **display** panel in your RAD application.

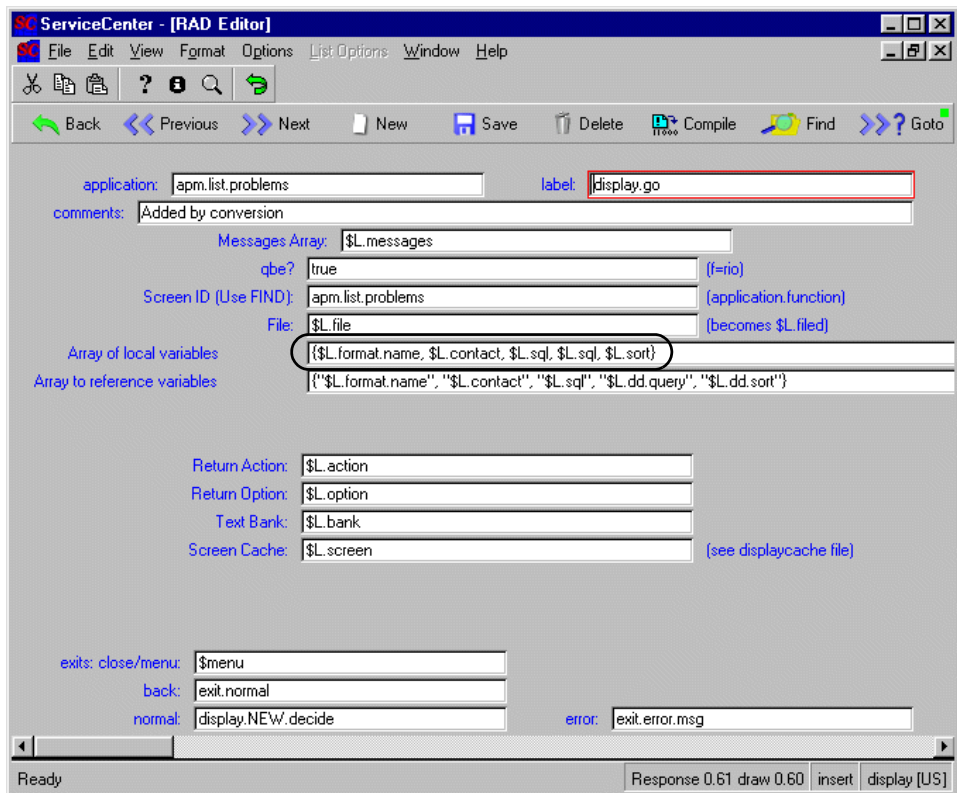


Figure 8-17: Display panel defining a local variable

When you are creating your displayoptions, the values you enter in the **Action** field of the displayoption record (Figure 8-20 on page 272) must be defined on a **decision** panel in your RAD application as `$L.action=<action type>` (Figure 8-18 on page 270).

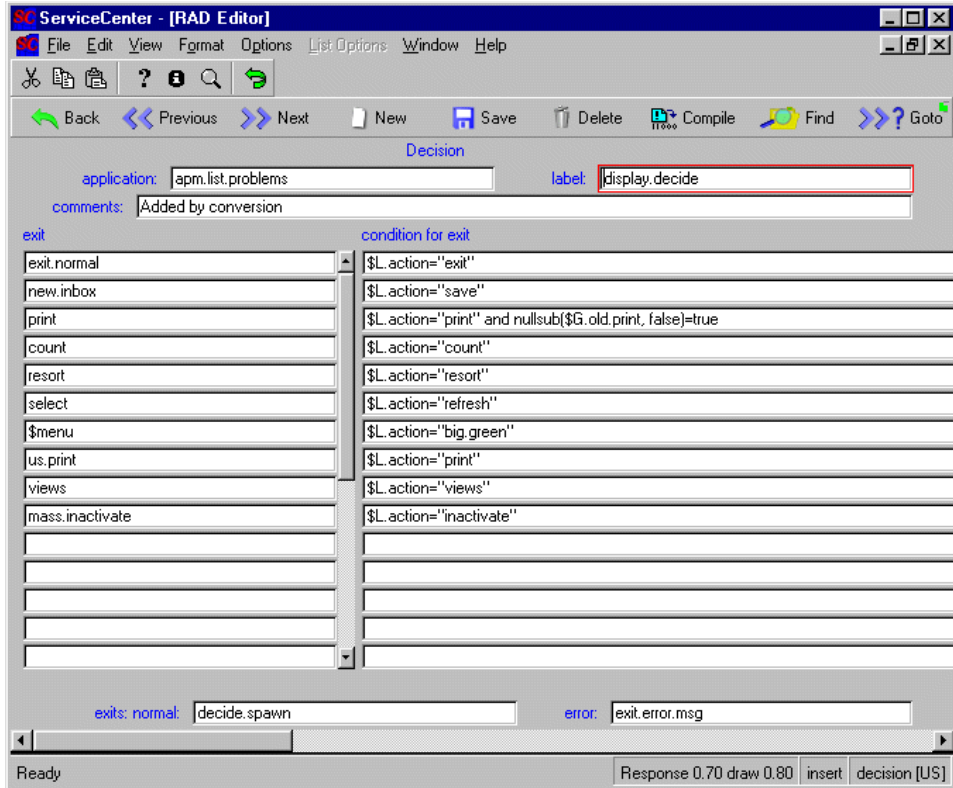


Figure 8-18: Decision panel defining displayoption actions

Script forms

You can create unique displayoptions for your script forms that may be unrelated to those appearing in the application that executes the script. Changes to the RAD code are unnecessary when creating displayscreen records for this purpose; the appropriate variables have been hardcoded into the `script.execute` application and need only be applied correctly.

Format variable

The *format variable* for `script.execute` is `$.script.format` and is hardcoded into the RAD layer of the application. Enter this variable into the **Format** field of any displayscreen record to bind its options to multiple script forms. This local variable is bound to the display form you entered in the **Format** field on the script definition record.

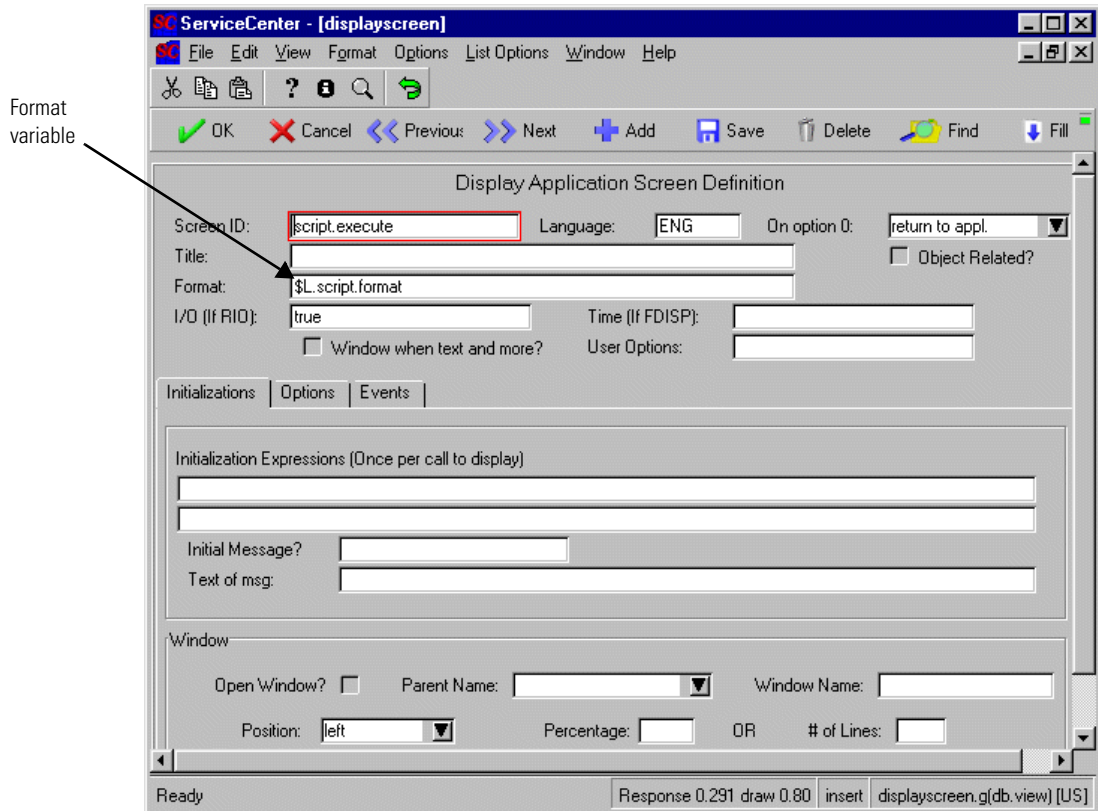


Figure 8-19: Displayscreen record using a format variable

If you want to apply unique options to a single form in a script, simply name the form in the **Format** field of the displayscreen record rather than use the local variable.

Display actions

Actions for displayoptions in scripts are defined on a **decision** panel in the RAD code of the `script.execute` application (Figure 8-18 on page 270). Unless you possess a RAD license and are capable of programming in RAD, you are limited to the following, hardcoded actions:

do nothing back close more redraw

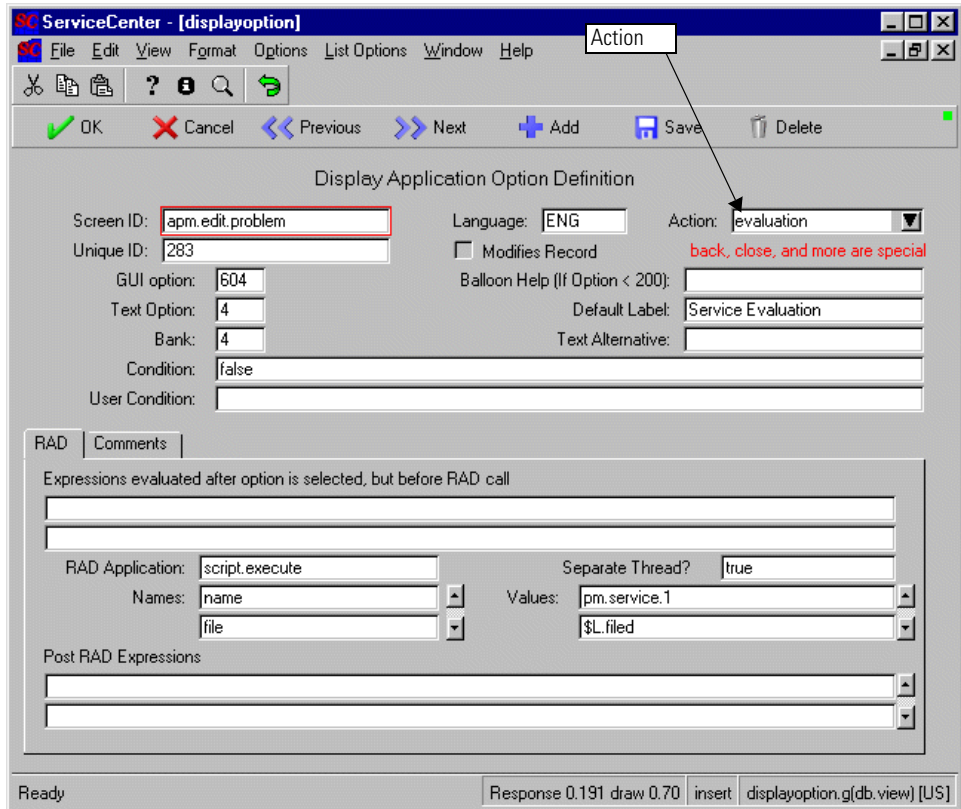


Figure 8-20: Displayoption record defining a Fill option

For instructions on accessing displayscreen records, refer to *Accessing Display Records* on page 239.

9 Advanced Operations

CHAPTER

Copying a Database Dictionary

You can make copies of your Database Dictionary record for backup purposes or to create another file with very similar dictionary definitions.

When you copy a Database Dictionary record, you have the option to copy the record only or the record and all individual data records contained in the file. If you copy the Database Dictionary record and all associated records, data in the original file is inaccessible during the copy process. The amount of time it takes to copy the file and records will vary depending on how many records are in the original file.

Copying Database Dictionary record only

The following illustrates how to make a copy of the Database Dictionary record in the *enduser* file distributed in the Inventory/Configuration application. In this example, we will copy only the Database Dictionary record, not the data records.

To copy a dbdict record:

- 1 Open the *enduser* record through Database Dictionary.
- 2 Select **Options > Copy/Rename**.
- 3 Enter *enduser1* in the **New Name** field of the rename screen.

Note: File names must be one word; no blanks and no special delimiters are allowed.

- 4 Select the **Copy dbdict only** radio button.

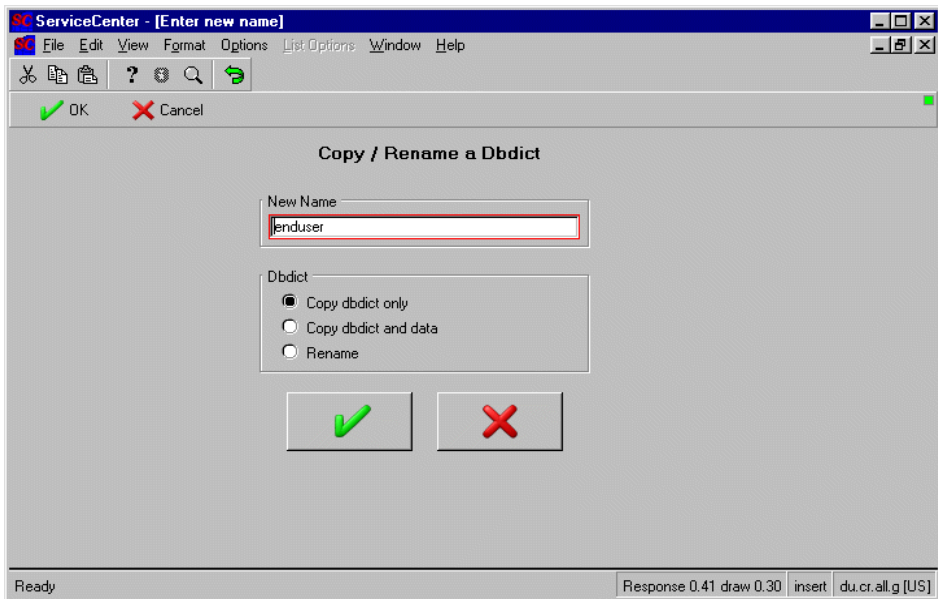


Figure 9-1: Copying the data structure record only

- 5 Click **OK** to begin file copy.

You are returned to the copied record. The message *Dbdict successfully copied* appears at the bottom of the screen.

- 6 You may now make changes to the *enduser1* copy of the *enduser* Database Dictionary record.

- 7 When finished with the record, click OK to exit.

Note: You are required to update/regenerate your Database Dictionary record only if you have added, deleted, or modified fields and/or keys.

Copying Database Dictionary record and data records

The following illustrates how to make an online backup copy of all the records in the *enduser* file by creating a new file with the same Database Dictionary structure and containing all records contained in the original file. Whenever you want to copy records from a file by copying the entire file, you must copy the associated Database Dictionary record as well.

To copy dbdict and data records:

- 1 Select Options > Copy/Rename.
- 2 In the New Name field, enter *enduserbackup*. (Remember, file names must be one word. No blanks and no special delimiters are allowed.)
- 3 Select the Copy dbdict and data option to copy all records and the Database Dictionary record from the *enduser* file to the *enduserbackup* file.
- 4 Click OK to copy the record and data stored in the file.

Note: The length of time it takes to copy a file with all records will vary depending on how many records exist in the file. It is recommended that copying a file with all associated records be performed during non-peak operation times.

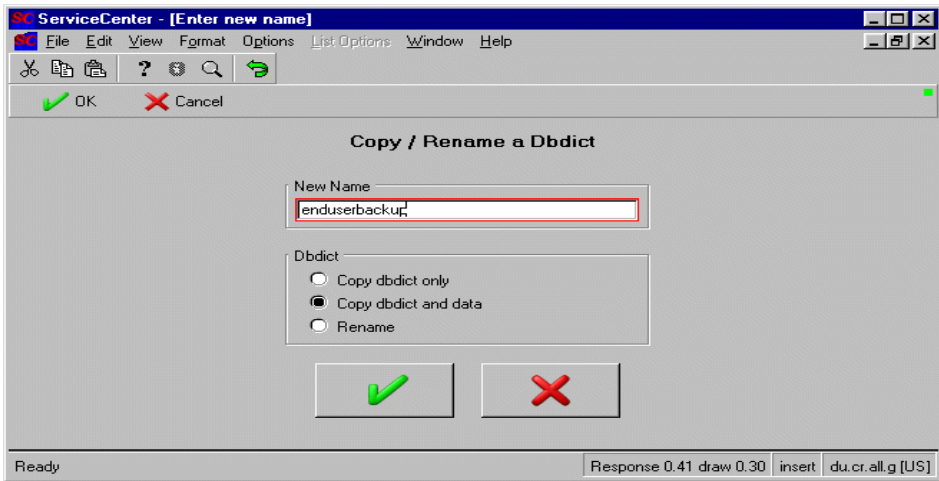


Figure 9-2: Making backup of current record and data

The Database Dictionary record for the *enduserbackup* file is displayed and the message **Dbdict successfully copied* appears at the bottom of the screen.

Note: When an asterisk displays in front of a message, it means there are additional messages to display which may have relevant information regarding this operation.

To view all messages associated with this operation, click the **i** (Messages) button just below the top menu bar.

This displays a small message window containing any additional messages.

You can now make changes to the *enduserbackup* Database Dictionary record.

- 5 Click **OK** to exit when you have completed modifications to this file.

Note: You will be required to update your Database Dictionary record only if you have added, deleted, or modified fields and/or keys.

Renaming a Database Dictionary Record

After you have created a Database Dictionary record, you may decide that you want to change the name of the file, or you might want to change the name of one or more of the files sent with the base system. For example, you might want to rename the *communications* file to *multiplexors*. This would change the file name but retain all of the field and key definitions in the dictionary.

If records exist in the file, renaming the Database Dictionary record will associate all of the records with the new dictionary name. This differs from copying a file because it takes only a few seconds to perform this function.

The following illustrates how to change the name of the *communications* file to *multiplexors*.

- 1 Open the *communications* Database Dictionary record.
- 2 Select Options > Copy/Rename.
The Copy/Rename dialog box is displayed.
- 3 In the **New Name** field, enter *multiplexors*. (Remember, file names must be one word. No blanks and no special delimiters are allowed.)
- 4 Select the **Rename** option to rename the file to *multiplexors*.
- 5 Click OK.
The renamed Database Dictionary record is displayed with the following message in the status bar: *Dbdict successfully renamed*.
- 6 Click OK to exit the record and save the file with the new name.

Deleting a Database Dictionary Record

If you decide that you no longer need a file, you can delete the Database Dictionary record.

Warning: If you delete a Database Dictionary, all records contained in the file are also deleted.

The following illustrates how to delete a Database Dictionary record called *endusercopy*.

- 1 Open the Database Dictionary record you wish to delete.
- 2 Click **Delete**.
This operation requires a confirmation, and a delete file confirmation window appears.
- 3 Click **Yes** to delete the record and all data associated with the *endusercopy* file.

Note: The amount of time it takes to perform the delete will vary depending on how many records are in the file. It is recommended that deleting a file with a large number of records be performed during non-peak operating times.

After the file is deleted, you are returned to the Database Dictionary prompt and the message *endusercopy file deleted* is displayed on the bottom of your screen.

- 4 Press **Back** to exit the Database Dictionary prompt.

Resetting the Database and All Records

If you decide the records in a file are no longer needed, but the file structure is to be retained for later use, you can perform a file reset.

The **Reset** option deletes all data records from a Database Dictionary file. In most instances, once a system has been set up and is in operation, this option will not be used. Most often it is utilized when clearing test data from a pre-implemented system or test file.

Warning: This option is only used when the intention is to permanently remove data file records. Once a reset has been performed, the records cannot be retrieved.

The following illustrates how to reset a Database Dictionary record called *endusercopy2*.

- 1 Open the Database Dictionary record you wish to reset.
- 2 Select Options > Reset.

Note: This option does not appear for files that contain no data records.

You are prompted to confirm this reset action.

- 3 Click **OK** to confirm this reset and erase all records in this file.
- 4 Click **Cancel** to quit and return to the Database Dictionary record.
- 5 Click **Schedule** (clock) to schedule the reset to perform at a designated time.

If you clicked **OK**, the reset performs immediately in the foreground. When completed, a message appears at the bottom of the screen confirming the time/date of the file reset and the removal of all records.

The asterisk (*) at the beginning of the message indicates there are additional messages generated and available to read in relation to this operation.

- 6 Click **i** in the Tool Tray to view these messages.
- 7 Read the messages and identify if any problems or errors occurred during the operation.

Scheduling the reset

If you chose to schedule the reset operation at step 5 on page 279 above, you are prompted with a schedule window.

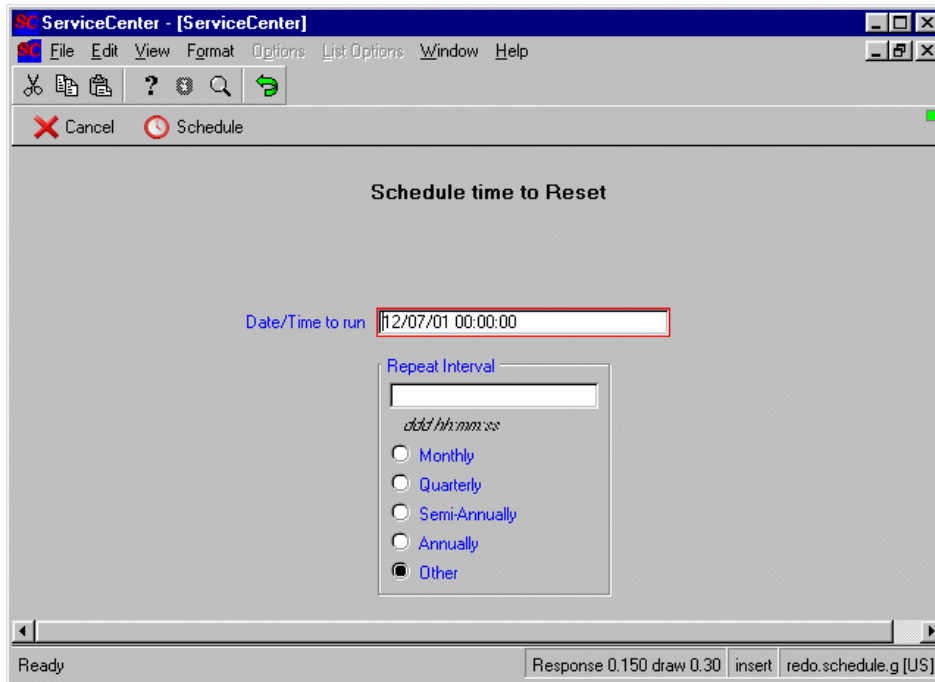


Figure 9-3: Scheduling file resetting action

- 1 Set the date and time for the operation to run in the **Date/Time** field.
- 2 If you want this operation to run periodically, flushing out data records from this file regularly, provide the appropriate date and time for the interval in the **Repeat Interval** field.
- 3 Click **Schedule** (the clock) to confirm the schedule and activate it.

You are returned to the Database Dictionary file which will be reset.

Information Pooling

ServiceCenter supports a multiple data-storage pool structure consisting of several 2-gigabyte pools. These pools provide a way of separating data into multiple Windows NT, UNIX (or MVS) files.

There are two types of pools; data and index. Every Database Dictionary file has one of each of these information files. The data pool stores the data records for that file. The index pool stores the index and relationships of fields in the Database Dictionary.

