ServiceCenter®

SCAuto for Remedy ARS

Version 1.0

June, 1999

Peregrine Systems, Inc.
3611 Valley Centre Drive
San Diego, CA 92130

Peregrine SYSTEMS®
The Infrastructure Management Company™

TOC PREV NEXT INDEX

This edition applies to versions 3 and 2.X of the licensed program for **ServiceCenter**®,
version 4.0 of **Remedy ARS**, and version 1.0 of **SCAuto for Remedy ARS**

Contents

Preface

Overview

Welcome to Peregrine Systems' *SCAuto for Remedy ARS*, a bi-directional gateway between Peregrine's ServiceCenter™ and Remedy Corporation's ARS™. This guide describes how to implement the SCAuto for Remedy ARS interface for integration with Peregrine Systems' ServiceCenter.

The target audience for this guide is a consultant with experience in both ServiceCenter and Remedy ARS customization, as well as the other skills listed below. Implementation of this product without assistance from Peregrine's Professional Services organization or a qualified Peregrine partner company is not recommended unless the customer can supply all of the necessary prerequisite knowledge.

SCAuto for Remedy ARS is part of the suite of SCAutomate (SCAuto) interface products which integrate ServiceCenter with premier network and systems management tools.

Additional information about SCAutomate can be found in the *SCAutomate Applications for Windows NT and UNIX Guide*.

Prerequisite Knowledge

Implementation of the gateway requires some customization of both ServiceCenter and ARS, in order to allow each system to store a small amount of information about the other. For example, to integrate ServiceCenter Problem Management with Remedy Help Desk, each system must be customized to store the other system's ticket number within each problem ticket.

Although step-by-step instructions for performing this customization are included in this guide, it is recommended that implementation of this product be performed by an individual who is familiar with the operation of both products, or by two individuals working together, each of whom is familiar with one of the products.

This guide assumes that you have:

- Thorough knowledge of the operation of both ServiceCenter and Remedy ARS, and of the operating system platforms on which those products are installed. This may include UNIX, and/or OS/390 in the case of ServiceCenter.
- Prior experience customizing ServiceCenter, or equivalent training.
- Prior experience customizing Remedy ARS, or equivalent training.
- Experience in installation and operation of products on Windows NT.
- Knowledge of Microsoft JScript, Netscape JavaScript or ECMAscript.

Changes to the JScript code delivered with this gateway **are required**, **unless** the gateway is implemented between **new**, **unmodified** installations of both ServiceCenter and Remedy