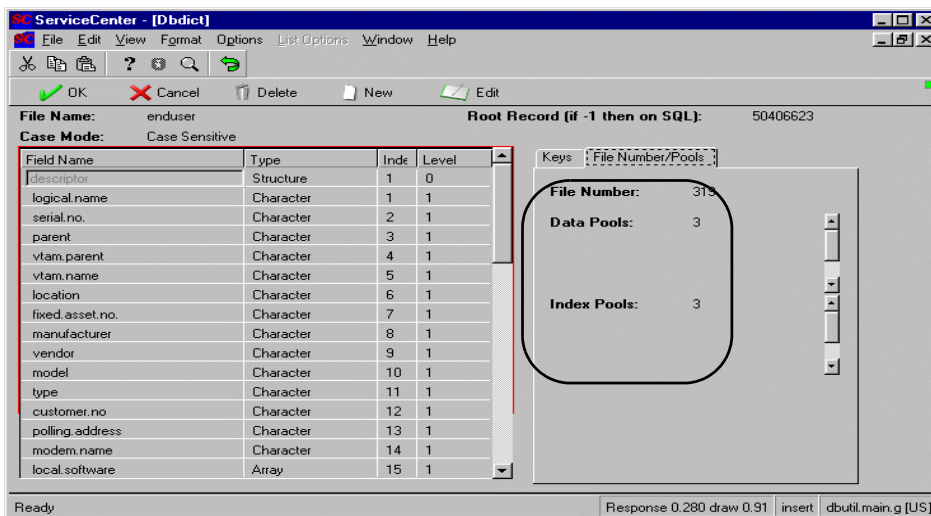


Figure 9-4: Data and index pools

The primary pool is pool 3, the SCDB.DB1 file. New pools can be created (4 = scdb.db2, 5 = scdb.db3, etc.). For more information and specific details on P4 pool operations see the *P4 Filesystem* section of the *Data Administration Guide for more information*.

When a new Database Dictionary file is created, the default data and index pool values are set to 3. Once more data and index pools have been created, file information can be moved to different pools. Index and data information can also be stored in different pools from one another, to allow greater control over the size and use of each data pool. See the *P4 Filesystem* section of the *Data Administration Guide for more information*.

Modifying pool settings

- 1 To display the information pool modification window, select **Options > Move To New Pool**. A dialog box appears.

The dialog provides drop-down menus to specify the location and pool number for data and index information.

- 2 Select a new value from the list to modify the settings.
- 3 Click **OK** to confirm the new settings.

You are prompted to confirm the new settings and move the data accordingly.

- 4 Click **Yes** to confirm the action.

See the *P4 Filesystem* section of the *Data Administration Guide* for more information on *Data Policy and Information Pool* settings.

Searching for a Field

The ability to search the dictionary for a specific field name can be very useful when dealing with a large database dictionary. The following example illustrates how to search for a field.

To search for a database field:

- 1 From within the Database Dictionary record you want to search, select **Options > Field Search**.

This brings up the Field Search window prompt.

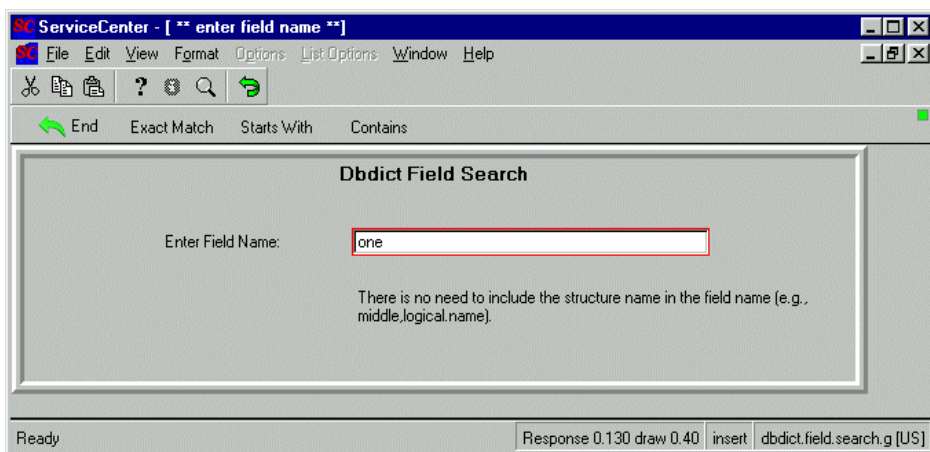


Figure 9-5: Fieldname search capability

- 2 Enter the field name to search, for example, *one*.
- 3 Click **Exact match** if you are sure that what you entered is a field in the dictionary record.
- 4 Click **Starts with** to find all fields starting with the string entered.
- 5 Click **Contains** to search for fields containing the string entered anywhere in the field name. (Used in this example).
- 6 Click **End** to cancel the search.

If the system finds a match, it responds by giving a message similar to the following: **Field "phone" found: type=character; level=3; index=2*

The message gives the *field type*, the *level* (how many structures down the field is located), and the *index* (location within its parent structure). For example, *one* appears in:

- a *character* type field (**phone**)
 - in the third level structure (**alternate.contact**)
 - as the second field in the **alternate.contact** structure.
- 7 If multiple fields are found by the search, multiple messages are returned, one message for each field found. The message in the above example displays an asterisk (*), indicating it does have additional messages to display.
- 8 To reveal all messages from this operation, click **i** in the tool tray.

This displays a small message window containing all messages produced by the last operation performed.

In this example, these messages reveal that *one* is present in at least two fields in this Database Dictionary file. However, since the **Contains** button was used, the letters are found in their specific order, regardless of where they appear in the field name.

For a more exact match on the word *one* you can use the **Exact match** or **Starts with** button.

(Using these options against this example returns no positive hits using **Exact match** or **Starts with**.)

Creating Subtables from an Array of Structures

ServiceCenter enables a dbdict administrator to manage data more effectively by creating subtables of unique and non-unique attributes within an array of structures. You can use this feature to:

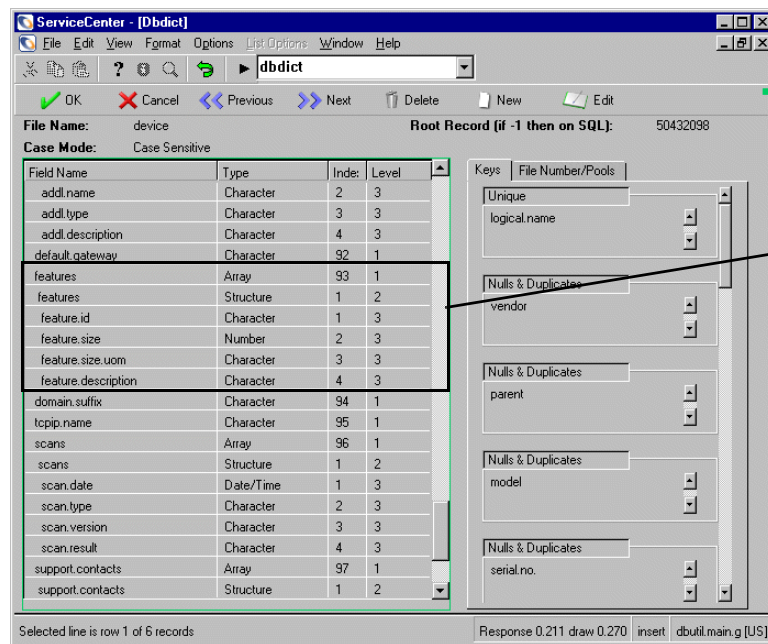
- Improve mapping to external SQL database tables.
- Implement a more cost-effective solution for managing attribute information.
- Simplify queries.

The dbdict administrator can identify two subtable names for each array of structures in the dbdict. One table contains the names of unique attributes; the second table names non-unique attributes. A pop-up utility dialog box enables you to identify which attributes are unique.

The subtable feature helps you create queries that can return detailed information. This type of available detail can improve business and management decisions. You can create subtables for an array of structures in any dbdict. ServiceCenter ships with subtables already created for all arrays of structures in the inventory dbdicts.

Organization in the Database Dictionary

In each database dictionary, an array of structures appears in a group. The group is organized hierarchically, with the field of type *array* listed first. The field of type *structure* is listed next, and is indented once from the field of type array. The remaining fields are indented and listed below the field of type structure, and can be of types Character, Number, Location, or Date/Time.



Array of Structures

Figure 9-6: The *features* array of structure in the *device* dbdict.

How to create subtables from an Array of Structures

To create subtables from an array of structures, first name both the unique file and the attribute file. Next, designate which fields in the structured array are unique. Finally, create the subtables.

To name the unique and attribute files:

- 1 Login as *falcon*.
- 2 Search for the dbdict from which you want to create subtables.
For this example, the *device* dbdict is used.
- 3 Scroll down the **Type** column and locate an Array of Structures.
An array of structure is identified by a field of type Array with a field of the same name, type Structure directly below the Array field. The Structure field name is indented.
For this example, the array of structure is **features**.
- 4 Locate the **features** field of type **Structure**.
- 5 Insert your cursor in the **Type** column for the *features structure* field.

features	Array	93	1
features	Structure	1	2
feature.id	Character	1	3
feature.size	Number	2	3
feature.size.uom	Character	3	3
feature.description	Character	4	3

Figure 9-7: Structure field of the *features* array of structure

- 6 Click Edit.

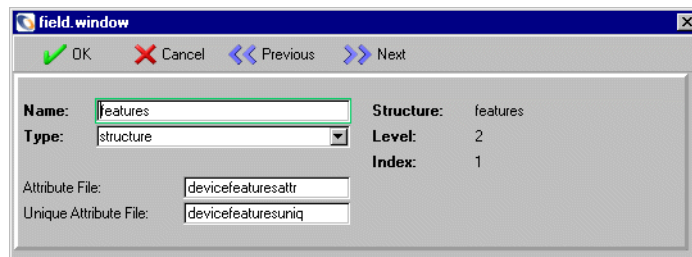


Figure 9-8: The field.window dialog box

- 7 In the **field.window** dialog box, type a name for the attribute file.
The standard naming convention for the attribute file is:
dbdict + array + attr
For this example, the attribute file is named **devicefeaturesattr**.

Note: No spaces or special characters are allowed in the subtable names.

- 8 In the **field.window** dialog box, type a name for the unique file.

The standard naming convention for the unique file is:

dbdict + array + uniq

For this example, the unique file is named `devicefeaturesuniq`.

Note: If the attribute file field is left blank, only the unique subtable will be created.

- 9 Click OK.

To designate unique fields in the array of structure:

- 1 Locate the field you want to designate as unique.
For this example, the unique field will be the `feature.id` field.
- 2 Insert your cursor in the Type column for the field you want to designate as unique.
- 3 Click Edit.

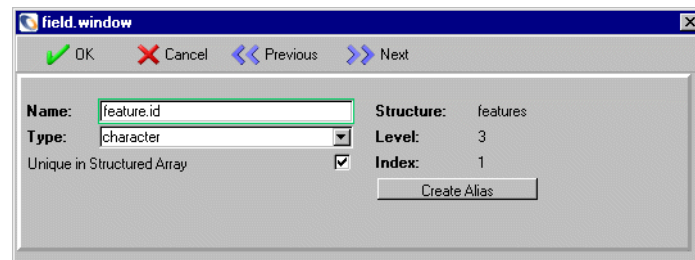


Figure 9-9: The Unique in Structured Array check box

- 4 Click the **Unique in Structured Array** check box.
- 5 Click OK.
- 6 Repeat steps one through five for each field you want to designate as unique.

To create the unique and attribute tables:

- 1 From the database dictionary, click OK.
ServiceCenter updates the dbdict and creates the unique and attribute subtables from the array of structures.

To view the unique and attribute subtables:

- 1 From the ServiceCenter Main menu, or the command line, navigate to the Database Dictionary utility.
- 2 Search for the dbdict that contains the array of structure from which the subtables were created.

The unique and attribute files you named should appear in the recordlist.

- 3 Double-Click on the file you want to view.

Actions

Create File

When a file with a structured array is created, and the sa.unique.key, sa.unique.filename, and sa.attribute.file fields are populated, the RTE will automatically create the unique and the attribute files.

Warning: A field must be designated as unique *before* the array subtables are first created. A new subtable is not created if a field is designated as unique *after* the original table was created.

Delete File

When a file is deleted that has a structured array, and the unique and attribute fields are populated, then the RTE will automatically delete the unique and the attribute files.

Warning: If you copy a table (for backup or testing purposes), the references for the array tables *still point to the original array tables*. When you delete your copy, the cascading delete will delete the *original* array tables with it.

Delete Record

When a delete is done against a record, the RTE will delete the main and the attribute entries, but will not delete the unique entries.

Insert

When an insert is done against a record, the RTE will insert the main, the attributes, and the unique entries that are needed.

Reset

A reset against one of these files will reset the main file and attribute file but will leave the unique file alone.

Regen

A regen will only regen the main file. You will have to explicitly ask for a regen of the unique and attribute files if it is wanted.

Update

When an update is done, the RTE will update everything that has changed but will not get rid of any unique entries.

10 Adding and Modifying Fields in the Database Dictionary

CHAPTER

Important: Menu items and messages may appear differently than shown, depending on your version of ServiceCenter.

Adding a New Field

You can add fields to an existing Database Dictionary record with or without records in the file. Existing records in the file will not contain any data in the newly added fields unless each individual record is updated with data or a Mass Update is performed.

Note: The same field name cannot appear more than once in the Database Dictionary (except for arrays and arrayed structures, which produce different field levels within the file).

When you access an existing Database Dictionary, the cursor is automatically positioned on the word **descriptor**. The **descriptor** field is referred to as the high level file structure. This field is displayed in all database dictionaries.

The cursor must be on a structure when adding fields. If you want to add a field within a structure other than the **descriptor** structure, place the cursor on the field defined as the structure before executing the **add field function**. Refer to *Fields within structures* on page 300 for more information.

Forming field names

Follow these rules when forming field names.

- Valid characters include uppercase and lowercase letters of the alphabet, numeric characters 0 through 9, and the *period* character.
- Field names must begin with a letter of the alphabet.
- Spaces and all other special characters are not permitted.
- Do not use ServiceCenter reserved words (uppercase or lowercase):

AND	IF	NULL
BEGIN	IN	OR
DO	ISIN	STEP
ELSE	LIKE	THEN
END	MONTH	TRUE
FALSE	NAME	WHILE
FOR	NOT	UNKNOWN

Scalar fields

A scalar field is a simple data element comprised of a single occurrence of data (as opposed to an array).

The following steps demonstrate how to add a scalar field to the Database Dictionary record. In this example, we are adding a logical (true/false) field to the device file that determines whether or not a given user device resides on a local area network (lan).

- 1 Open the Database Dictionary record for the device file using either:
 - The Database Dictionary utility.
 - or
 - The Forms Designer. See the Forms Design sections for questions about the Forms Designer.

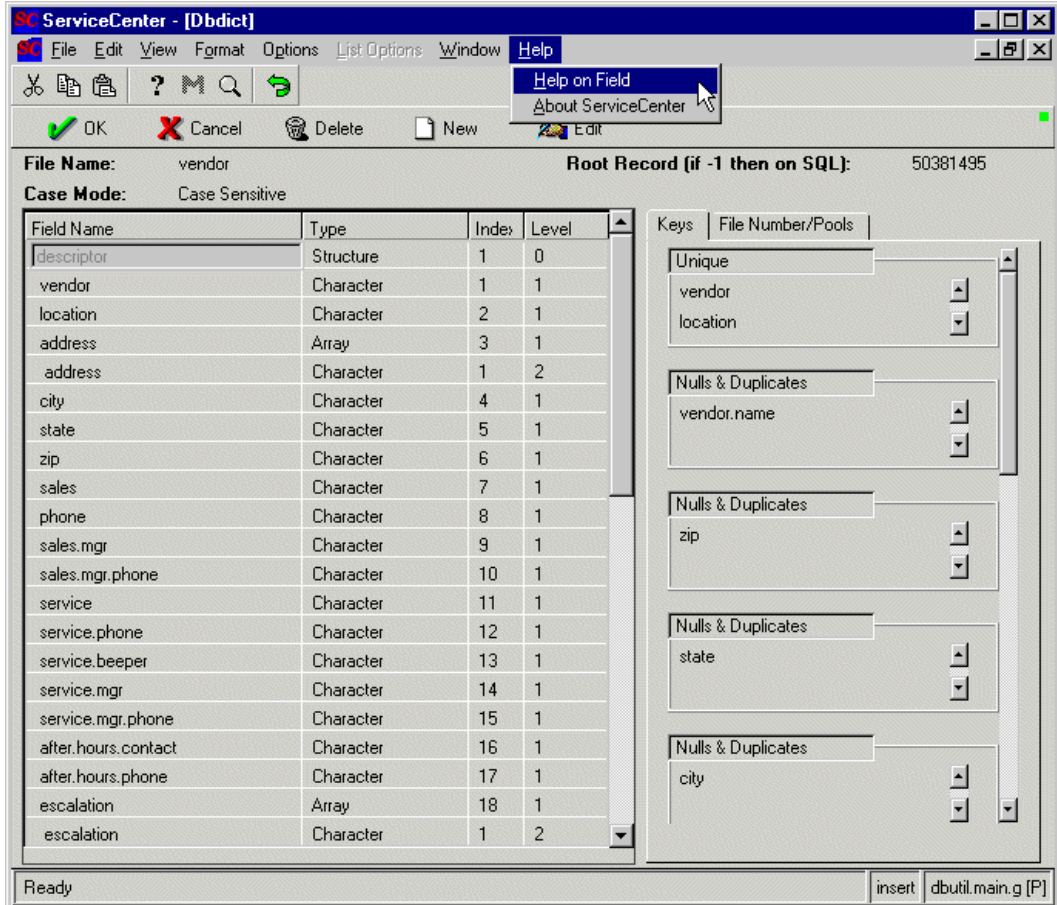
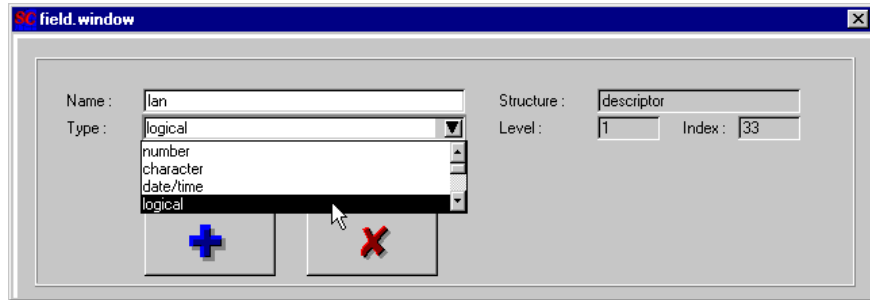


Figure 10-1: Example Database Dictionary record—device

- 1 Once the file is open, place the cursor in the descriptor structure
- 2 Click New.
The field editing window appears.
- 3 Enter lan in the Name field.

- 4 Enter **logical** in the **Type** field or select *logical* from the drop-down list of valid data types.



- 5 Click the **Add** button to add this field to the *device* Database Dictionary record.
The field **lan** now appears at the bottom of the list of fields in the *device* Database Dictionary record.
- 6 Use the scroll bar to the right of the fields to locate the new field at the bottom of the fields list.

Note: All fields added to the Database Dictionary record (if entered in uppercase) will be converted to lowercase to be consistent with fields already defined in dictionaries and on formats.

- 7 You can now re-edit your Database Dictionary record, update it, or cancel without updating.
Note: You must update your Database Dictionary record when adding a new field, or the Database Dictionary record will remain the same. You must also add new field(s) to the appropriate format(s) via Forms Designer.
- 8 Click **OK** when you have finished adding fields to the Database Dictionary record to close and automatically update the file. This returns the message *Record updated in the dbdict file.*

Valid field types

Number-allows you to input only numeric data in the field.

Character-allows you to input any available characters in the field (numbers, letters, special characters).

Date/Time—requires field to contain the following formatted data:
MM/DD/YY HH:MM:SS or MM/DD/YYYY HH:MM:SS for years of 2000 or
DD HH:MM:SS for time intervals.

You can enter a 2-digit number for the year field for dates between 1950 and 2049. For dates outside this range, you must enter a 4-digit year. (for example, 69 would be treated by the system as 1969)

The SS is assumed to be :00 if left blank. HH:MM:SS is optional if the time is 00:00:00. The system will default HH:MM:SS to 00:00:00

Logical—contains one of the following logical values: *true*, *false*, *NULL* or *unknown*. ServiceCenter accepts input of *yes* or *no* and converts it to *true* or *false* respectively.

Label—used only in RAD applications.

Record—used only in RAD applications.

Array—compound data element that contains one or more elements of a single data type.

Structure—compound data element that contains several sub-fields, each of which may have a different data type.

Expression—used only in RAD applications.

Arrays

An array is used to track multiple occurrences of a given field. The values in the field are all the same field type. For example, maybe you want to track all of the components for a personal computer (PC) in the *enduser* file. To accomplish this, you can create an array field, further defined as a character type, rather than creating multiple scalar fields.

The following screens illustrate how to add a field to the *enduser* file to track the various components comprising a PC workstation.

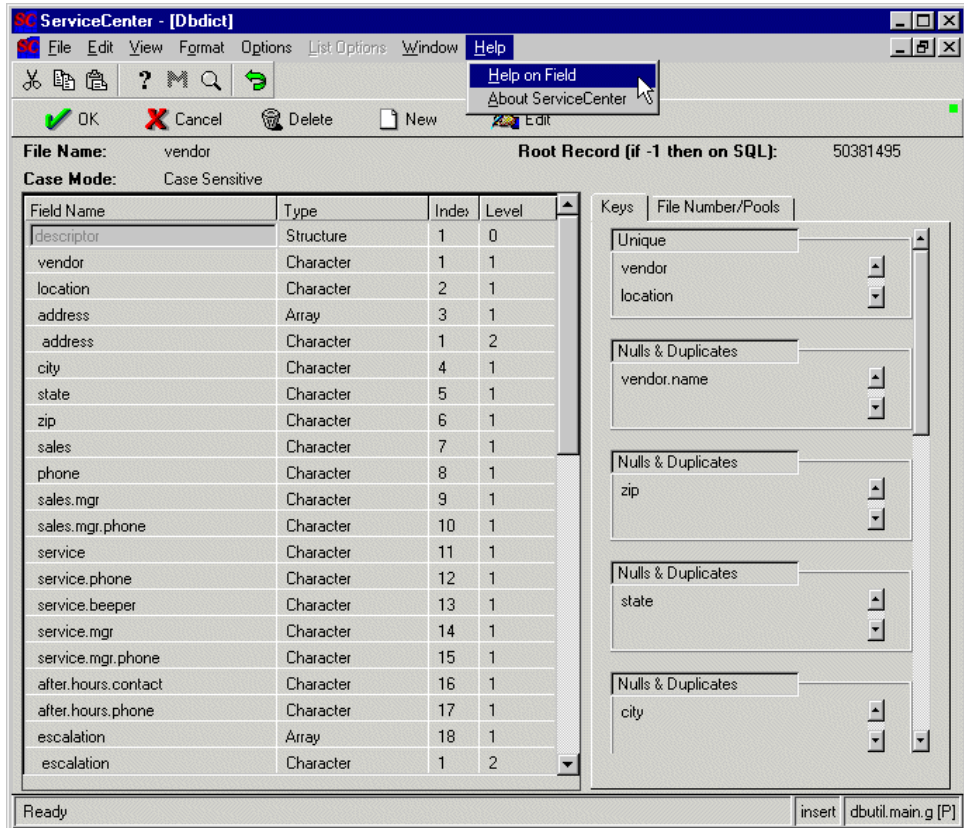


Figure 10-2: Enduser Database Dictionary record

- 1 Place the cursor on the **descriptor** structure and click **New**.
- 2 Enter `pc.components` in the **Name** field of the pop-up window.

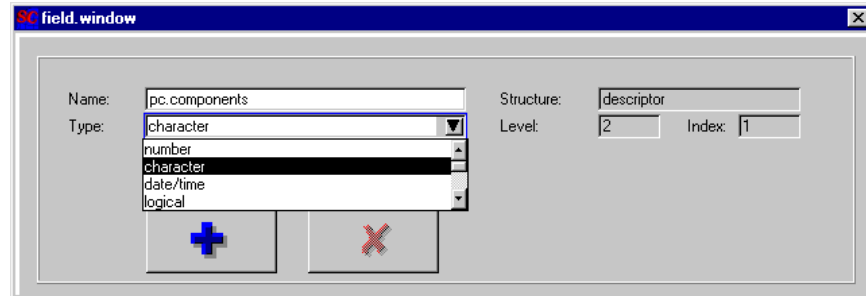
Note: When specifying field names, no blanks are allowed between words. Always use a period as a delimiter.
- 3 Enter `array` in the **Type** field or select it from the drop-down menu.
- 4 Click **Add**.

Notice the message *Enter data type of array's element* at the bottom of your screen.

A new window also appears, which looks similar to the previous. This window is requesting a data type for the array element you have created.

Notice, *pc.components* is already displayed in the Name field and the Type field is blank. When adding an array field, you also need to specify the type of array (e.g. character, number, date/time, etc.).

- 1 Enter character in the Type field, or select *character* from the drop-down list of valid data types.
- 2 Click Add to add the new array element.



- 3 Use the scroll bar to locate the new array and element at the bottom of the fields list.
- 4 There are two entries in the record for the field **pc.components** (Figure 10-3 on page 298). The first entry defines the field as an array. The second entry defines the data type of the array. Both entries are listed as **pc.components** (using the same field name for both entries is the system default, however, you can override this if you choose). The two entries are added to the bottom of the list of fields in the *enduser* Database Dictionary record. The second entry is also indented from the first entry (this occurs only if the number of fields specified in the Database Dictionary record is less than 50).
- 5 You can now re-edit your Database Dictionary record, update it, or cancel without updating.

Note: You must update your Database Dictionary record when adding a new field, or the Database Dictionary record will remain the same. You must also add new field(s) to the appropriate format(s) via Forms Designer. Refer to the *Forms Designer* section for information on coordinating record changes.

- Click OK once you are finished adding fields to the Database Dictionary record, to close and automatically update the file. This returns the message *Record updated in the dbdict file.*

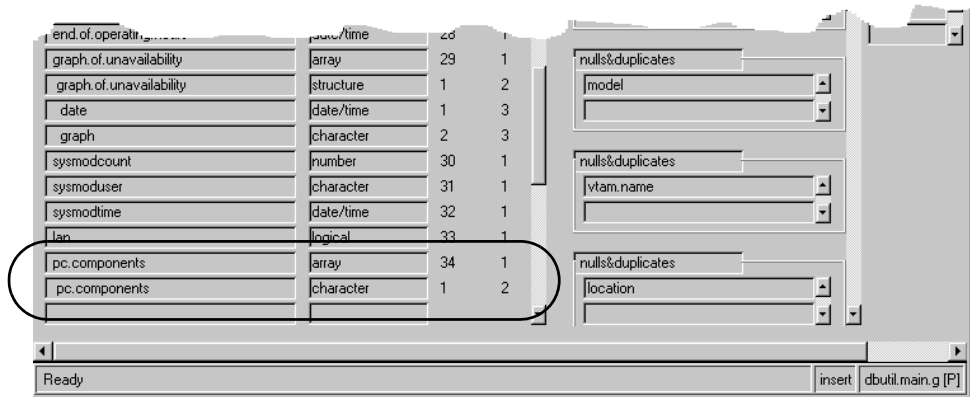


Figure 10-3: New array fields added to enduser record

Structures

Structures are used in conjunction with subformats. Use structures when you have a common group of fields that you want displayed on multiple formats. For example, Incident Management has three structures defined in the problem Database Dictionary record, the **header**, **middle**, and **action** structures. The fields defined in the **header** and **action** structures are set up to be used in every problem format, regardless of the category. Rather than painting each category format with the same input fields, specific formats are created and called in as a subformat on every category screen. The fields defined on a given subformat may be of different type definitions.

Note: On the primary format the input field defined as a structure is associated with the subformat containing some or all of the fields defined within that structure. Fields defined on the given subformat must be defined within the associated structure, but not all fields defined within the associated structure need to be contained on the given subformat.

- 6 You can now add fields within the structure you just created (see *Fields within structures* on page 300).
- 7 Click OK once more to save your edits and end, or click **Cancel** to quit end without recording edits.
- 8 Reopen the record from the prompt now to re-edit your Database Dictionary record or update it.

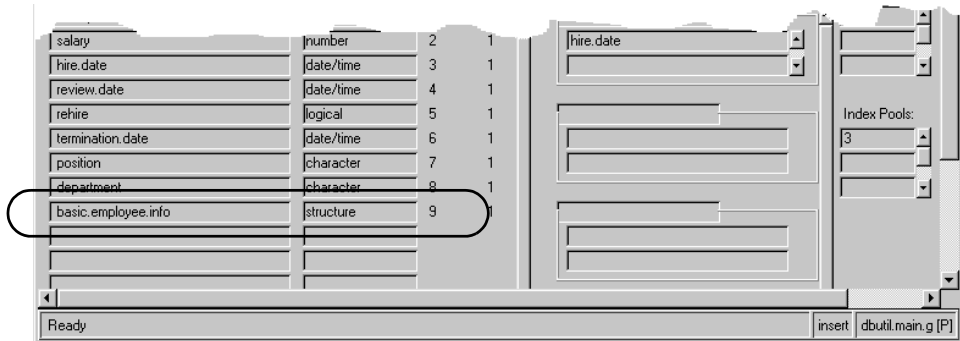


Figure 10-5: Database Dictionary record with new structure

Note: You must update your Database Dictionary record when adding a new field, or the Database Dictionary record will remain the same. Refer to the *Forms Designer* section of the *System Tailoring Guide* for details on adding new fields to ServiceCenter forms.

- 9 Click OK once you are finished adding fields to the Database Dictionary record, to close and automatically update the file. This returns the message *Record updated in the dbdict file.*

Fields within structures

As mentioned above, it is important to define fields within structures when these fields exist on a format(s) associated with the input field structure name. To define fields within structures, your cursor (focus) must be on the structure name in the Database Dictionary record. If fields do not need to be defined within a given structure in a Database Dictionary record, the cursor should be positioned on the *descriptor* structure when adding the fields.

The following screens illustrate adding fields to the *basic.employee.info* structure created in the previous section.

- 1 Position the cursor on the structure *basic.employee.info*.

- 2 Press the New button.
Notice *basic.employee.info* is written in the **Structure** field.
- 3 Enter *name* in the **Name** field of the pop-up window.
- 4 Enter *character* in the **Type** field, or select *character* it from the drop-down list of valid data types.
- 5 Click **Add** to add the new field in this structure.

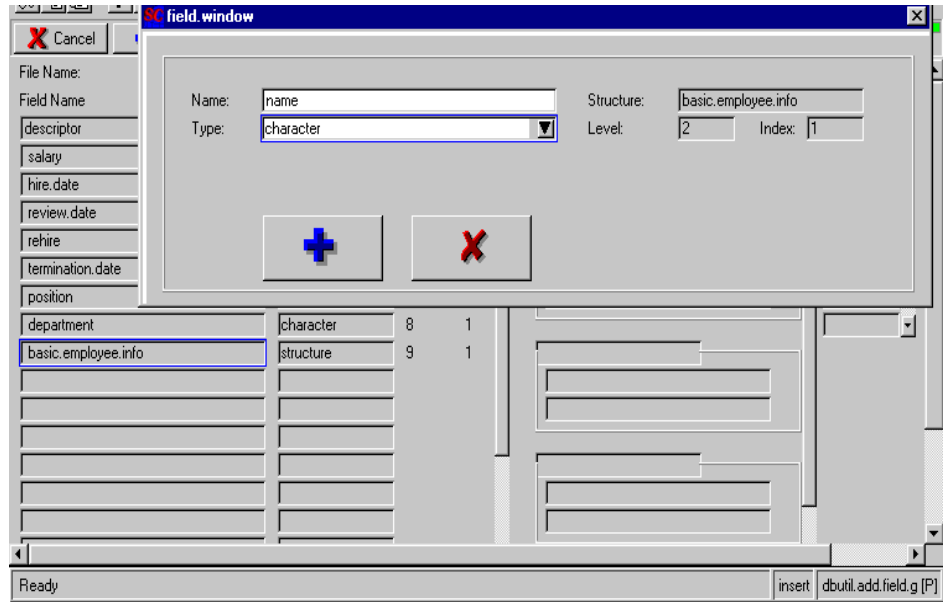
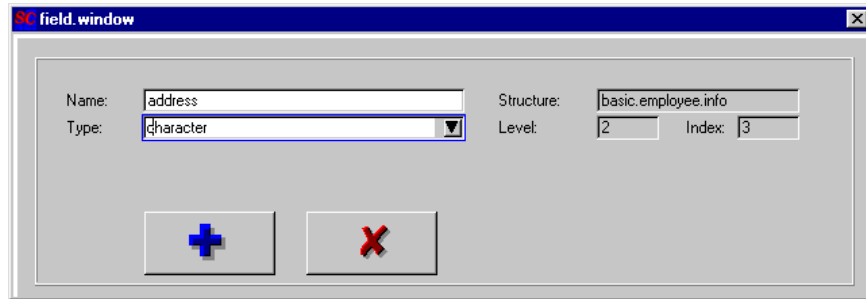


Figure 10-6: Adding fields to a new structure

The **name** field is added to the bottom of the list of fields under the *basic.employee.info* structure.

- 6 Place the cursor on the *basic.employee.info* structure.
- 7 Click **New**.
- 8 Enter *telephone* in the **Name** field in the pop-up window.
- 9 Enter *character* in the **Type** field or select *character* from the drop-down list of valid data types.
- 10 Click **Add** to add the new field.
The **telephone** field is added to the bottom of the list of fields under the *basic.employee.info* structure.
- 11 Place the cursor on the *basic.employee.info* structure and click **New**.
- 12 Enter *address* in the **Name** field of the pop-up window.

- 13 Enter character in the Type field or select *character* from the drop-down list of valid data types.



- 14 Click Add to confirm the new field.

The **address** field is added to the bottom of the list of fields under the *basic.employee.info* structure (Figure 10-7 on page 302).

- 15 Click OK once more to save your edits and end, or click Cancel to quit end without recording edits.

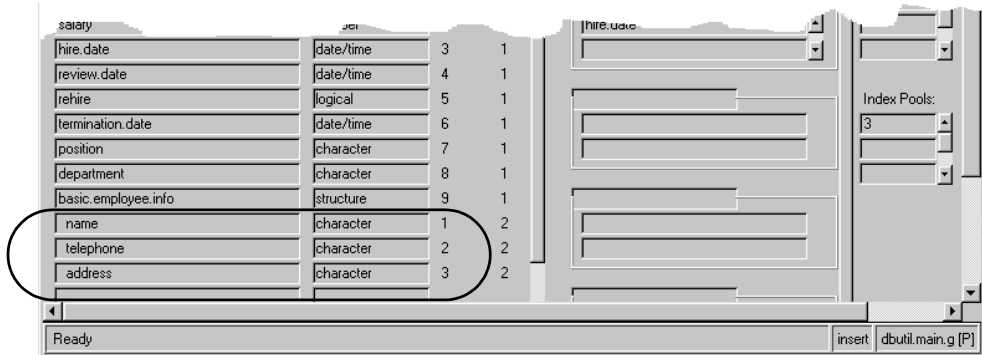


Figure 10-7: New fields added to new structure

- 16 Reopen the record from the prompt now to re-edit your Database Dictionary record or update it.

Note: You must update your Database Dictionary record when adding a new field, or the Database Dictionary record will remain the same. You must also add new field(s) to the appropriate format(s) via Forms Designer. Refer to the *Forms Designer* section of the *System Tailoring Guide* for more information.

- 17 Click **OK** once you are finished adding fields to the Database Dictionary record, to close and automatically update the file. This returns the message *Record updated in the dbdict file.*

Note: When creating categories in Incident Management, define any additional fields in the newly created categories within the *middle* structure in the *problem* Database Dictionary record.

Arrayed structures

An arrayed structure is comprised of more than one field whose field type definitions may or may not be the same. You may want to define an arrayed structure to your Database Dictionary record so that on a given form (format) attached to a file, you have the ability to scroll more than one field at a time. In other words, when the cursor is on a field to scroll information, associated information contained in one or more other fields will scroll as well.

Warning: You cannot use the SQL Query method against fields contained within an arrayed structure.

For example, you might want to track contact names and phone numbers in one of your inventory files. When you are scrolling the array of contact names, you also want the array of associated telephone numbers to scroll.

Another good example of an arrayed structure is the *graph of unavailability*. This structure is defined in the standard inventory files distributed with the Inventory/Configuration application. If you wish to create your own inventory files and track availability, you need to add the following availability fields to the inventory file(s):

Database Dictionary Field Name	Field Type
number.of.problems	number
explicit.unavailability	date/time
implicit.unavailability	date/time
perceived.unavailability	date/time
start.of.operating.hours	date/time
end.of.operating.hours	date/time
graph.of.unavailability	array

Database Dictionary Field Name	Field Type
graph.of.unavailability	structure
date	date/time
graph	character

The field names need to be spelled exactly as they are shown in the above table and defined with the specified field type.

Let's assume you have a *devicepc* file, and would like each record within this file to contain statistical information on availability. The following screens layout the steps for adding the availability fields which pertain to the *devicepc* file.

Note: These **graph** fields also need to be added to the inventory format(s). Refer to the *Forms Designer* section of the *System Tailoring Guide*.

- 1 Place the cursor on the **descriptor** field.
- 2 Click **New**.
- 3 Enter *graph.of.unavailability* in the **Name** field of the pop-up window.
- 4 Enter *array* in the **Type** field.

5 Click Add.

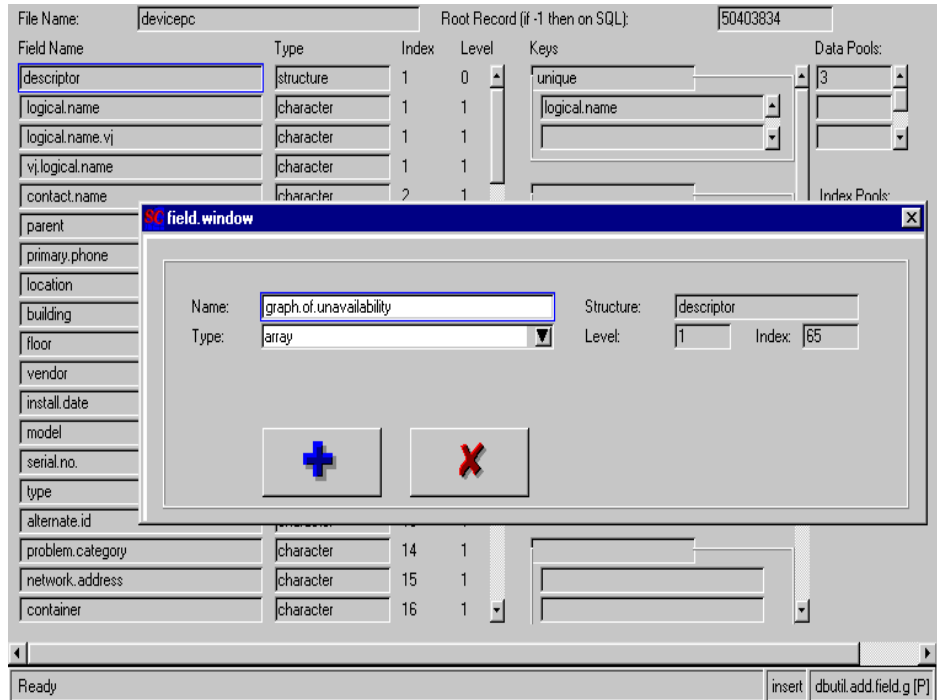


Figure 10-8: Adding structured array field

The message *Enter data type of array's element* appears in the bottom of the screen and the pop-up window refreshes.

The Name field still displays *graph.of.unavailability*, but the Type field is blank. In order to create an arrayed structure, you must define the type of array as a structure.(e.g. character, number, date/time, structure).

- 6 Enter structure in the Type field or select *structure* from the drop-down list.
- 7 Click Add.

Name	Type	Count	Order
y2k_status	character	55	1
is_down	logical	56	1
device_type	character	57	1
description	character	58	1
sysmodcount	number	59	1
sysmoduser	character	60	1
sysmodtime	date/time	61	1
contract_id	number	62	1
subtype	character	63	1
problem_priority	character	64	1
graph.of.unavailability	array	65	1
graph.of.unavailability	structure	1	2

Figure 10-9: New structured array field

There are two entries in the Database Dictionary record for the field **graph.of.unavailability**. The first entry defines the field as an array. The second entry defines the data type of the array. Both entries are listed as **graph.of.unavailability** (using the same field name for both entries is the system default; however, you can override this if you choose). The two entries are added to the bottom of the list of fields in the *devicepc* Database Dictionary record.

Note: The second entry is indented in the above example. This occurs only if the number of fields specified in the record is less than 50.

- 8 Tab to the second occurrence of the **graph.of.unavailability** field (which is the *structure* type).

Important: For any field to be part of a structure, the cursor must be positioned on that structure field name before clicking **New** for the new field to be associated with it.

- 9 Click **New**.

Notice the **Structure** field contains the value *graph.of.unavailability* which is the current structure to which you are adding fields.

- 10 Enter *date* in the **Name** field of the pop-up window.
- 11 Enter *date/time* in the **Type** field.

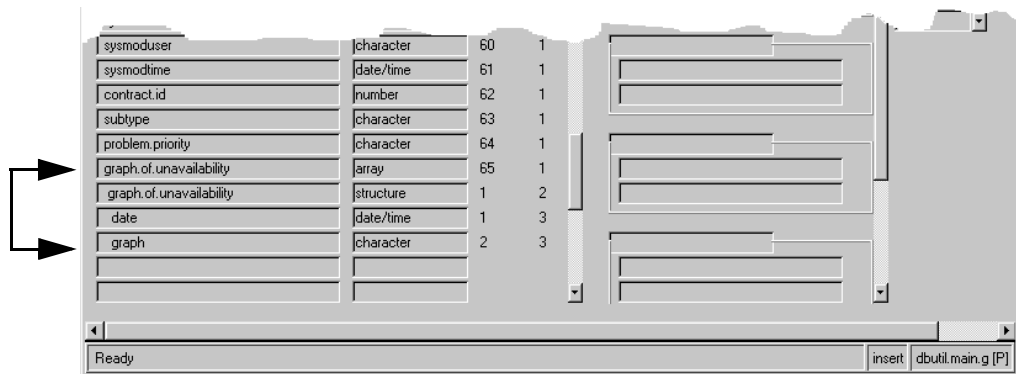
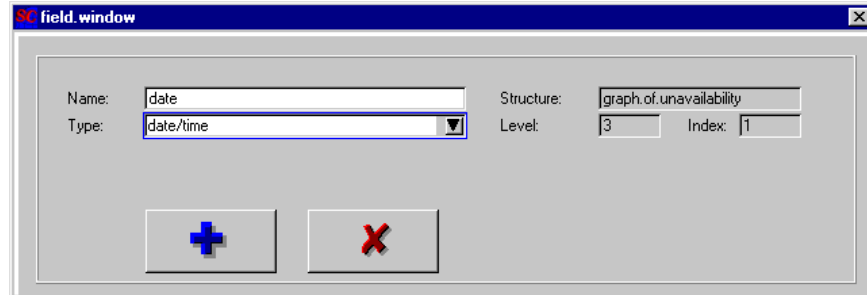


Figure 10-10: Fields added to structure within array

12 Click Add.



Notice the **date** field is displayed under the structure field **graph.of.unavailability**.

13 Select the second occurrence of the **graph.of.unavailability** field (which is the *structure* type).

Important: For any fields to be part of a structure, the cursor must be positioned on that structure field name before clicking **New**.

14 Click **New**.

Notice the **Structure** field contains the value *graph.of.unavailability* which is the current structure to which you are adding fields.

15 Enter *graph* in the **Name** field of the pop-up window.

16 Enter *character* in the **Type** field.

17 Click **Add**.

Note: The **graph** field is added as the last field under the structure field **graph.of.unavailability**, and the message *graph field added to the pcdevices file* is issued. In this case, the **graph** field is behind the window at the bottom of the screen.

You have now added all of the required fields comprising the **graph.of.unavailability** structure. From here you can re-edit your Database Dictionary record, update it, or cancel without updating.

Note: You must update your Database Dictionary record when adding a new field, or the Database Dictionary record will remain the same. You must also add new field(s) to the appropriate format(s) via Forms Designer. Refer to the *Forms Designer Guide* for more information.

- 18 Click **OK** once you are finished adding fields to the Database Dictionary record, to close and automatically update the file. This returns the message *Record updated in the dbdict file*.
- 19 Press **Back** to end this session with the Database Dictionary and return to the main menu.

Alias Fields

An alias field is a field that has a different name but the same index, level and type as an existing field. Alias fields can be used to streamline file maintenance, to force fully keyed queries (as in the Incident Management and Service Management modules), and to facilitate virtual joining.

If regular Purge/Archives are performed on files with different field names for the search criteria (e.g., *close.time* in the **problem** and **probsummary** files, *msg.time* in the **msglog** file, and *evtime* in the **eventin** file), you can build an alias in each file called **archive.time** and then perform all archive activity based on **archive.time** without regard to the file name.

In Service Management and Incident Management, alias keys are used to ensure that a fully keyed query is performed each time a QBE argument is entered. Refer to the *Incident Management Tailoring Guide* for more information.

In cases where a field must be defined as a normal link as well as a virtual join, but the target file and/or field is different, an alias can be defined for the source field. For example, review the *devicepc* Database Dictionary record and the *device.pc* format and link records. In this example, the *vj.logical.name* field is the alias of the *logical.name* field.

File Name	File Number	Root Record	Data Pools	Index Pools	Shadowed
devicepc	232	50403834	{3,..}	{3,..}	false

Field Name	Type	Index	Level	Keys	Data Pools
descriptor	structure	1	0	unique	{3}
logical.name	character	1	1	logical.name	
logical.name.vj	character	1	1		
vj.logical.name	character	1	1		
contact.name	character	2	1		
parent	character	1	1		
primary.phone	character	4	1		

Ready insert dbutil.main.g [P]

Alias fields

Shared Level

Shared Index Reference Number

Figure 10-11: Alias fields used in inventory Database Dictionary files

Adding an alias field

The steps detailed below describe using the Arrayed Structure Maintenance Utility (as options) to maintain alias fields in the Database Dictionary. In this example, a new alias for the flag field called *network.name.flag* is added to the *probsummary* file.

Warning: Do not attempt to use this feature to maintain non-alias fields; use the standard Database Dictionary utility described in previous pages of this chapter.

- 1 Select the Toolkit tab in the system administrator's home menu.
- 2 Click Database Manager.

- 3 Enter =dbdict in the Form field of the Database Manager dialog box.
- 4 Press Enter.
A blank dbdict record form is displayed.

Note: You must have SysAdmin capabilities to access Database Dictionary from the Database Manager.

- 5 Enter probsummary at the Name field prompt.

The screenshot shows a window titled "ServiceCenter - [dbdict]" with a menu bar (File, Edit, View, Format, Options, List Options, Window, Help) and a toolbar (Back, Add, Search). The main area contains a form with the following fields:

- Name: (highlighted with a red box)
- Root Record (if -1 then on SQL):

Below these fields is a table with the following columns: Name, Type, Index, Level, Keys, Flags, and Names. The table is currently empty.

At the bottom of the window, the status bar shows "Ready" on the left and "Response 0.260 draw 0.481 insert dbdict.g(db.search) [US]" on the right.

Figure 10-12: Search form for dbdict records in Database Manager

- 6 Click Search.

The probsummary record is displayed.

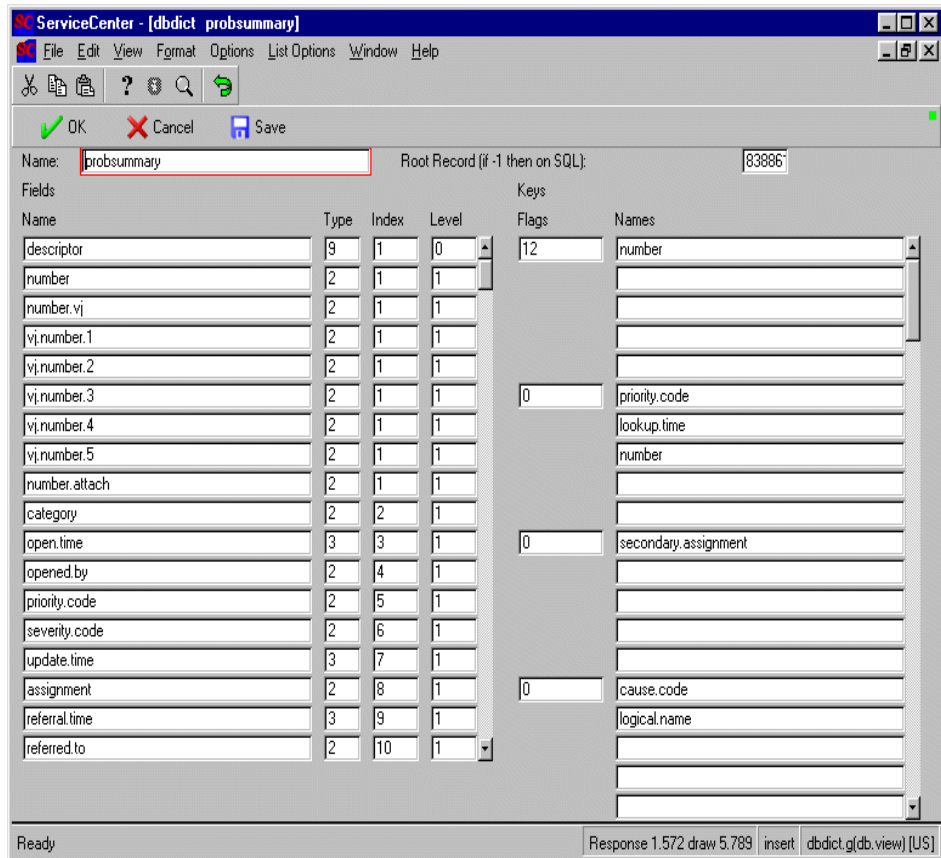


Figure 10-13: System file record (probsummary) displayed in Database Manager

- 7 Scroll through the probsummary record using the scroll bars to the right of each column.

These are the fields and keys of the probsummary database dictionary record displayed in an editable format.

- 8 Pull down the **Options** menu.
- 9 Select **as options**.

The **as options** function manipulates arrays of structures, allowing insert, delete, copy and move.

The screen is refreshed displayed in read-only mode.

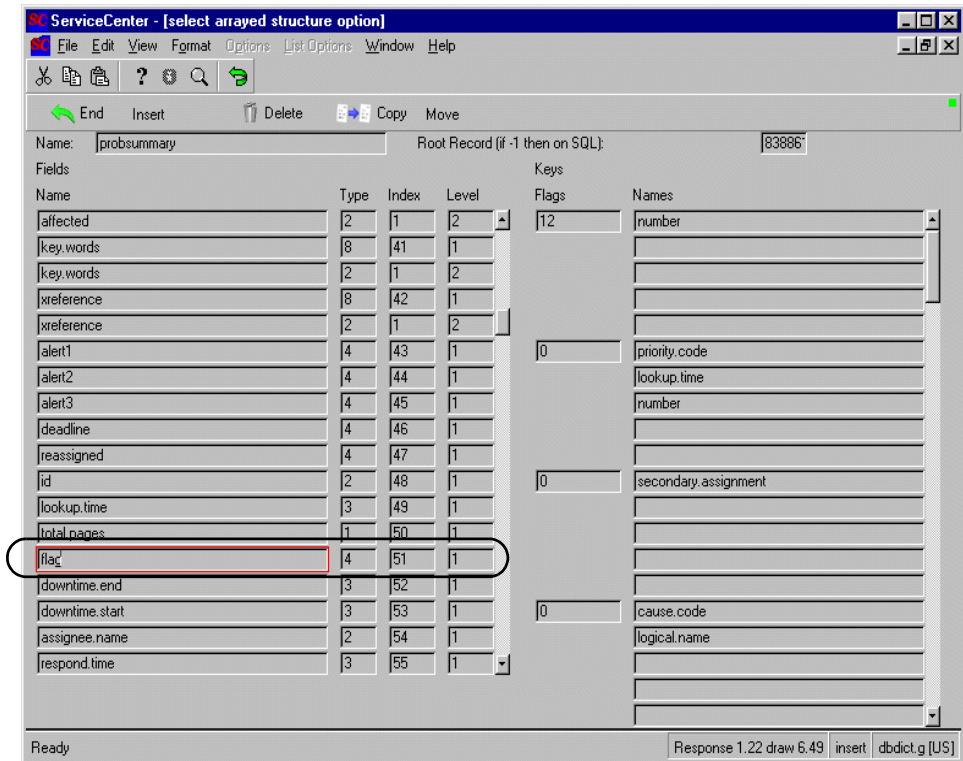


Figure 10-14: flag field of probsummary file record

10 Scroll the fields array until the **flag** field appears on the screen.

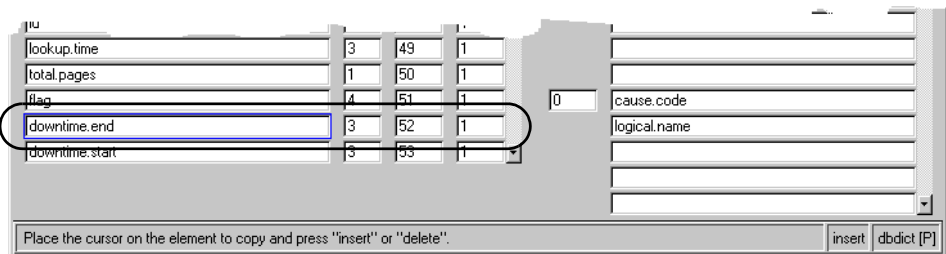
11 Position your cursor on the **flag** field.

12 Select copy (F9).

The screen is refreshed in unprotected mode with new option buttons.

13 Position your cursor on the first alias key - in this example there are none, but we will insert the new alias field immediately following the **flag** field, so select the **downtime.end**

14 Select insert.



This action inserts a copy of *flag* under the original, with the same type, index and level. The record will be protected against data entry.

Field Name	Type	Index	Level
id	2	48	1
lookup.time	3	49	1
total.pages	1	50	1
flag	4	51	1
flag	4	51	1
downtime.end	3	52	1
downtime.start	3	53	1
assignee.name	2	54	1
respond.time	3	55	1
contact.name	2	56	1
contact.name.vj	2	56	1

Ready insert dbdict [P]

Figure 10-15: Duplicate/alias flag field

- 15 Select End to exit. The screen will be unprotected, allowing data entry.
- 16 Scroll to the second *flag* field, and change its value to *network.name.flag*.

Field Name	Type	Index	Level
lookup.time	3	49	1
total.pages	1	50	1
flag	4	51	1
network.name.flag	4	51	1
downtime.end	3	52	1
downtime.start	3	53	1
assignee.name	2	54	1
respond.time	3	55	1
contact.name	2	56	1

Ready insert dbdict [P]

Figure 10-16: Modifying alias flag field

- 17 Click Save (F2) to confirm the change.

Warning: Do not select Exit. To do so will discard your changes and return to the empty prompt screen

The Database Dictionary record is re-displayed and the original options are available.

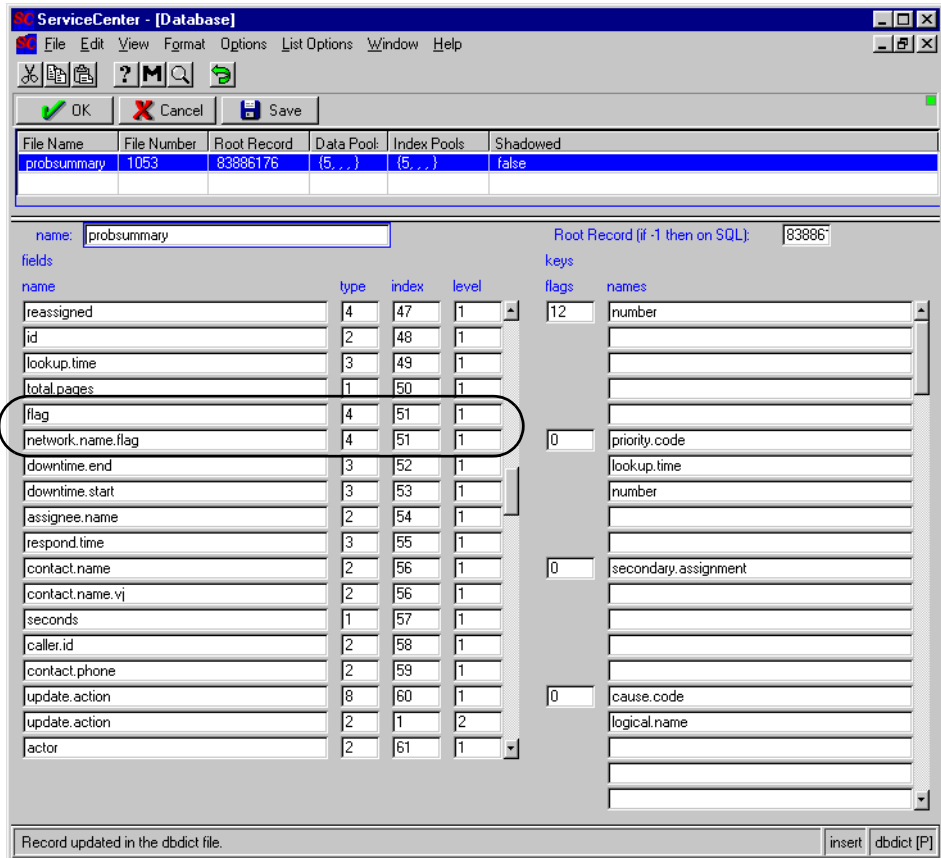


Figure 10-17: probsummary file record with alias fields

18 After the record has been updated, exit using OK.

Alias fields should always be defined immediately following their “twin,” so that the index numbers in the Database Dictionary record are sequential and all duplicate indices are adjacent.

Adding a Key

A key is an identifying field in a file, used when queries are processed so all fields in a record do not need to be searched for set criteria. A field or fields in a key must be defined as a field or fields in the Database Dictionary record. When a Database Dictionary file is created through Forms Designer, ServiceCenter automatically uses the first field defined on the associated format to create a *unique* key. You need to manually modify the Database Dictionary record to add the necessary keys to support on-line and reporting queries, as well as sort sequences. Many other default system files, set up in the base system, may need to be changed, based on your individual search and reporting requirements. This section details the manual addition and manipulation of Database Dictionary record/file keys.

Important: If records already exist in the file when you add a key, updating the Database Dictionary record will cause ServiceCenter to automatically perform a regen.

Note: If you add keys to a file that contains records, be careful when defining key types. For example, you should not define a new key as **no nulls** (no blank values) if records exist in the file that contain no data in the particular field(s) comprising the key.

In this case, each record not containing a value in the key field would require an update, populating the field with data. The updates must be performed prior to adding the **no nulls** key in order to satisfy the new key definition.

Adding a key as the first key

The following operations require that you be in the Database Dictionary application. Enter the application via the Toolkit tab on the main menu.

The following example illustrate how to add the **customer.no** field as a *no nulls* key to the **enduser** Database Dictionary record.

Important: `customer.no` must be defined as a field in the *enduser* Database Dictionary record.

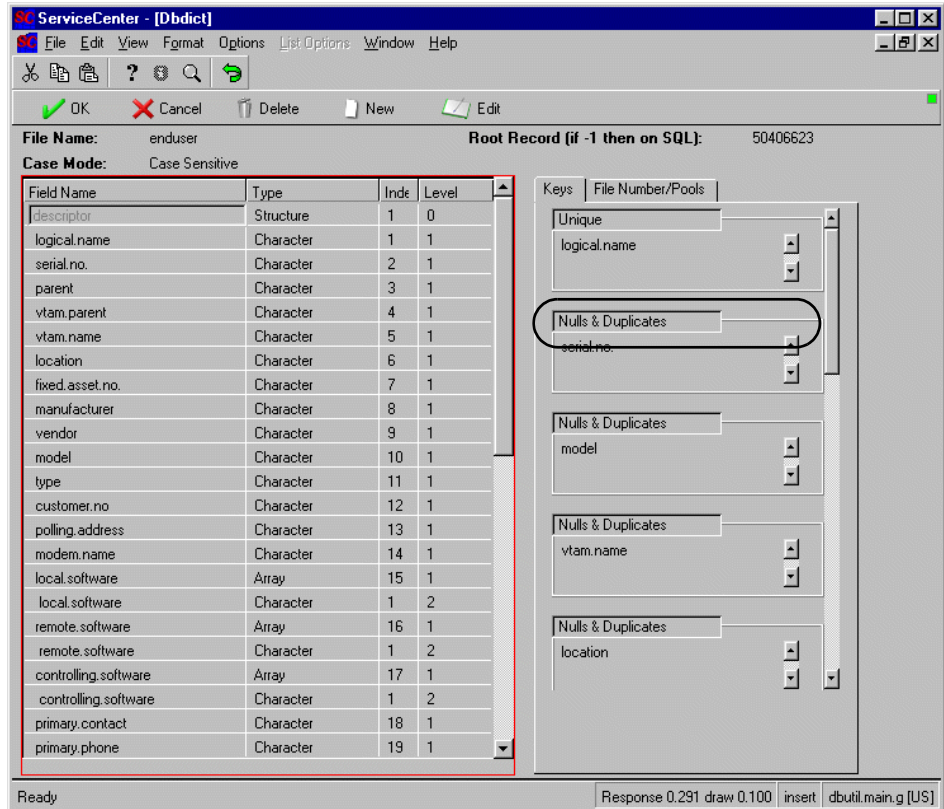


Figure 10-18: Adding a key to the enduser Database Dictionary record

- 1 Place the cursor on the key type of the first key, *unique* in this example.
- 2 Click New.
- 3 In the new key window, select a valid key in the **Type** field by either typing in the value or selecting it from the drop-down list under the arrow button. Use *no nulls* for this example. (Peregrine recommends using no nulls or unique as the first key, always.)

The following are valid key types:

Nulls & Duplicates—allows nulls and duplicates in the field(s) comprising the key.

No Nulls—allows duplicates but no nulls (i.e., must contain data in at least one of the fields in the key).

No Duplicates—allows nulls but no duplicates (i.e., data must be unique, but field may be left blank).

Unique—must contain unique data and cannot be left blank.

- 4 Place the cursor in the blank field names array.
- 5 Type `customer.no` in the field names array.
- 6 Click **Add** to add this key to the `enduser` Database Dictionary record.

The key `customer.no` now appears as the first key in the list of keys in the `enduser` Database Dictionary record. All other keys are moved down the screen. You can scroll the keys array to view the remainder of keys in the list.

File Name	File Number	Root Record	Data Pool	Index Pools	Shadowed
enduser	319	50406623	{3 }	{3 }	false

File Name:	enduser	Root Record (if -1 then on SQL):	50406623		
Field Name	Type	Index	Level	Keys	Data Pools:
descriptor	structure	1	0	no nulls	3
logical.name	character	1	1	customer.no	
serial.no.	character	2	1		
parent	character	3	1		
vtam.parent	character	4	1		
vtam.name	character	5	1	unique	3
location	character	6	1	logical.name	
fixed.asset.no.	character	7	1		
manufe					

Figure 10-19: New key added as first key

- 7 Press **OK** once you are finished adding keys to the Database Dictionary record, to close and automatically update the file. This returns the message *Record updated in the dbdict file.*

If records already existed in the file when you added a key, updating the Database Dictionary record will prompt you to confirm a regeneration of the file records.

- 8 You can either click **OK** (regen file) to continue with the regen, click **Cancel** (F3) to abort the regen and restore the Database Dictionary record to the version prior to the latest changes, or schedule a time for the file regeneration, using **Schedule** (clock button).

- 9 To remove the key, select it, and click **Edit**.
The edit window appears. You may ignore this.
- 10 Click **Delete** on the main record format.
A prompt is displayed asking you to confirm the deletion.
- 11 Click **Yes** to continue or **No** to abort the deletion.
The next viable key is displayed
- 12 Click **OK** if not modifying this key.
You are returned to the Database Dictionary record minus the key.
- 13 Click **OK** to save the record.
- 14 The Regen warning screen is displayed, asking you to confirm the action.

Inserting a key between other keys

The following screens illustrate how to insert the key `customer.no` between the keys `logical.name` and `serial.no.` in the `enduser` file.

Note: This is an example only, since we have already added `customer.no` to the Database Dictionary record in the last exercise. If you would like to perform this exercise, you should delete the `customer.no` key from the `enduser` Database Dictionary record before proceeding.

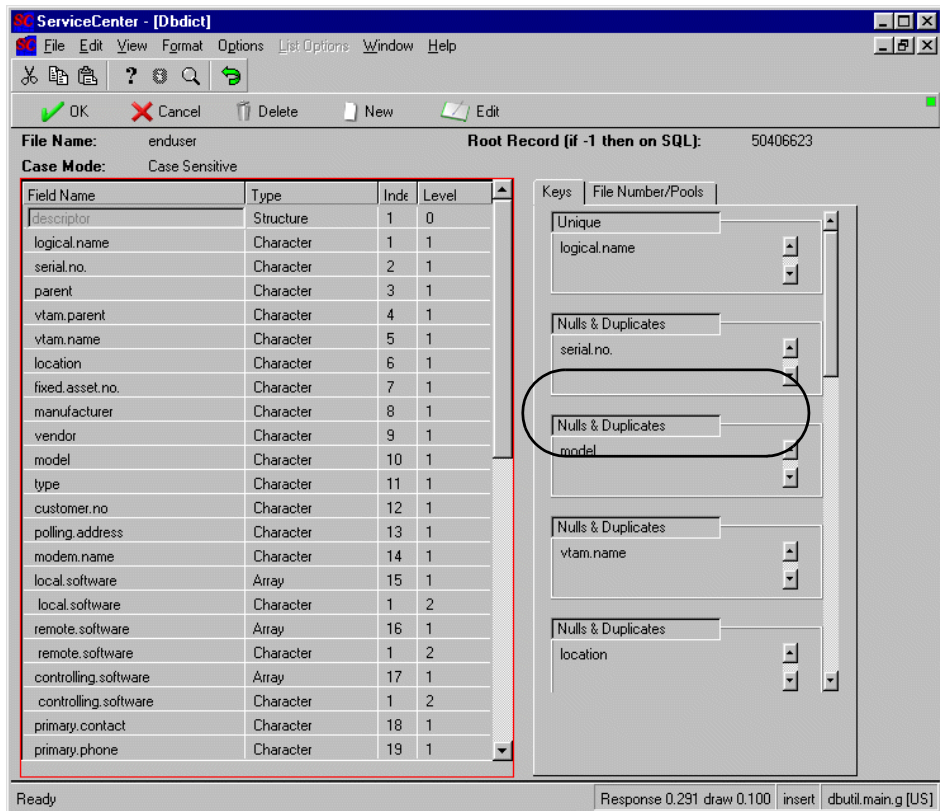


Figure 10-20: Inserting a key to the enduser Database Dictionary record

- 1 Click on the key type of the second key, *nulls&duplicates* in this example, above the `serial.no` key.
- 2 Click New.

- 3 In the new key window, select a valid key in the **Type** field by either typing in the value or selecting it from the drop down menu under the arrow button. Use *no nulls* for this example.
- 4 Enter *customer.no* in the field names array.
- 5 Click **Add** to add this key to the *enduser* Database Dictionary record.

The key *customer.no* now appears after the first key, in the second place in the list, and the *serial.no.* key has moved down the list of keys in the *enduser* Database Dictionary record. All other keys are moved down the screen. You can scroll the keys array to view the other keys in the list.

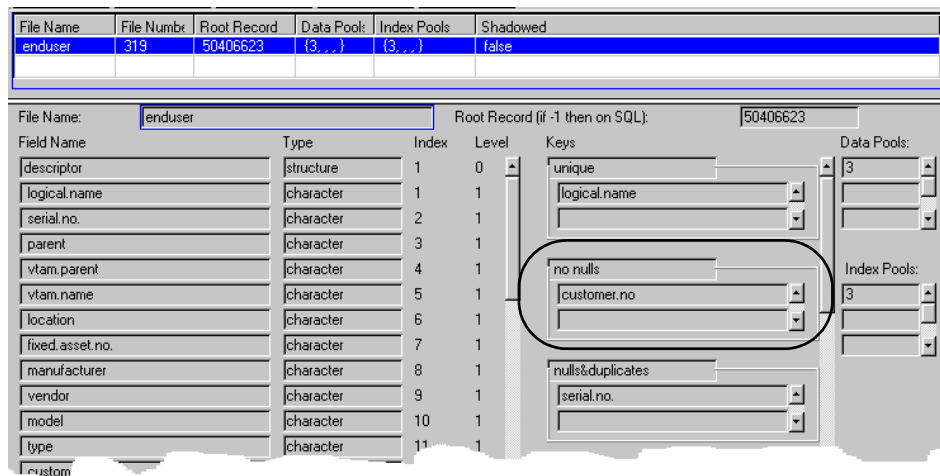


Figure 10-21: New key inserted between existing keys

- 6 You can now re-edit your Database Dictionary record, update it, or cancel without updating.

Note: You must update your Database Dictionary record when adding a new key, or the Database Dictionary record will remain the same.
- 7 Press **OK** once you are finished adding keys to the Database Dictionary record, to close and automatically update the file. This returns the message *Record updated in the dbdict file.*
- 8 You can either click **OK** (regen file) to continue with the regen, click **Cancel** (F3) to abort the regen and restore the Database Dictionary record to the version prior to the latest changes, or schedule a time for the file regeneration, using **Schedule** (clock button).
- 9 To remove the key, select it, and click **Edit**.

- 10 The edit window appears, but ignore the window, and instead click **Delete** on the main record format.
- 11 The next viable key is displayed, if not modifying this key click **OK**.
You are returned to the Database Dictionary record minus the key.
- 12 Click **OK** to save the record.

Adding keys to bottom of the key list

The following screens illustrate how to add a concatenated key comprised of the fields **customer.no** and **location** to the bottom of the key list.

- 1 Scroll down the keys until you reach the bottom of the list.

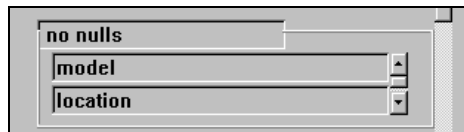
The last key in the *enduser* Database Dictionary record is **type**.

This is a single key because it is comprised of only one field. In the case of a concatenated key (regardless of how many fields comprise the key) only two fields will display in the key list.

To view all of the fields comprising a concatenated key, you must use the scroll arrow buttons to the right of the array or edit the key. Refer to *Modifying a Key* on page 331 for more information.

Note: The blank header represents the beginning of the next key. The number of lines displayed per key depends on the number of fields concatenated in that key.

Example: A key defined with the fields **model** and **location** would be displayed as:



2 Position the cursor on the blank header field located below the key type.

File Name	File Number	Root Record	Data Pool	Index Pools	Shadowed
enduser	319	50406623	{3...}	{3...}	false

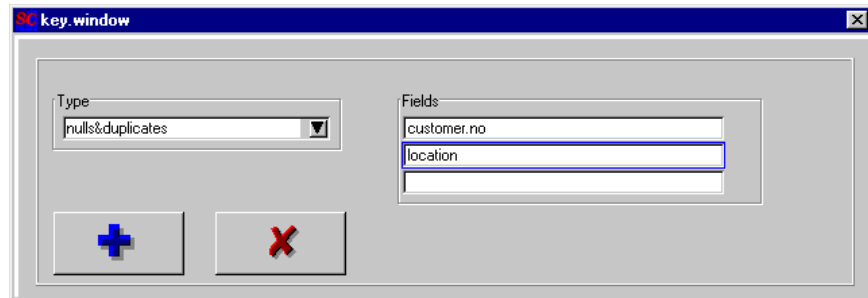
File Name:	enduser	Root Record (if -1 then on SQL):	50406623		
Field Name	Type	Index	Level	Keys	Data Pools:
descriptor	structure	1	0		3
logical.name	character	1	1	nulls&duplicates	
serial.no.	character	2	1	vendor	
parent	character	3	1		
vtam.parent	character	4	1		
vtam.name	character	5	1	nulls&duplicates	3
location	character	6	1	parent	
fixed.asset.no.	character	7	1		
manufacturer	character	8	1		
vendor	character	9	1	nulls&duplicates	
model	character	10	1	type	
type	character	11	1		
customer.no.	character	12	1		
polling.address	character	13	1		
modem.name	character	14	1		
local.software	array	15	1		
local.software	character	1	2		
remote.software	array	16	1		
remote.software	character	1	2		

Key added to the enduser file. insert dbutil.main.g [P]

Figure 10-22: Inserting a key at the end of the enduser Database Dictionary record

- 3 Click New to display the key definition window.
- 4 Enter nulls&duplicates in the Type field, or select *nulls&duplicates* from the drop-down list.
- 5 Press the Tab key to move to the fields array or place the cursor in the first field of the fields array.
- 6 Enter customer.no in the first field of the array.

- 7 Enter location in the second field of the fields array.



- 8 Click Add button to commit this new key to the database dictionary file.
The message *Key added to the enduser file* is displayed at the bottom of your screen.
The concatenated key comprised of **customer.no** and **location** now appears in the bottom of the list of keys in the *enduser* Database Dictionary record.

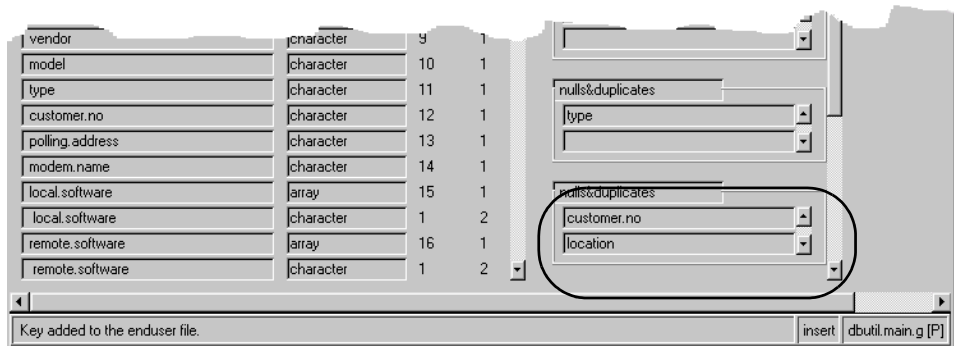


Figure 10-23: Concatenated key added to end of the enduser record

By placing the cursor in the appropriate position in the keys section of the Database Dictionary record, you can now press **New** to add additional keys to the Database Dictionary record without closing the window.

- 9 You can now re-edit your Database Dictionary record, update it, or cancel without updating.

Note: You must update your Database Dictionary record when adding a new key, or the Database Dictionary record will remain the same.

- 10 Press **OK** once you are finished adding keys to the Database Dictionary record, to close and automatically update the file. This returns the message *Record updated in the dbdict file.*
- 11 You can either click **OK** (regen file) to continue with the regen, click **Cancel** (F3) to abort the regen and restore the Database Dictionary record to the version prior to the latest changes, or schedule a time for the file regeneration, using **Schedule** (clock button).
- 12 To remove the key, select it, and click **Edit**.
- 13 The edit window appears, but ignore the window, and instead click **Delete** on the main record format.
- 14 The next viable key is displayed, if not modifying this key click **OK**.
You are returned to the Database Dictionary record minus the key.
- 15 Click **OK** to save the record.

Modifying a Field

There may be some instances when you want to modify field names or field types in a Database Dictionary record. When you modify a field name, you must change the associated input field on any forms (formats) associated with the file. Modifying a field name does not affect existing records in the file.

If you wish to modify a field type, certain considerations must be taken into account prior to any modifications:

- Under no circumstances should a scalar field be changed to an array or structure, or vice versa, regardless of whether or not data exists in the file.
- If no data exists in the file and you need to change a scalar field to an array or structure, or vice versa, simply delete the original field and add a new field defining the appropriate data type. Refer to *How to Add a New Field* and *How to Delete a Field* in this chapter.
- If data exists in the file and you need to change a scalar field to an array or structure (or vice versa) you must rename the original field (e.g., **old.address**) and add a new field (e.g., **address**) with the appropriate data type defined. You must then perform a *Mass Update* to move the contents of the original field (**old.address**) to the newly defined field (**address**). Refer to the *Database Manager* section of the *System Tailoring Guide* for additional information in the update procedure.

Modifying field types: character (scalar) <-> array

The following screens illustrate how to change a field with existing data, called address, from a character type scalar field to an array. Let's assume you created a new file called *vendorview* from the format vendor.

- 1 Select the **address** field name.
Only one entry for **address** should appear, as it is currently a scalar field.
- 2 Click **Edit**.
- 3 Enter `old.address` in the Name field of the pop-up window.
- 4 Leave the Type field value as *character* (Figure 10-24 on page 325).
- 5 Click **OK** to close the window and commit the change to the file.

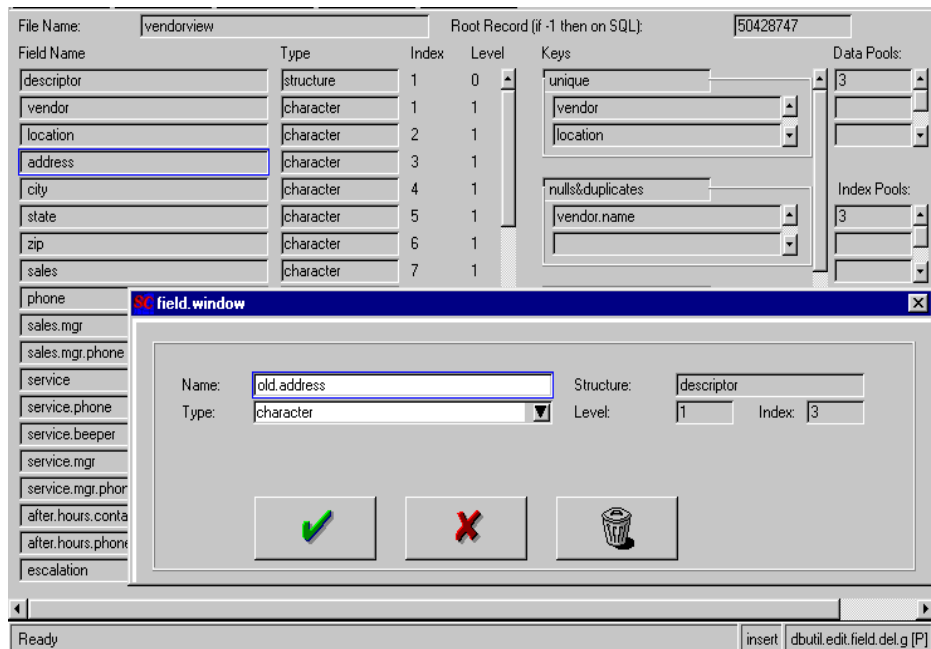
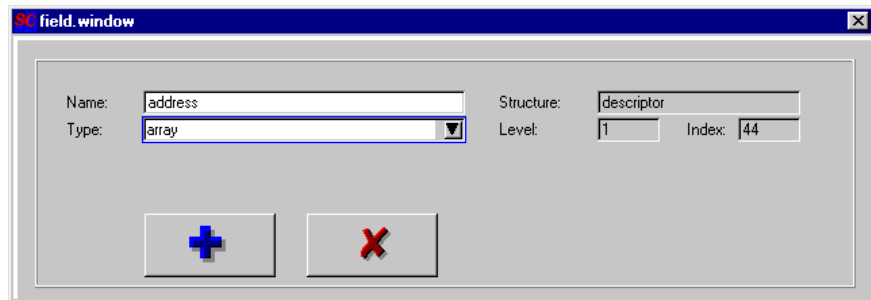


Figure 10-24: Modifying an existing field name

Notice the *address* field name is now *old.address* in the Field Name array.

- 6 Select the **descriptor** structure, and click **New**.
- 7 Enter *address* in the **Name** field of the pop-up window.
- 8 Enter *array* in the **Type** field.
- 9 Click **Add** to commit the new array to the file.

When adding an array field, you also need to specify the type of array (e.g., character, number, date/time, etc.). The system prompts you for this information now.



A new *field.window* appears with *address* displayed in the Name field and the Type field blank. The message *Enter data type of array's element* also appears at the bottom of the format window.

- 10 Select the **Type** field and enter *character*.

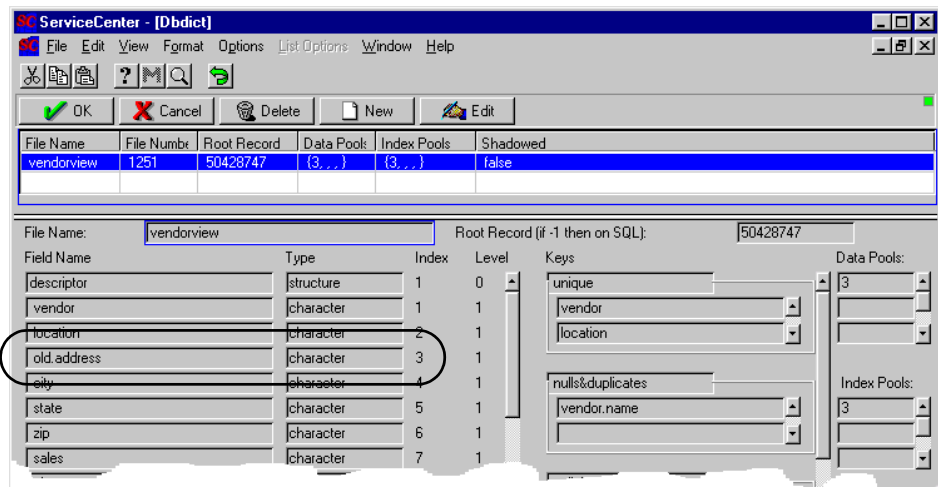


Figure 10-25: Renamed field

11 Click Add.

Note: Notice two entries for **address** now exist in the Database Dictionary record since it has been defined as a character array.

12 Click OK to close the window and save the changes.

13 Click OK to end this session with the vendorview file.

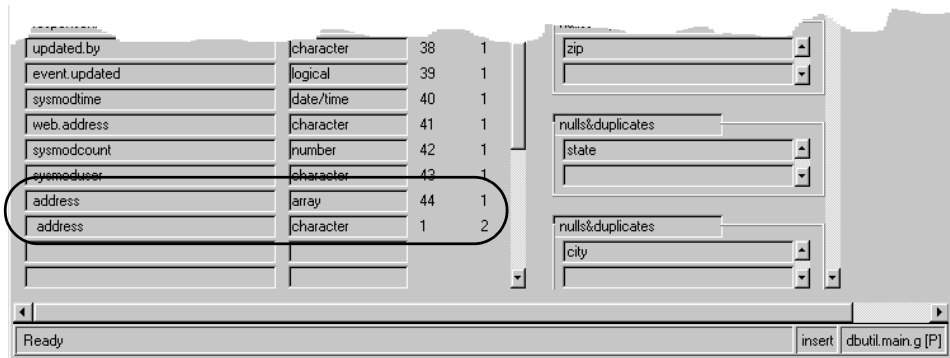


Figure 10-26: Added array fields

Note: This is not the end of the process, but because data exists in the file, you need to move the contents of the field now called **old.address** to the field called **address**. To accomplish this, you should perform a Mass Update in Database Manager on all records in the file that have data in the **old.address** field. Refer to the *Database Manager* section of the *System Tailoring Guide* for additional information in the Mass Update procedure. Once the data has been moved, it is recommended that you also remove the **old.address** field from the Database Dictionary record.

Mass Update

The Mass Update instruction string to move the data from the field called **old.address** to the first element in the newly created array called **address** is:

```
1 in address in $file = old.address in $file
```

Changing data types: *character* <-> *number*

You may want to change the type of a scalar field (e.g., from a *character* field to a *number* field).

- If no data exists in the file, you can edit the field and change the field type quickly without any other modification required.
- If data exists in the file, you must first change the type of the field in the Database Dictionary record, and then convert the existing data in the modified field to be consistent with the new field type. This must be done for all records in the file that contain data in the modified field.

The following example illustrate how to change the field called *zip* in the *vendorview* file from a *character* type field to a *number* type field. Let's assume that data exists in the file.

- 1 Select the *zip* field name.
- 2 Click Edit.

The field editing window is displayed.

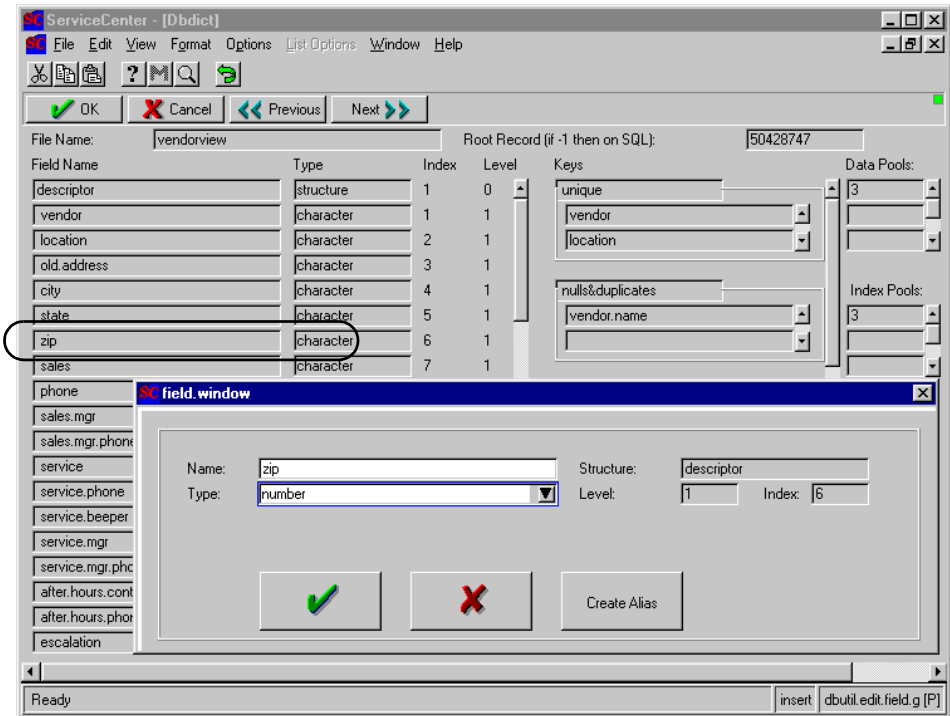


Figure 10-27: Modifying an existing field type

3 Enter number in the Type field.

This replaces *character* with *number* in the Type field.

- 4 Click OK to close the window and commit the change to the file.

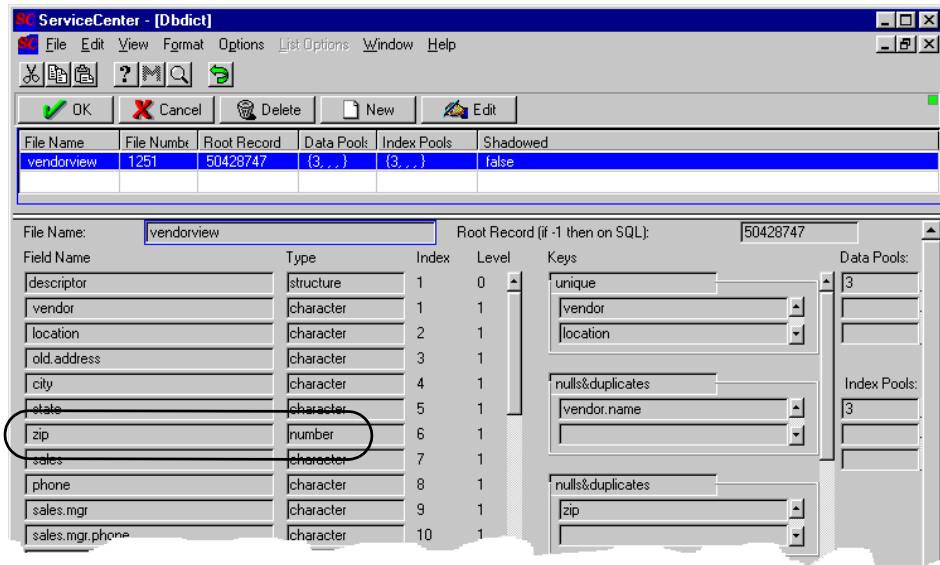


Figure 10-28: Existing field with modified data type

- 5 Click OK to update your Database Dictionary record.

The message *Record updated in the dbdict file* is displayed to confirm the update.

Note: This may not be the end of the process. If records exist in the file, you need to perform a Mass Update to change the field type of the **zip** field from *character* to *number* (the type of the field, e.g., character, number, date/time, etc., is stored in the data record with each associated value).

All records containing a value in the field **zip** need to be updated to reflect the new field type. Refer to the *Database Manager* section of the *System Tailoring Guide* for additional information in the Mass Update procedure.

Mass Update

The syntax used would be:

```
zip in $file=nullsub{val(zip in $file,1)zip in $file}
```

Note: If some of the records in the file contain data in the modified field that invalidates the new field type, (in the above example, records that have a character in the **zip** field that is now defined as a number field), the data will not be converted and a *cannot evaluate* message is generated for each record failing the conversion. Those records that do not fail the conversion will be updated.

Modifying a Key

As new users are added to ServiceCenter and additional requirements are specified, you may find the need to modify keys in your database dictionary records in order to optimize performance.

Note: As with adding new keys, if records already exist in the file when you modify a key, ServiceCenter automatically performs a regeneration of file data when you update the Database Dictionary record.

A modification to a key may be a change in the field(s) comprising the key or a change in the key type definition.

Important: When modify keys in a file that contains records, be careful when defining key types. For example, do not change a key from a *nulls & duplicates* type to *no nulls* if the field(s) comprising the key are all null (contain no value). In this case, each record that did not contain a value in the field(s) comprising the key would require an update, putting data in the field before the key could be changed to *no nulls*.

Modifying keys: *nulls & duplicates* <-> *unique*

The following example illustrate how to change the key type of *serial.no.* in the *enduser* file from *nulls & duplicates* to *unique*. If data exists in the file, you must ensure that a unique value exists in the *serial.no.* field for every record in the file. If this condition is not met, you will have to update each record individually or perform a Mass Update.

- 1 Position the cursor on the key type *nulls&duplicates* above the key field *serial.no.*
- 2 Click Edit.

- 3 Enter *unique* in the **Type** field, or select *unique* from the drop-down list (Figure 10-29 on page 332).

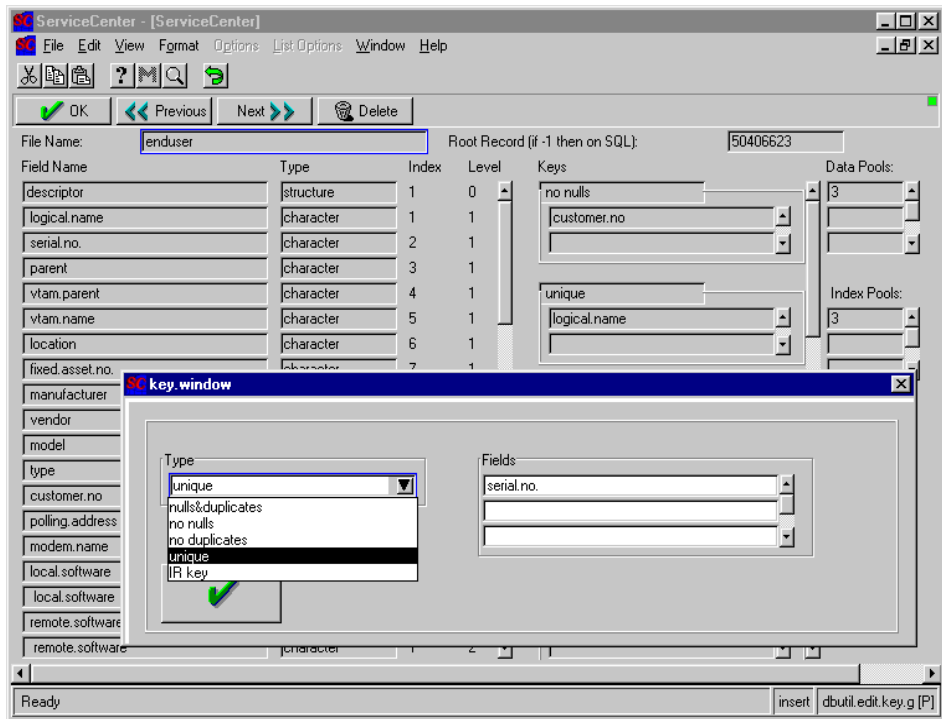


Figure 10-29: Modifying an existing key type

- 4 Click **OK** to close the window and commit the changes to the file.
- 5 You can now re-edit your Database Dictionary record, update it, or cancel without updating.
- 6 Click **OK** to close and automatically update the file.

This returns the message *Record updated in the dbdict file.*

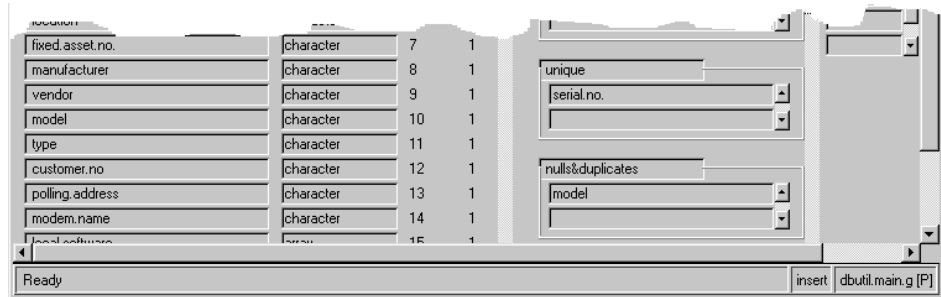


Figure 10-30: Existing key with modifying type value

- 7 If records already existed in the file when you modified a key, updating the Database Dictionary record will prompt you with this screen to confirm a regen on the file.
- 8 You can either click **OK** (regen file) to continue with the regen, click **Cancel** (F3) to abort the regen and restore the Database Dictionary record to the version prior to the latest changes, or schedule a time for the file regeneration, using **Schedule** (clock button).

Modifying keys: single <-> concatenated

The following screens illustrate how to modify the single key model in the *enduser* Database Dictionary record to make it a concatenated key comprised of the fields *model* and *location*.

- 1 Position the cursor on the key type *nulls&duplicates*, above the key *model*.
- 2 Click **Edit**.

3 Add *location* in the second element of the field names array below *model*.

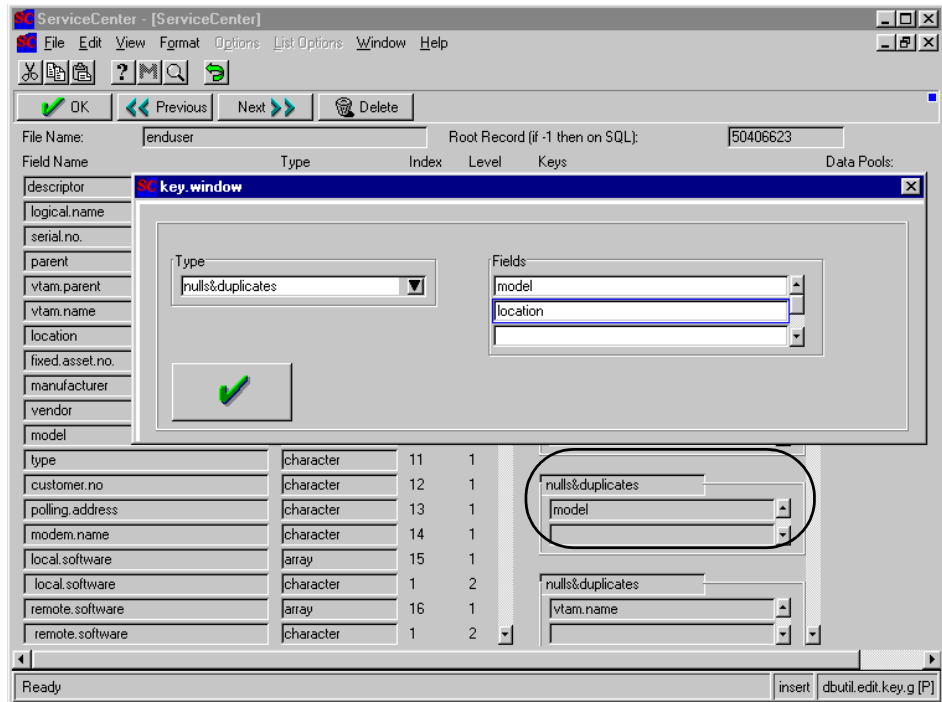


Figure 10-31: Adding concatenated value to existing key

4 Click OK to close the window.

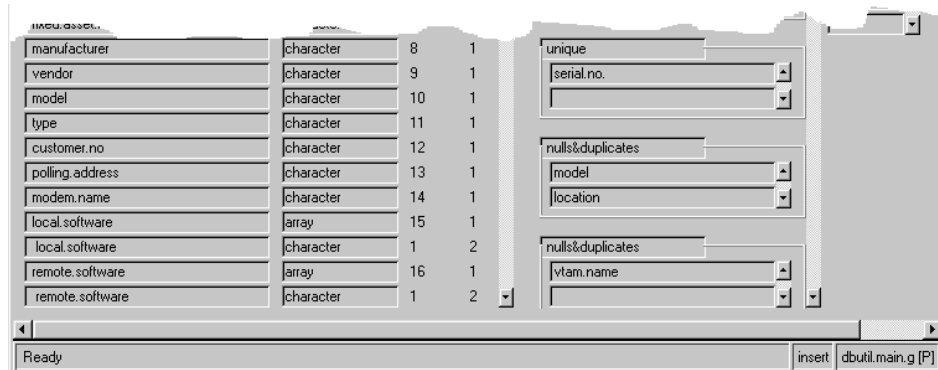


Figure 10-32: Existing key modified to become concatenated

- 5 You can now re-edit your Database Dictionary record, update it, or cancel without updating.
- 6 Click **OK**, to close and automatically update the file.
This returns the message *Record updated in the dbdict file*.
If records already existed in the file when you modified a key, updating the Database Dictionary record will prompt you to regenerate the file.
- 7 You can either click **OK** (regen file) to continue with the regen, click **Cancel** (F3) to abort the regen and restore the Database Dictionary record to the version prior to the latest changes, or schedule a time for the file regeneration, using **Schedule** (clock button).

Deleting a Field

After you have created a Database Dictionary record, you might decide that a field(s) originally defined in the dictionary is no longer needed. If no data exists in the file, you will see a delete option key on the screen when you edit the given field. You can delete the field from the Database Dictionary record. Also remember to delete the field from all associated formats.

Note: When you delete a field from a Database Dictionary record, you must check the key list to see if the field is a key or part of a key in the file. If so, you must modify or delete the key to remove the field name. Remember, all keys defined to the file must also exist as fields. The field should also be removed from any associated format(s).

If data exists in the file, you cannot delete a field(s) from the Database Dictionary record without first removing the data from that field. The delete key is not present when you edit the given field. If you do not want the field(s) and associated data in the file, you can perform a Mass Update to blank out (NULL) the data in the existing records. You can then rename the field(s) to depict that it is no longer in use (e.g., **old.field**, **old.field1**, etc.)

The following screens illustrate how to delete the field **old.vtam.name** from a copy of the *enduser* file, called *enduser1*. Let's assume this Database Dictionary record is a copy with no data existing in the *enduser1* file, meaning no records have been saved to this file.

- 1 Select the **old.vtam.name** field.

2 Click Edit.

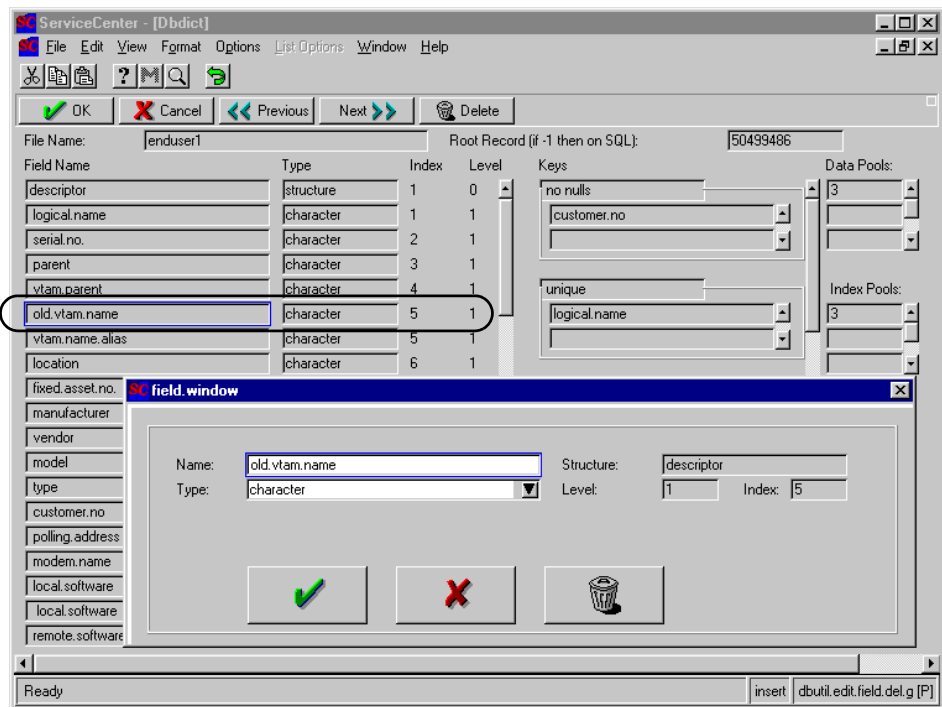


Figure 10-33: Deleting a field

- 3 Verify the field you selected is displayed in the pop-up window.
- 4 Click **Delete** either in the *field.window* or on the main Database Dictionary form to delete this field from the *enduser1* Database Dictionary record.
The delete operation requires a confirmation to remove the field.
- 5 Click **Yes** in the dialog box to confirm the delete.
The screen re-displays with the field **old.vtam.name** removed from the list of fields.
- 6 You can now re-edit your Database Dictionary record, update it, or cancel without updating.
Note: Remember to delete the field from the appropriate forms. Refer to *Forms Designer* for details on modifying forms.
- 7 Press **OK**, to close and automatically update the file. This returns the message *Record updated in the dbdict file.*

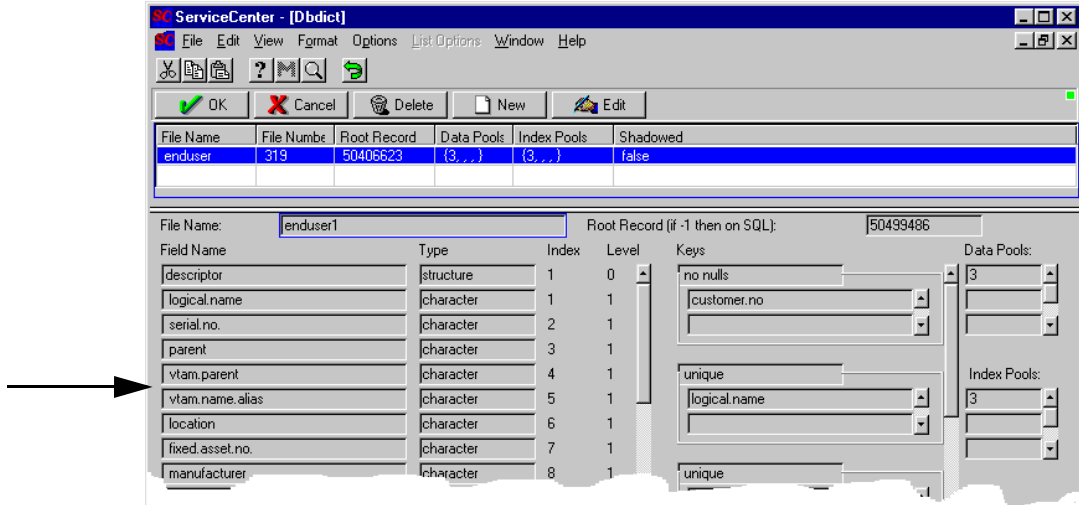


Figure 10-34: Field removed from record

Deleting a Key

Along with the need to modify existing keys, you might also want to delete keys no longer required. Remember, there is a direct correlation between the number of keys defined in a Database Dictionary record and the length of time it takes to add a record to that file.

If you delete a key in a Database Dictionary record, ServiceCenter automatically performs a file regeneration when you update the Database Dictionary record.

The following screens illustrate how to delete the *nulls & duplicates* key for old.vtam.name from the *enduser* Database Dictionary record.

- 1 Select the key type, *nulls & duplicates*, above the *vtam.name* key.
- 2 Click **Edit**.
- 3 Verify the key you selected is displayed in the window in the bottom of your screen.

- Click **Delete** on the main Database Dictionary format to delete this key from the *enduser1* Database Dictionary record.

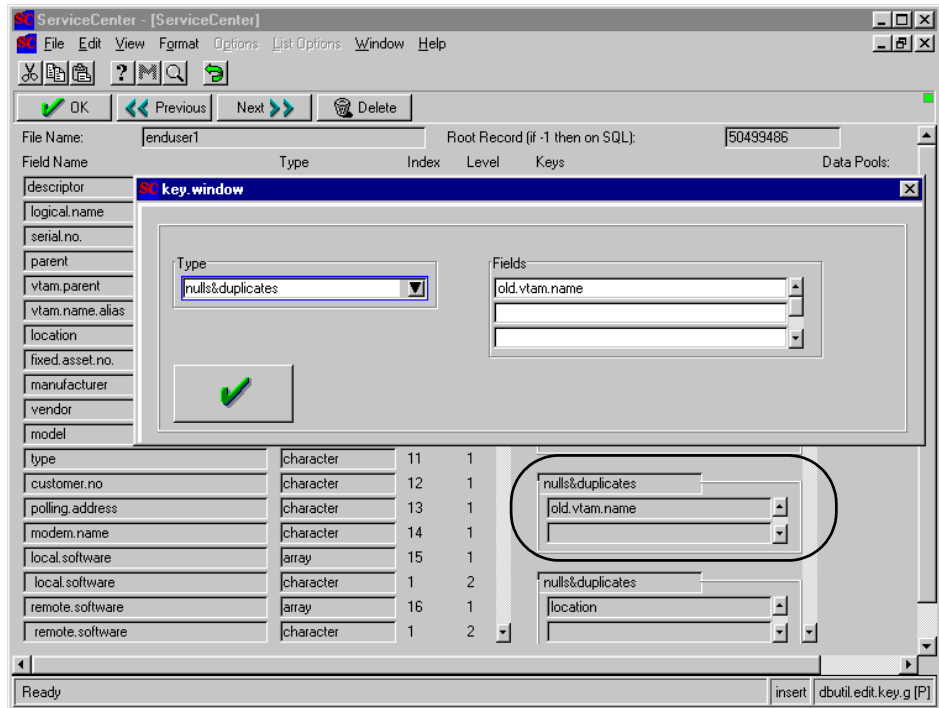


Figure 10-35: Deleting a key from record

A dialog box now appears, requiring a confirmation of the delete action.

- Click **Yes** to confirm the delete.

The pop-up window reappears with the next key and key type definition displayed (in this example, **model**).

- Click **OK** to close the window without modifying this key.

The screen re-displays with the key `old.vtam.name` removed from the list of keys and the keys below it moved up to take its former place.

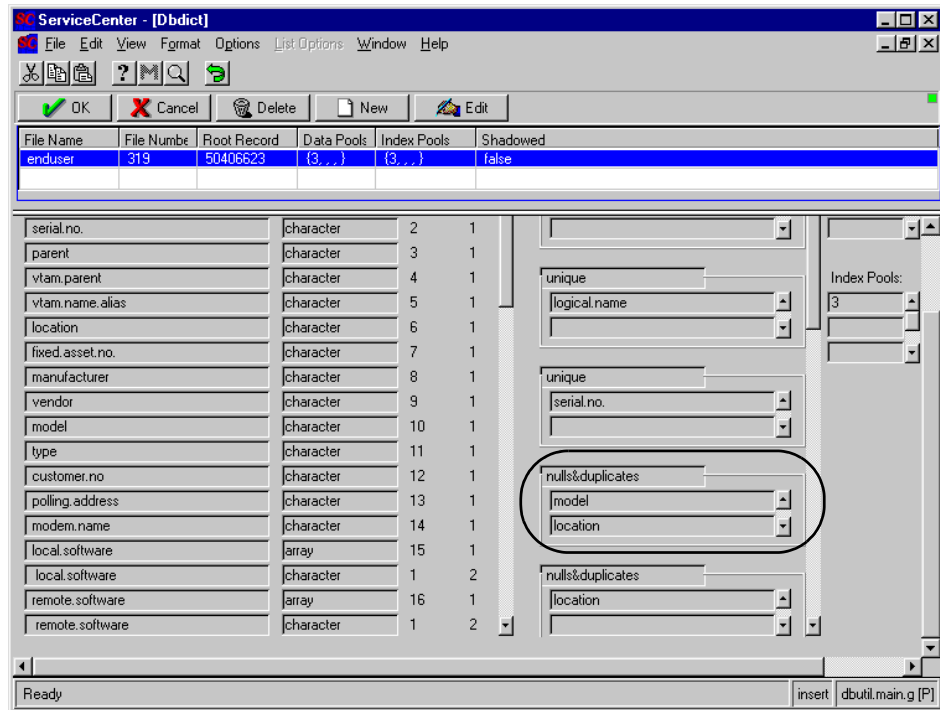


Figure 10-36: Record with key deleted

- 7 You can now re-edit your Database Dictionary record, update it, or cancel without updating.
- 8 Click **OK**, to close and automatically update the file.
This returns the message *Record updated in the dbdict file.*
- 9 You can either click **OK** (regen file) to continue with the regen, click **Cancel** (F3) to abort the regen and restore the Database Dictionary record to the version prior to the latest changes, or schedule a time for the file regeneration, using **Schedule** (clock button).

11 Building the Calendar

CHAPTER

The ServiceCenter Calendar is a collection of databases that support the definitions of working hours for different work groups. These work hour definitions are then used by various ServiceCenter applications to calculate Date Intervals and Alert Dates.

Calculating Alert Dates is the process of taking a known point in time and subtracting or adding (depending on the needs of the application) a known interval of time in order to determine the Alert Date.

Calculating Date Intervals is the process of taking two dates and finding the difference between them.

The calculation of Alert Dates and Date Intervals can be based on a default work hours (7 days/week x 24 hours/day) or on customized work hours (for example, 5 days/week x 8 hours/day). The ServiceCenter Alert Calculation routines allow you to define custom work hours for each Assignment Group within your organization or to define common work hours for multiple Assignment Groups.

When Alerts are processed, holidays and break times (non-work hours) are not included in the calculations. For example, consider the following:

- You have defined a shift with a start time of 8:00, and an end time of 5:00.
- You have defined a break time from 12:00 to 1:00 for this shift.
- An alert is scheduled to occur 4-1/2 hours after the start of the shift.

Result: The alert will occur at 1:30, because the system will not count the break time (non-work hours) in calculating when the Alert should occur. In other words, an alert will never occur during time you have designated as non-work hours.

The ServiceCenter Calendar uses the following databases:

- The *calholidays* database defines the starting and ending dates of valid company holidays.
- The *caldutyhours* database defines the normal daily working hours for a particular work or assignment group.
- The *caldaily* database contains a definition for each group for each day of the year (or any time frame you choose) and is used at run time to calculate Date Intervals and Alert Dates. (The Incident Management Unavailability routines also use this database.)

Note: The maintenance of these databases follows normal Database Manager procedures.

The objective of the ServiceCenter Calendar is to support the definition of the working hours for *groups of individuals*. It is not intended to accommodate the working hours of each individual operator. If you decide to use the Calendar to store individual operator working hours, you should first consider the large number of records that this might create. For instance, if you have 100 operators and you define one year's worth of *caldaily* records, you will have to manage 36,500 records. As operator numbers grow and date ranges expand, this number could become quite large and may end up causing performance incidents. Under normal circumstances, you will find that people who work in the same business area and work the same shift can be grouped together under one working table with no adverse effects on control.

Figure 11-1 on page 343 shows the process for creating a ServiceCenter Calendar.

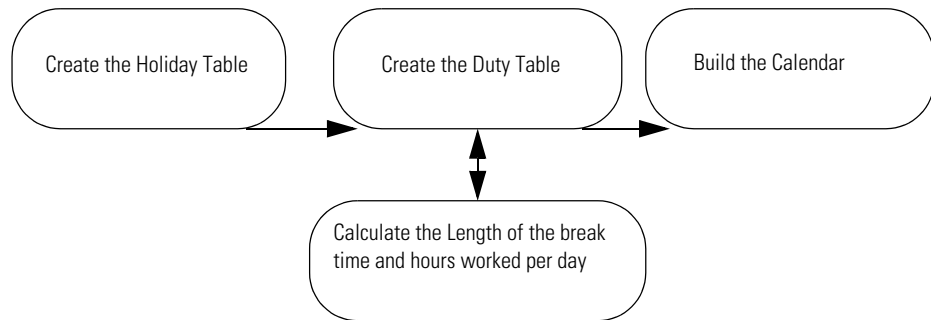


Figure 11-1: Process for Creating a ServiceCenter Calendar

The objective of the Calendar Build process is to use the information that you provided when you created the Duty Table and Holiday Table to generate a record in the `caldaily` database for each calendar day within a designated time frame.

Note: Due to the number of data integrity checks performed on each record generated to the `caldaily` database, the Calendar Build process can take one or two minutes to complete. You should avoid unnecessary executions of this routine and may want to consider running it during off-peak hours.

Preparing the Files

Before beginning the Calendar Build process, do the following:

- 1 Ensure that the Holiday Table is defined and correct (refer to *The Duty Hours Form* on page 354.).
- 2 Access the Duty Hours record and ensure that the fields are filled out correctly.

You are now ready to build the Calendar.

Building the Calendar

To build a new calendar:

- 1 With the Duty Hours form from which you want to build the calendar displayed, select *build* from the **Options** menu.

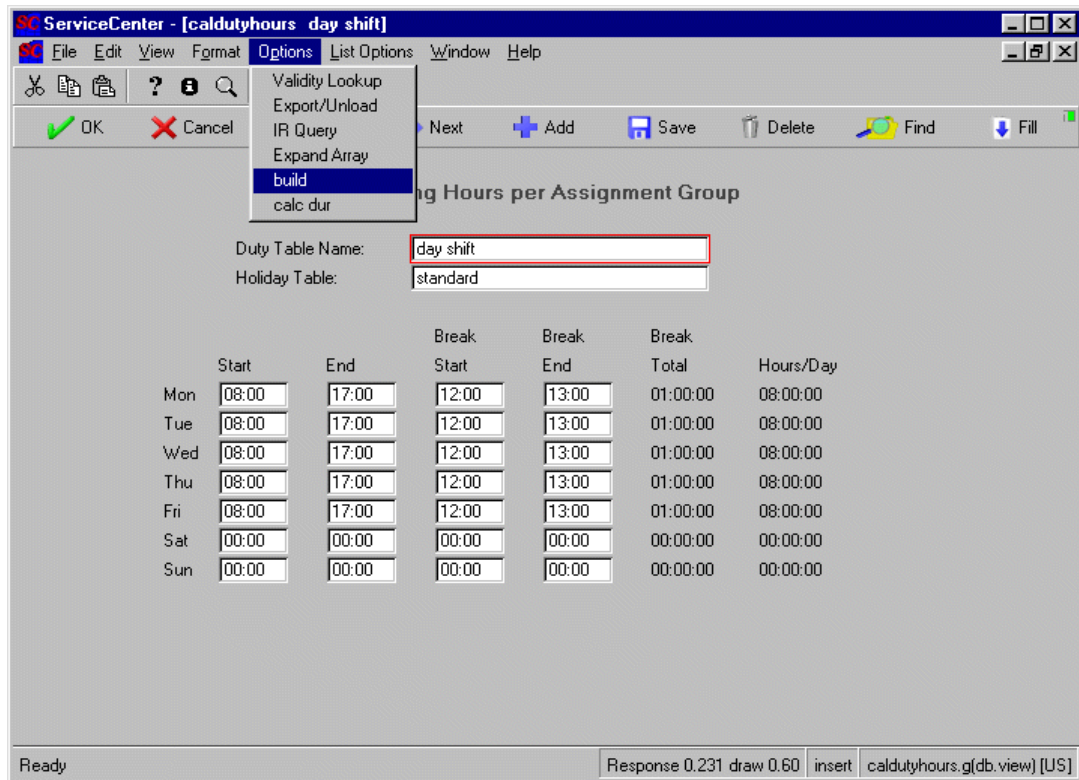


Figure 11-2: Duty Hours Form—Options Menu

The Build Daily Calendar form appears.

The **Duty Group** is the name of the Duty Group in the current Duty Hours record from which you want to build the calendar. Records created in the caldaily database will have this name.

The **Holiday Table** is the name of the Holiday Table you have indicated you want to be used when generating the caldaily records.

Warning: Any existing **caldaily** records for the indicated Duty Group that fall within the Range Start and Range End are deleted before new ones are created.

- 2 Type the **Range Start** (the first day of the Calendar) and **Range End** (the last day of the Calendar). The current date is filled in by default.

The Build process will create **caldaily** records for all dates that fall within the range of dates you have specified, including the Range Start and Range End dates.

- 3 Click **Build**.

The Calendar Build process may take several minutes to complete depending on the number of **caldaily** records that must be generated.

When the process is completed, a message appears in the Build Daily Calendar form, indicating that *All Calendar records for the indicated date range have been built*.

- 4 Click **End** to return to the Duty Hours form.

Processing rules

The following rules are in effect when generating **caldaily** records via the Calendar Build process:

- You must provide a Range Start and Range End.
- The Range End must occur after the Range Start.
- You must provide the name of the Duty Group (the Holiday Table name is optional).
- The Start, End, Break Start, Break End, Break Total and Hours/Day for each day of the week in the **caldutyhours** record cannot be left blank. If any of these fields are blank, the routine ends.
- The routine deletes any **caldaily** records that exist for the Duty Table which fall within the Range Start and Range End before adding new ones.
- If a Holiday begins or ends during what would normally be considered working hours for a particular shift, then the Shift Start (or Shift End) is updated to reflect the Holiday Start or Holiday End. For any day whose hours are adjusted by this process, the breaks are set to 00:00 (no breaks are accounted for).

For example, the normal working hours for the Duty Group split shift are 22:00 to 06:00, Monday through Friday.

- If Independence Day was defined as a Holiday to start on 07/04/yyyy 00:00 and end on 07/05/yyyy 00:00, then the caldaily record for 07/03/yyyy would have a Start Time of 07/03/yyyy 22:00 (normal start time) and an End Time of 07/04/yyyy 00:00 (the Holiday Start) and a Daily Duration of 02:00 (two hours).
- The caldaily record for 07/04/yyyy would have a Start Time of 07/05/yyyy 00:00 (the Holiday End) and an End Time of 07/05/yyyy 06:00 and a Daily Duration of 06:00 (six hours). Note that both the 07/03/yyyy and 07/04/yyyy records are updated to have a Holiday Name of Independence Day so that you will know which caldaily records have been adjusted due to a Holiday.

In order to avoid fragmenting a shift (like just described), adjust the Holiday Start and End Times to account for starting and ending times for the off shifts. For instance, in this example it may be better to define the Holiday Start Time as 07/04/yyyy 06:00 and an End Time of 07/05/yyyy 06:00 (this would mean that the split shift will work the evening of 07/03/yyyy but will not work 07/04/yyyy).

- If both the normal daily Start and End times fall within the time frames of a Holiday record, then that entire day is updated to show no working hours and the Holiday Name field is automatically updated to display the holiday name.

Viewing the Calendar

To view the new calendar you have created:

- 1 Return to the Calendar menu.
- 2 Click the **Daily Hours** button.
The Daily Hours form appears.
- 3 Enter the name of the calendar you want to view.
- 4 Press **Enter**.

The Daily Hours form appears, with the first item in the QBE Record list entered into the form. The Record list contains an entry for every day that was created by the calendar build.

Name	Date	Dayofweek	Dayofyear	Holiday
day shift	01/01/01 00:00:00	1	1	New Year's Day
day shift	01/02/01 00:00:00	2	2	
day shift	01/03/01 00:00:00	3	3	
day shift	01/04/01 00:00:00	4	4	
day shift	01/05/01 00:00:00	5	5	

CALENDAR EVENT ADD/EDIT

Duty Table Name:

Date:

Day of Week: 1 Monday

Day of Year: 1

Holiday Name:

Start Time:

Break Start:

Break End:

Break Duration: 00:00:00

End Time:

Daily Duration: 00:00:00

Weekly Total: 1 08:00:00

Selected line is row 1 of 32 records retrieved Response 0.140 draw 0.100 insert caldaily.g(db.view) [US]

Figure 11-3: The Completed Calendar

12

Calendar Forms

CHAPTER

There are three steps to creating a ServiceCenter Calendar:

- 1 Create a *Holiday Table* that will allow the Calendar to select certain days as holiday (non-work) days.
- 2 Establish the shift hours, break times, and non-work days using the *Duty Hours* form. This form will also allow you to calculate the break times and hours/day worked for each day of the shift.
- 3 Use the *Calendar Build* process to create a Daily Hours record for each day of the Calendar, using the parameters you have established.

This section includes instructions for creating a Holiday Table and establishing the work shift that will be used by the Calendar. The Calendar Build process is detailed in *Building the Calendar* on page 345.

Accessing the Calendar Menu

- 1 Log in as an administrator (for example, a *FALCON* login).
- 2 From the ServiceCenter main menu, click the **Utilities** tab. The **Utilities** menu appears.
- 3 Click the **Administration** button. The **Administration** menu appears.
- 4 Click the **Calendar** tab. The Calendar menu appears.

Click one of the buttons to access the Calendar forms:

- a **Daily Hours**—accesses a form where you can search the caldaily database for existing Calendar information.

- b **Duty Hours**—accesses a form where you can establish the start and end times of a shift, as well as the break times and hours/day worked.
- c **Holidays**—accesses a form where you can set up the holidays you want included in the Calendar.

Overview of Calendar forms

Daily Hours

The Daily Hours (`caldaily`) database stores information about each day of a particular shift record for the time frame you specify.

The Daily Hours records are created during the Calendar Build process based on the data entered in the Duty Hours and Holidays forms.

Although you can use the Daily Hours form to change an individual day in a Calendar, the need to do this probably would not occur. The Daily Hours form is used primarily to search the `caldaily` database.

To access the Daily Hours form:

- 1 Click the **Daily Hours** button on the **Calendar** menu.
The Daily Hours form appears.

The screenshot shows a window titled "ServiceCenter - [caldaily]" with a menu bar (File, Edit, View, Format, Options, List Options, Window, Help) and a toolbar (Back, Add, Search, Find, Fill). The main area is titled "CALENDAR EVENT ADD/EDIT" and contains the following fields:

- Duty Table Name: (highlighted with a red box)
- Date:
- Day of Week:
- Day of Year:
- Holiday Name:
- Start Time:
- Break Start:
- Break End:
- Break Duration:
- End Time:
- Daily Duration:
- Weekly Total:

The status bar at the bottom displays "Ready" on the left and "Response 0.130 draw 0.30 insert caldaily.g(db.search) [US]" on the right.

Figure 12-1: Daily Hours Form

The Daily Hours form fields

Duty Table Name—the name of the Duty Table from which this daily record was built.

Date—the date this record controls. The hours for this date/time field must be 00:00. Therefore, the time section of the field is not displayed.

Day of Week—the numerical representation and text description of the day of the week of this record. It is automatically calculated and cannot be manually updated.

The definitions for each day are:

1—Monday	5—Friday
2—Tuesday	6—Saturday
3—Wednesday	7—Sunday
4—Thursday	

Day of Year—the numerical representation of the day of the year of this record. It is automatically calculated and cannot be manually updated.

Holiday Name—the name of the Holiday that occurs on this day. The Calendar Build application automatically populates this field when it builds this record. You may also use this field for notes, or reminders, to yourself. This is a documentation field only and does not have a direct impact on processing.

Start Time—the starting date/time that the particular Duty Group is available for work. The date of this field must be the same date as defined in the Date field.

Break Start—the date/time that the non-working hours begin. This date/time must occur between the Start Time and End Time.

Break End—the date/time that the non-working hours end. This date/time must occur between the Start Time and End Time and must be greater than the Break Start.

Break Duration—the difference in time between the Break End and Break Start. This value is automatically calculated during manual add and update processing and is automatically calculated by the Calendar Build routine. Note that this field is protected from updating with a Format Control setting of 4.

End Time—the ending date/time that the particular Duty Group is available for work. The date of this field must be equal to the date defined in the Date field, or to the Date field plus 1 day (in other words, ‘1 00:00:00’).

Daily Duration—the value of this field is automatically calculated during add and update processing, and is automatically calculated by the Calendar Build routine. It first finds the difference in time between the **End Time** and **Start Time** and then subtracts the **Break Duration**.

Weekly Total—the total amount of time defined in the **Daily Duration** of each *caldaily* record for the upcoming week. A value is stored in this field only when the **Dayofweek** is 1 (Monday). The total includes the currently displayed Monday record.

The *caldaily* database

To search the *caldaily* database:

- 1 Type information in the Daily Hours form fields to access a particular record. Press **Enter**.

or

With the blank Daily Hours form displayed, press **Enter** to access a list of all the entries stored in the *caldaily* database.

A QBE Record list appears with the first entry in the list displayed in the Daily Hours form.

- 2 Click on the entry you want to use.
Information for that entry is filled into the form.

The Duty Hours Form

The Duty Hours form is used to add to or update the `cal-dutyhours` database. Shift start and end times, and break start and end times are defined, as well as non-work hours for the week.

You can also designate the name of the Holiday Table you want to use when the Calendar is built. Any work days that fall on holidays that you have established in the Holiday Table will be defined as holidays during the Calendar Build process.

The Duty Hours (`cal-dutyhours`) database allows you to store information about a particular shift, including the work shift start time and end time, and the break start time and end time.

You can designate which days are non-work days and indicate which Holiday Table you want used during the Calendar Build process. The Duty Hours form is also used to calculate the break total and the work hours/day.

To access the Duty Hours form:

- 1 Click the **Duty Hours** button on the Calendar menu.
The Duty Hours form appears.

ServiceCenter - [caldutyhours]

File Edit View Format Options List Options Window Help

Back Add Search Find Fill

Normal Working Hours per Assignment Group

Duty Table Name:

Holiday Table:

	Start	End	Break Start	Break End	Break Total	Hours/Day
Mon	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>		
Tue	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>		
Wed	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>		
Thu	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>		
Fri	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>		
Sat	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>		
Sun	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>		

Ready Response 0.151 draw 0.40 insert caldutyhours.g(db.search) [US]

Figure 12-2: Duty Hours Form

The fields on the Duty Hours form

Duty Table Name—a required field that defines the Duty Table (for example, *day shift*). Normally, this field reflects the name of the Assignment Group to which the table relates.

Holiday Table—the name of the Holiday Table used in conjunction with this Duty Table when generating the *caldaily* records for this Duty Table. Refer to *Chapter 5, The Calendar Build Process* for more information.

Start—the shift starting time for each day of the week.

End—the shift ending time for each day of the week.

Break Start—the break starting time for each day of the week. This is an optional field. If left blank, it defaults to 00:00.

Break End—the break ending time for each day of the week. This is an optional field as long as a Break Start is not defined for the particular day.

Break Total—the amount of time per day spent on break (non-working hours).

Hours/Day—the amount of time per day counted as working hours.

The caldutyhours database

To search the caldutyhours database:

- 1 Type the name of the record you want to access in the **Duty Table Name** field of the Duty Hours form to access a particular record. Press **Enter**.

or

With the blank Duty Hours form displayed, press **Enter** to access a list of all the entries stored in the caldutyhours database.

A QBE Record list appears with the first entry in the list displayed in the Daily Hours form.

- 2 Click on the entry you want to use.

The information for that shift is entered into the Duty Hours form.

Creating a new Duty Hours record

Consider the following information about the fields on the Duty Hours form before creating a new entry in the `calduthyhours` database:

Start—To define a non-working day for a shift, set the **Start** and **End** times to 00:00.

To define a full day for a shift (in other words, the Alert interval is based on 24 hours) set the **Start** time to 00:00 and the **End** time to 23:59. The **Start** time should reflect the actual day of the week that the shift starts.

End—To define a *non-working* day for a shift, set the **Start** and **End** times to 00:00.

To define a full day for a shift (in other words, the Alert interval is based on 24 hours) set the **Start** time to 00:00 and the **End** time to 23:59.

To enter a shift end that ends on the next day (for instance, Third Shift starts at 23:00 and ends at 07:00 the next day) enter the time that the shift ends (in this case, 07:00). The Calendar Build process automatically accounts for the extra day.

Break Start—Note that the break period is counted as non-working hours. The time defined in this field must fall within the shift **Start** and shift **End** times.

To define a shift that does not have a break period, define the **Break Start** as 00:00. Note that an Alert interval will never occur during a break period. For instance, assume that the following definitions exist in a `calduthyhours` record: **Start**: 08:00; **End**: 17:00; **Break Start**: 11:30; **Break End**: 12:30. If a ticket was opened at 10:00 and the Alert Stage 1 interval was 02:00 (2 hours), then the initial Alert time is calculated to be 12:00. However, since this time falls within the **Break Start** and **Break End** times, the Alert time is adjusted to 13:00 (original alert time - break start (12:00 - 11:30 = 00:30) + break end (12:30 + 00:30 = 13:00)).

Break End—If a **Break Start** is defined, then a **Break End** must also be defined. If left blank, it is defaulted to 00:00. Note that the break period is counted as non-working hours. The time defined to this field must fall within the Shift Start and Shift End times and must also occur after the **Break Start** time.

Break Total and Hours/Day—The *calc dur* option automatically calculates the **Break Total** and **Hours/Day**.

Note: If the Duty Table you are creating is for a shift that starts during one day and ends during the following day (the shift spans midnight), enter the Start and End times as usual, based on a 24-hour clock. For example, a shift that starts at 11:00 P.M. and ends the next day at 7:00 A.M. would have a Start time of 23:00, and an End time of 7:00. When the Break Total and Hours/Day calculations are made, the change from one date to the next is automatically accounted for.

To create a new Duty Hours record:

- 1 Click the **Utilities** tab on the ServiceCenter main menu.
- 2 Click the **Calendar** tab.
- 3 Click the **Duty Hours** button. The Duty Hours form appears.
- 4 Type a Duty Table Name.
- 5 Type the name of the Holiday Table you want to use in the **Holiday Table** field (*standard* in the example).
- 6 Type the shift start time, end time, break start time, and break end time for each work day. The format is hh:mm, using a 24-hour clock. For non-work days, enter 00:00 for all start and end times.
- 7 To calculate the Break Total and Hours/Day, select *calc dur* from the Options menu.

The break totals and hours/day for each work day are automatically totaled.

Important: Do not leave any fields blank. A null field will stop the Calendar Build process.

The sample completed form is shown below.

- 8 Click **Add** to add this Duty Table to the caldutyhours database.

Updating an existing entry

To change an existing Duty Hours entry:

- 1 Click the **Utilities** tab on the ServiceCenter main menu.
- 2 Click the **Administration** button.
- 3 Click the Calendar tab.
- 4 Click the **Duty Hours** button. The Duty Hours form appears.
- 5 Search for the record you want to modify:
 - a Type the name of the record you want to modify. Press **Enter**.
or
 - b With the blank Duty Hours form displayed, press **Enter**. The Duty Hours form appears with a QBE Record list of all Duty Hours records. The first entry in the list appears in the form.

Select the Duty Table Name you want to update. In the example, *day shift* has been selected. The data from that record is entered into the Duty Hours form.
- 6 Make the desired changes to the Duty Hours fields.

Important: If you make changes that affect the number of hours worked, the length of the break time, or change the status of a day (for example, work to non-work), the hours/day and break total must be recalculated as follows:

Select *calc dur* from the Options menu. The break totals and hours/day for each work day are automatically totaled.

- 7 Click **Save**. The updated record is saved to the *calduyhours* database.

Deleting an existing entry

To delete a shift record from the *calduyhours* database:

- 1 Select the shift you want to delete in the Duty Hours Record list.
- 2 Click **Delete**. A prompt appears asking if you want to delete this record.
- 3 Click **Yes**.

The Duty Hours form is redisplayed with the shift deleted from the Record list. A message appears in the system status bar, *Record deleted from the calduyhours file*.

Holidays

From the Holiday Names form, you can search the `calholidays` database for existing holidays or Holiday Tables. You can also edit existing Holiday Tables, or create new tables.

ServiceCenter is shipped with a Holiday Table called `standard`. The `standard` Holiday Table contains the following holidays:

New Year's Day	Independence Day
President's Day	Labor Day
Memorial Day	Thanksgiving
	Christmas

To access the Holiday Names form:

- 1 Click the **Holidays** button on the **Calendar** menu.
The Holiday Names form appears.

The screenshot shows a software window titled "ServiceCenter - [calholidays]". At the top is a menu bar with "File", "Edit", "View", "Format", "Options", "List Options", "Window", and "Help". Below the menu bar is a toolbar with icons for Cut, Copy, Paste, Help, Find, and Refresh. A second toolbar contains buttons for "OK", "Cancel", "Previous", "Next", "Add", "Save", "Delete", "Find", and "Fill".

Holiday	Start Date	End Date
Memorial Day	05/28/01 00:00:00	05/29/01 00:00:00
Independence Day	07/04/01 00:00:00	07/05/01 00:00:00
Labor Day	09/03/01 00:00:00	09/04/01 00:00:00
Thanksgiving	11/22/01 00:00:00	11/24/01 00:00:00
Christmas	12/24/01 00:00:00	12/26/01 00:00:00

Below the table is a section titled "Holiday Names" containing a form with the following fields:

- Holiday Name:
- Start Date:
- End Date:
- Holiday Tables:

At the bottom of the window, a status bar displays "Selected line is row 1 of 7 records" on the left, "Response 0.120 draw 0.80 insert" in the middle, and "calholidays.g(db.view) [US]" on the right.

Figure 12-3: Holiday Names Form

The fields on the Holiday Names form

Holiday Name—the name of the Holiday. Examples of Holiday Names are New Year's Day and Independence Day. This is a required field.

Start Date—a required field that defines the starting date/time of the Holiday.

End Date—a required field that defines the ending date/time of the Holiday.

Holiday Tables—an arrayed character field that defines the Holiday Tables that use this Holiday. A Holiday Table Name is a descriptive term that allows you to group together Holidays that apply to a particular area.

The calholidays database

To search the calholidays database:

- 1 Type the name of the record you want to access in the calholidays **Holiday Name** field to access a particular record. Press **Enter**.

or

With the blank Holiday Names form displayed, press **Enter** to access a list of all the entries stored in the calholidays database.

A QBE Record list appears with the first entry in the list displayed in the Holiday Names form

- 2 Click on the entry you want to use.

Creating a new holiday entry

Consider the following information about the fields on the Holiday Names form before creating a new Holiday entry:

Holiday Name—You can use the same Holiday Name more than once, but define each occurrence of the name to have different start dates and end dates. For instance, you could define New Year's Day with a Holiday Table of *standard* and a Start Date of 01/01/yyyy 00:00 and an End Date of 01/02/yyyy 00:00, and then define a different New Year's Day with a Holiday Table of *special* and a Start Date of 12/31/yyyy 22:00 and an End Date of 01/01/yyyy 22:00. There is no limit to the number of Holiday definitions that you can define for a particular Holiday.

Start Date—New Year's Day normally starts on January 1 at midnight (the start of the day). Therefore, the Start Date is 01/01/yyyy 00:00. Note that the Holiday can begin at a time other than midnight. For instance, if Third Shift normally works the hours of 22:00 (10:00 P.M.) to 07:00 (A.M.), it would then be appropriate to have the Start Date for New Year's Day set to 12/31/yyyy 22:00. If the next shift to work is Third Shift and they will work their normal hours, then the End Date (explained below) would be 01/01/yyyy 22:00.

It is not necessary to include a time in the Start Date and End Date fields; the time will default to 00:00 if left blank. If you want to add a time (for example, for a shift that will work part of a holiday), type the time in the format *hh:mm*.

End Date—New Year's Day normally ends on January 2 at midnight (the start of the day). Note that *caldaily* records built for the dates that have shift starts and shift ends within the date range of the *Holiday* record will not have any working hours.

It is also possible for Holidays to span more than 24 hours. For instance, you could define an entry for Christmas to start on 12/24/yyyy 00:00 and end on 12/26/yyyy 00:00.

To create a new Holiday entry:

- 1 Click the **Utilities** tab on the ServiceCenter main menu.
- 2 Click the **Administration** button.
- 3 Click the **Calendar** tab. The Calendar menu appears.
- 4 Click the **Holidays** button. The Holiday Names form appears.
- 5 Type the data for the holiday you wish to create.
- 6 When all the fields have been completed, click **Add**.

The new record is added to the *calholidays* database. If this is the first entry created using the Table name of *special*, a new Holiday Table is created with this name.

Note: You can add a holiday to more than one Holiday Table by listing the tables in the **Holiday Tables** field.

- 7 Click **OK** to return to the initial Holiday Names form. Repeat steps 5 and 6 for each holiday you want to create.

Updating an existing holiday entry

To change an existing Holiday entry:

- 1 Click the **Utilities** tab on the ServiceCenter main menu.
- 2 Click the **Calendar** tab.
- 3 Click the **Holidays** button. The Holiday Names (**calholidays**) form appears.
- 4 Enter the name of the holiday record you want to modify (New Year's Day in the example below), or press **Enter** to display a QBE Record list of all holidays in the **calholidays** database.

Note: Only the holiday name is required to search the database. However, to narrow your search, you can add other information, such as the start or end dates, or the Holiday Tables name, if known.

- 5 Press **Enter**.
- 6 Click on the entry you want to modify, and the information will be entered into the Holiday Names form.

In the example, select the New Year's Day entry.

- 7 Change the data as required.

In the example, the **End Date** has been changed to 01/03/2000.

The screenshot shows a window titled "ServiceCenter - [calholidays]" with a menu bar (File, Edit, View, Format, Options, List Options, Window, Help) and a toolbar with icons for OK, Cancel, Previous, Next, Add, Save, Delete, Find, and Fill. Below the toolbar is a table with the following data:

Holiday	Start Date	End Date
New Year's Day	01/01/01 00:00:00	01/02/01 00:00:00
New Year's Day	01/01/01 00:00:00	01/03/01 00:00:00
President's Day	02/19/01 00:00:00	02/20/01 00:00:00
Memorial Day	05/28/01 00:00:00	05/29/01 00:00:00
Independence Day	07/04/01 00:00:00	07/05/01 00:00:00

Below the table is a form titled "Holiday Names" with the following fields:

- Holiday Name:
- Start Date:
- End Date:
- Holiday Tables:

At the bottom of the window, the status bar displays: "Selected line is row 2 of 8 records" and "Response 0.90 draw 0.70 insert calholidays.g(db.view) [US]"

Figure 12-4: Updated Holiday Names Form

8 Click Save.

A message appears in the status bar that the record has been updated in the calholidays file.

Deleting an existing holiday entry

To delete an existing holiday:

- 1 Use the Holiday Names form to search for the holiday record you want to delete.
- 2 With the record displayed, click Delete.
A prompt appears asking if you want to delete this record.
- 3 Click Yes.

A message appears in the status bar that the record has been deleted from the calholidays file.

Creating a new Holiday table

You can create a new holiday table in the same way that you created a new holiday entry in an existing table (see “Creating a new holiday entry” on page 15):

- 1 Type the data for the holiday in the fields on the Holiday Names form.
- 2 In the **Holiday Tables** field, type a unique name of your choice.
- 3 Click **Add**.
A new table is created.
- 4 Repeat steps 1 through 3 to add more holidays to the table.

The new table will have only the holiday you created, and each holiday that you want to include in the new table must be entered individually.

13

CHAPTER

Calculating Dates and Times

This section describes using RAD routines or Format Control to specify dates and intervals for the Calendar function.

The Alert Date Calculation Process

The Alert Date Calculation application takes a known date and calculates an ending date by applying a known interval to the duty hours defined to a particular duty table in the `caldaily` file. You can invoke the process by executing `calendar.calc.interval` via a RAD subroutine call or executing the `calendar.calc.interval.fc` as a Format Control Additional Option or Subroutine call (refer to the *RAD Guide* or to the *Format Control chapter of this manual*).

RAD subroutine example

This is an example of an application that calls the `calendar.calc.date` application via a standard RAD subroutine call.

The parameter input fields are described below.

Starting Date

Field: times,1 The date/time the Alert calculation process uses as the starting point in time. This parameter must be type 3 data (date/time). Normally, this is either `tod()` or the date/time of the open or update time of a ServiceCenter record. If a date/time is not passed to this field, the routine ends and no calculations are performed.

Interval

Field: times,2 The relative amount of time for the **Alert Interval**. This parameter must be a type 3 (date/time). If a value is not passed, the **Modified Date** is set to the **Starting Date**. Note that you can pass a negative interval of time. This is beneficial when it is necessary to calculate the date/time so many hours before an event is scheduled to occur (for instance, x days and hours before the `planned.end` of a Change).

Duty Table

Field: prompt The `caldaily` records that have a **Duty Table** of this name are used to calculate the **Alert Date**. If no `caldaily` records are found with this name, the **Modified Date** is set to the **Starting Date** plus **Interval** based on a 24 hour/day clock.

Modified Date

Field: times, 3 This is the date/time that is calculated by adding the **Interval** to the **Starting Date** and using the Duty Table records. Data is returned through this field.

Note: 1. Do not pass data to the other parameters on this panel!

2. The normal exit should continue to the next panel in the logical process flow. The error exit should return to the last displayed input/output panel (typically either an `rio`, `fdisp`, or `display`).

Format Control example

This is an example of a format control record that calls the application `calendar.calc.date.fc` via a subroutine call. Note the data field names are for demonstration purposes only. The format has also been modified to show all parameters that must be passed to this application.

The parameter input fields are described below.

times,1

The starting date/time used by the Alert calculation process. Normally, this is either `tod()` or a date/time that represents the beginning of some process (for instance, the `open.time` of an incident ticket or change request). If a date/time is not passed to this field, the routine ends and no calculations are performed. This parameter must be type 3 data (date/time).

times,2

The interval of time used by the Alert calculation process. If a date/time is not passed to this field, the routine ends and no calculations are performed. This parameter must be type 3 data (date/time).

prompt

The name of the Duty Table (`caldaily`) records that are used to calculate the **Date Interval**. If no `caldaily` records are found with this name, the **Ending Time** is set to the **Starting Date** time plus the **Date Interval** based on a 24 hour/day clock.

file

The current format control file variable. You will always pass *\$file* to this parameter.

name

The input field name within the *\$file* file variable where the calendar calculation routine will store the results of the calculation process (in other words, the ending time). You must pass the input field name exactly in the `dbdict` which defines the *\$file* variable.

Processing Rules

The following rules are in effect when this routine is executed:

- If a start date is not passed to the routine, it is defaulted to the current date and time (i.e., `tod()`).
- If an interval is not passed to the routine, the Ending Time is set to the start time and the routine ends.
- If a duty table is not passed to the routine, the Ending Time is set to the start time + interval using a 24 hour/day clock.
- If no `caldaily` records are found that have the Duty Table name and have a date greater than or equal to the start time, the Ending Time is set to the start time + interval using a 24 hour/day clock.
- If the start time is less than the date of the first `caldaily` record found for the specified duty group, the interval of time from the start time to the date of the first `caldaily` record is based on a 24 hour/day clock.
- If any skips are encountered in the `caldaily` records (in other words, the difference in dates between two concurrent records is greater than 1 day, for instance if the current record has a date of 01/15/92 00:00 and the next record has a date of 01/25/92 00:00), the interval of time between dates is counted as 24 hours/day.
- The modified date will never be set to a time that occurs between a break start and break end defined to a `caldaily` record. If the originally calculated ending time does occur between the break times, then the following calculations occur. The difference in time between the originally calculated ending time minus the break start is added to the break end. The ending time is set to the break end. For instance, if the calculated time is 11:30:00, and the break start is 11:00:00 and the break end is 12:00:00, then the ending time is set to 12:30:00.

Alert Date calculation examples

The table below is a summary listing of a typical representation of `caldaily` records for the Duty Table day shift. These dates are used in the calculation process for the examples shown.

Date	Day of Week	Shift Start	Shift End	Break Start	Break End	Holiday
01/01/92	3	-	-	-	-	New Year's Day
01/02/92	4	08:00	17:00	11:00	12:00	
01/03/92	5	08:00	17:00	11:00	12:00	
01/04/92	6	-	-	-	-	
01/05/92	7	-	-	-	-	
01/06/92	1	08:00	17:00	11:00	12:00	
01/07/92	2	08:00	17:00	11:00	12:00	
01/08/92	3	08:00	17:00	11:00	12:00	
01/09/92	4	08:00	17:00	11:00	12:00	
01/10/92	5	08:00	17:00	11:00	12:00	
01/11/92	6	-	-	-	-	
01/12/92	7	-	-	-	-	
01/13/92	1	08:00	17:00	11:00	12:00	
01/14/92	2	08:00	17:00	11:00	12:00	
01/15/92	3	08:00	17:00	11:00	12:00	
01/16/92	4	08:00	17:00	11:00	12:00	
01/17/92	5	08:00	17:00	11:00	12:00	
01/18/92	6	-	-	-	-	
01/19/92	7	-	-	-	-	

	Start	Interval	Duty Table	Modified End Date
1	12/15/91 14:00	04:00	day shift	12/15/91 18:00
2	12/31/91 22:00	06:00	day shift	01/02/92 13:00
3	01/02/92 11:30	02:00	day shift	01/02/92 14:00

4	01/02/92 00:00	02:00	day shift	01/02/92 10:00
5	01/03/92 15:00	3 00:00	day shift	01/16/92 15:00
6	01/05/92 11:00	1 04:30	day shift	01/09/92 13:30

- 1 The start date is before the earliest defined caldaily record. Therefore, the calculation is based on 24 hours/day.
- 2 The start date is 02:00 (2 hours) before the earliest defined caldaily record. Therefore 2 hours are applied to the interval, and 4 hours remain. The date 01/01/92 is a holiday with no working hours. The date 01/02/92 starts at 08:00. The time from 08:00 to 11:00 is counted (3 hours). The break times from 11:00-12:00 are not counted. The time from 12:00 to 13:00 is counted (1 additional hour) and the interval of 6 hours is reached.
- 3 The start date falls within the defined break hours. Therefore, the start time is adjusted forward to the end of the break (12:00). The interval of 02:00 is added to 12:00 to arrive at 14:00.
- 4 The start date occurs before the start of the shift. Therefore, the start time is adjusted forward to the beginning of the shift (08:00). The interval of 02:00 is added to 08:00 to arrive at 10:00.
- 5 The start date is 2 hours before the end of the Friday shift. This time is subtracted from the interval, leaving 2 22:00:00. The next week is defined to account for 40 hours. Subtracting 40 hours defined for that week leaves an interval of 1 06:00 (30 hours). The dates 1/13/92, 1/14/92 and 1/15/92 account for 24 hours (8 hours each day). The remaining 6 hours are applied to 1/16/92 with 3 hours occurring in the morning (11:00 - 08:00) and 3 hours occurring in the afternoon (15:00 - 12:00).
- 6 The start date occurs on a Sunday. Therefore, the start time is adjusted forward to the start of the shift on Monday. Monday, Tuesday and Wednesday account for 24 hours (8 hours each day) leaving 04:30. The first 3 hours occur in the morning (11:00 - 08:00) and the remaining 01:30 is added to the end of the break period (12:00) arriving at 13:30.

The Date Interval Calculation Process

The Date Interval Calculation application takes two known dates and calculates the interval between them by adding the daily intervals of the `caldaily` records whose dates are within the range of the known dates. You can invoke the process by executing `calendar.calc.interval` via a RAD subroutine call or `calendar.calc.interval.fc` as a Format Control Additional Option or Subroutine call.

RAD subroutine example

This is an example of an application that calls the `calendar.calc.interval` application via a standard RAD subroutine call.

The parameter input fields are described below.

Starting Date

Field: times,1 The date/time the Alert calculation process uses as the starting time. Normally, this is either `tod()` or a date/time that represents the beginning of some process (for instance, the `open.time` of an incident ticket or change request). If a date/time is not passed to this field, the routine ends and no calculations are performed.

Interval

Field: times,2 The relative amount of time between the Starting Date and the Ending Date. Do not pass data to this parameter.

Duty Table

Field: prompt The `caldaily` records that have a Duty Table of this name are used to calculate the Date Interval. If no `caldaily` records are found with this name, the Interval is set to the Ending Date minus the Starting Date based on a 24 hour/day clock.

Ending Date

Field: times, 3 The date/time the Alert calculation process uses as the ending point in time. Normally, this is a date/time that represents the end of a process (for instance, the `close.date` of an incident ticket or change request). If a date/time is not passed to this field, the routine ends and no calculations are performed.

- Note:** 1. Do not pass data to the other parameters on this panel!
2. The normal exit should continue to the next panel in the logical process flow. The error exit should return to the last displayed input/output panel (typically either an `rio`, `fdisp`, or `display`).

Format Control example

This is an example of a format control record that calls the application `calendar.calc.interval.fc` via a subroutine call. Note that the data field names are for demonstration purposes only. The format has also been modified to show all parameters that must be passed to this application.

The parameter input fields are described below.

times,1

This is the starting date/time used by the Alert calculation process. Normally, this is either `tod()` or a date/time that represents the beginning of some process (for instance, the `open.time` of an incident ticket or change request). If a date/time is not passed to this field, the routine ends and no calculations are performed.

times,2

This is the ending date/time used by the Alert calculation process. If a date/time is not passed to this field, the routine ends and no calculations are performed.

prompt

This is the name of the Duty Table (`caldaily`) records that are used to calculate the Date Interval. If no `caldaily` records are found with this name, the Interval is set to the Ending Date - Starting Date based on a 24 hour/day clock.

file

This is the current format control file variable. You will always pass `$file` to this parameter.

name

This is the input field name within the *\$file* variable where the calendar calculation routine will store the results of the calculation process (in other words, the interval of time between the starting and ending dates). You must pass the input field name exactly as shown.

Processing rules

The following rules are in effect when this routine is executed:

- If a start date is not passed to the routine, no calculations are performed.
- If an end date is not passed to the routine, no calculations are performed.
- If a duty table is not passed to the routine, a default calculation is performed. The start date is subtracted from the end date using a 24 hour/day clock.
- If the end date is less than the start date, no calculations are performed.
- If no **caldaily** records are found for the duty table that have a date greater than or equal to the start date, the calculation is based on a 24/hour clock.
- If the starting date and ending date are both less than the starting date of the first record found for the duty table, the calculation is based on a 24 hour/day clock. For instance, if the starting date was 12/15/91 and the ending date was 12/20/91, but the date of the first **caldaily** record for a particular duty table was 01/01/93, then the interval is '5 00:00:00'
- If the starting date is less than the starting date of the first **caldaily** record found for a duty table but the ending date is greater than the starting date of the first **caldaily** record, then the difference in time between the starting date and the start date of the first **caldaily** record is based on 24 hours/day. However, the remaining calculation process (until the ending time is reached) is based on the definitions in the **caldaily** records. For instance, assume the following: the starting date is 12/28/91 00:00; the ending date is 01/02/92 08:00; the **caldaily** records begin with 01/01/92 00:00 which is a holiday (no hours are worked); the start time in the **caldaily** record is 07:00:00. Based on this, the calculated interval is 4 days 1 hour (4 01:00:00) calculated as (01/01/92 00:00 - 12/28/91 00:00 (4 days)) + (08:00 - 07:00 (01:00)).
- If no **caldaily** records have the duty table name, the calculation is based on 24 hours/day.
- If the ending date is greater than the range of the **caldaily** records for a given duty table, then the calculations for the dates that exceed the **caldaily** records are based on 24 hours/day.

- If any skips are encountered in the **caldaily** records (for example, the current record has a date of '01/15/92 00:00' and the next record has a date of '01/25/92 00:00') the time between dates is counted as 24 hours/day.

Date interval calculation examples

The following table gives examples of the date interval calculation process.

	Start	End	Duty Table	Interval
1	12/15/91 14:00	12/16/91 10:00	day shift	20:00
2	12/31/91 22:00	01/04/92 08:30	day shift	18:00
3	01/02/92 11:30	01/02/92 16:00	day shift	04:00
4	01/02/92 11:30	01/02/92 11:45	day shift	00:00
5	01/03/92 15:00	01/15/92 10:30	day shift	2 12:30
6	01/04/92 22:00	01/05/92 07:30	day shift	00:00
7	01/04/92 22:00	01/06/92 10:30	day shift	02:30

- 1 Both the start and end dates occur before the date of the first **caldaily** record. The difference between the two dates is based on a 24 hour/day clock.
- 2 The start date occurs before the date of the first **caldaily** record. The difference between these two dates (2 hours) is counted as an interval. No hours are counted for 01/01/92 since it is a holiday. The dates 01/02/92 and 01/03/93 are both defined for 8 hours. The date of 01/04/92 is a Saturday with no defined working hours, therefore, no hours are added to the interval. A total of 18 hours is returned.
- 3 The start date of 01/02/92 occurs during the defined break times for that day. Therefore, the start time is adjusted forward to the break end date/time (01/02/92 12:00). Since the end date occurs on the same day the difference between the two dates is calculated as 04:00.
- 4 Both the start date and end dates occur during the defined break times for that day. Therefore, an interval of 00:00 hours is returned.

- 5 The difference between the start date and the end of the first day is 2 hours (17:00 - 15:00). The date of 01/03/92 is defined for 8 working hours which are added to the interval. The dates of 01/04/92 and 01/05/92 have no working hours, therefore, no hours are added to the interval. The next week ends on 01/12/92 which is less than the defined end date, therefore, the total hours for the week (40 hours) are added to the interval (40 + 10). The dates 01/13/92 and 01/14/92 are both defined for 8 working hours, therefore, another 16 hours are added to the interval (50 + 16). The last date of the date range adds another 02:30 to the interval resulting in a total of 68 hours and 30 minutes which equals 2 20:30 (2 days, 20 hours, 30 minutes).
- 6 The start date occurs on a Saturday which has no defined working hours. The next day, 01/05/92, also has no defined working hours. Therefore, a zero (00:00) interval is returned.
- 7 The start date occurs on a Saturday which has no defined working hours. The next day, 01/05/92, has no defined working hours. The end date occurs on 01/06/92 and the difference between the end date (10:30) and the shift start (08:00) is equal to 02:30.

Index

A

- actions 245, 246, 252
- adding
 - structures to the Database Dictionary 299
- alert date calculation 341, 367–372
 - examples 370–372
 - Format Control 369–370
 - processing rules 370
 - RAD subroutine 368
- Alt Appl
 - error exit 66
 - normal exit 66
- array of structure
 - creating subtables 285
- array of structures
 - creating subtables from 285
- arrays 295
 - arrayed structure maintenance utility 309
 - arrayed structures 303
- Assisted Entry Example 121

B

- backups 275
- balloon help 236, 249
- bank (text mode) 249
- basic.employee.info structure 302
- break duration 352
- break end 352
- break start 352
- Break Total 358
- building the calendar 345–347

C

- caldaily
 - database 350, 353
 - fields 351–353
 - form 348
 - form fields
 - break duration 352
 - break end 352
 - break start 352
 - daily duration 353
 - date 351
 - day of week 351
 - day of year 352
 - duty table name 351
 - end time 352
 - holiday name 352
 - start time 352
 - weekly total 353
- calduyhours
 - creating a new entry 357
 - database 356
 - form fields 355–356
 - Break Total 358
 - updating an existing entry 359
- calendar
 - building 345–347
 - objectives 343
 - preparing files 344
 - fields Also see caldaily 351
 - menu

- accessing 349
- daily hours 350–353
- holidays 360
- viewing 348
- calendar.calc.date.fc 369
- calholidays
 - database 360, 362
 - form fields 361
- capability words
 - help capabilities 201
- CenterPoint Web site 14
- character fields 294
- creating
 - holiday 362–363
 - objects 23
 - oncall schedule 159–160
 - states 35–37
- customer support 13

D

- daily duration 353
- daily hours 350–353
 - form 348
- data pools 281
- data types
 - character
 - changing to number 328
- database
 - caldaily 342, 350, 353
 - caldutyhours 342, 356
 - calholidays 342, 360, 362
- Database Dictionary
 - backing up a file 275
 - copying 273
 - data pools 281
 - deleting 278
 - field names
 - graph.of.unavailability-array 306
 - index pools 281
 - keys
 - displayevent file 251–253
 - displayoptions file 250
 - pools 281–282
 - renaming 277
 - resetting records 278
- date 351
- date interval
 - calculating 341
 - calculation 373
 - examples 376
 - Format Control
 - example 374
 - processing rules 375
 - RAD subroutine example 373
- date/time
 - fields 295
- day of week 351
- day of year 352
- deleting
 - holidays 365
- descriptor file (scdb.lfd) 293
 - adding fields 291
- display application
 - accessing records 239–254
 - actions 243, 245, 246, 248, 252
 - do nothing 243
 - redraw 243
 - return to application 243
 - balloon help 236, 249
 - calling an application 264–267
 - defining separate threads 249, 253
 - displayevent record 251–254
 - fields 251
 - RAD call 253
 - displaymaster record 240
 - fields 240
 - displayoption record 247–250
 - action, setting 270
 - comments 250
 - creating 225
 - creating the displayoption record 225–230
 - Database Dictionary keys 250
 - fields 248–250
 - RAD 249
 - RAD call 249
 - RAD expressions 250
 - testing the option 229
 - displayscreen record 241

- actions 243
- creating 254–255, 268–272
- creating for custom RAD application 269–270
- events structure 246
- fields 242
- initialization 243
- initialization expressions 243
- main structure 242
- options 244
- scripting 270–272
- window structure 244
- fdisp command panel 241, 243, 248
- fdisp command panels 252
- keys
 - displayevent file 251–253
- options
 - adding 223
 - screen IDs 224–225
 - sorting columns 224
- process flow 238
- restricting access 255–263
- rio command panel 241, 243, 248
- rio command panels 252
- screen ID
 - definition 241
 - screen vs. form 241
- do nothing 243
- Document Engine
 - accessing 21
 - processes 37
 - Standard Variables 42
- duty group 345
- duty hours form 354, 359
- duty table
 - name 351, 355

E

- education services 15
- end time 352
- Error Msg
 - Default Message 60

F

- fdisp command panel 241, 243, 248

fields

- adding 291
- arrays 295
- changing data types 328
- data types 294
 - arrays 295
 - character 294
 - date/time 295
 - expression 295
 - label 295
 - logical 295
 - number 294
 - record 295
 - structure 295
- Database Dictionary
 - descriptor field 291
- deleting 335
- modifying 324–331
- scalar 292
- searching for 283

files

- arrayed structure maintenance utility 309
- problem 308
- probsummary 308

Format Control

- alert date calculation 369–370
- configuring global lists 180–181
- subroutines 180

Format Control Call 88

forms

- attaching lists to 177–178

functions

- add field function 291

G

global initer

- introduction 171
- limitations 172
- server-side components 172

global lists 177

- attaching to forms 177–178
- building on startup 178–179
- configuring with Format Control 180–181
- introduction 171
- moving to a client 172

returning lists to client 178–181
 graph.of.unavailability 306

H

help

creating 206–208
 fields in record 193–196
 help file access
 command line 199–200
 Database Manager 199
 help menu 200–201
 toolbar button 200
 keywords 196, 208–212
 creating records 209–210
 querying help file 198
 related information 196, 208–212
 displaying 211–212
 types
 field
 file specific 202
 form specific 203
 field contents 203–204
 form 205
 term 205–206
 user views 197
 viewing 192–198

holiday

creating 362–363
 deleting 365
 updating 364–??, 364–366
 holiday names 352, 360
 holiday table 345, 355, 360, 361
 creating 366

I

Incident Management

problem file 308
 probsummary file 308

indexes

pools 281

initialization expressions 243

K

keys

adding 315

first key 315

keys to bottom of key list 321

deleting 337

inserting 319

modifying 331

types 316

keywords 196, 208–212

creating records 209–210

L

lister

list records 173–176

advanced fields 176

field definitions 175

troubleshooting

regenerating

all lists 186–189

obsolete lists 185

status and configuration 182–185

lists

binding to a form 177–178

building on startup 178–179

configuring with Format Control 180–181

global data 177

regenerating 185

all lists 186–189

obsolete 185

returning to client 178–181

variable 175

logical fields 295

Look-up Function 92

Look-Up Processing

RAD call 95

Look-up Processing 88

lookup processing 49

M

menus

calendar

daily hours 350–353

holidays 360

messages

notifications 154

msglog file 308

N

- no duplicates 317
- no nulls 317
- Notification Engine
 - Oncall Exceptions 167
- notification file 144
- notification record 167–170
 - adding new fields 150–151
 - editing 148–149
- nulls & duplicates 316
- number fields 294

O

- objects
 - about 22
 - field descriptions 23
- on call scheduling 157–162
 - creating a schedule 159–160
 - modifying a schedule 160–161
 - panel field definitions 158–159
- on-line help, See help
- Online Lookup 93
- options
 - show list 244, 246, 247

P

- Peregrine Systems
 - Corporate headquarters 14
 - customer support 13
 - Worldwide Contact Information 14
- pools
 - Database Dictionary 281–282
- Primary Field Definitions
 - allow Null 60
 - bypass cond 60
 - error msg 60
 - prompt panel 61
 - range summary 62
 - validate cond 59
 - values summary 62
 - Weight Factor 60
- process panel
 - field descriptions 39
- processes
 - Document Engine 37

- processing rules
 - Calendar Build caldaily 346
- Purge/Archive
 - accessing 308

R**RAD**

- applications
 - calling from displayevent record 253
 - calling from displayoption record 249
 - creating displayscreen records for 269–270
 - setting action in displayoption 270
- data types
 - label 295
 - record 295
- expressions 250
- subroutine 368
- RAD Call 95
 - array of fields to process 92
 - calculate risk only 92
 - error 93
 - file variable to process 92
 - format name 92
 - name of field to process 92
 - normal 93
 - online lookup 92
 - risk maximum 92
 - store calc results 92
 - suppress not found msg 92
- RAD Subroutines
 - validate.fields 90
- RAD subroutines
 - validity table processing 49
- Range Detail Definitions 70
 - min desc 73
 - range min 73
- Range Detail Specifications
 - max desc 74
 - range cond 75
 - range max 74
 - range stmts 75
 - relation 74
 - weight 74
- Range Summary

- minimum value description 62
- records
 - resetting 278, 280
- redraw 243
- Relation
 - and/or 74
- reserved words 292
- rio command panel 241, 243, 248
- Run Time Evaluation 134
- Run Time Validation Example 115

S

- screen IDs 253
 - identifying 224–225
- scripting
 - displayscreen record
 - creating 270–272
- Secondary Query Definitions
 - allow many 63
 - alt appl 66
 - condition 63
 - filename 63
 - lookup fields 65
 - names 66
 - no recs opt 65
 - QBE format 65
 - query type 63
 - select query 64
 - select sorts 64
 - val error msg 64
 - val fields 65
 - val query 64
 - val sorts 64
 - values 66
- ServiceCenter
 - reserved words 292
- sorting
 - columns 224
- Special Processing Example 127
- start time 352
- statements
 - validity table processing 49
- states 35
 - field descriptions 37
- structures 298

- adding arrayed structures 299
- arrayed structures 303
- basic employee.info 302
- fields within 300
- SysAdmin capability
 - Allow Null override 60

T

- technical support 13
- threading 250, 253
- training services 15

U

- unique 317
- updating
 - holidays 364–??
- updating a holiday 364–366

V

- Validate Fields
 - how to 88
- Validate.fields 92
- Validity Definitions
 - how to print 86
 - how to validate 82
- Validity Table Definitions
 - how to create 75
- Validity Table Primary Field Definitions
 - NULL default 60
- Validity Table Processing
 - examples 98
 - how to invoke 88
 - how to validate fields 88
- validity table processing
 - alphabetical processing sequence 49
 - case conversion 59
 - defined 49
 - sequence numbers 49, 57
 - validity file
 - accessing 50–55
 - field definitions 56–66
- Validity Validation Rules 82
- Value Detail Definitions
 - cond 69
 - desc 69

stmts 70
value 69
weight 70
variables
 \$vrfield 56, 139, 142
 display 175
 list 175

W

weekly total 353

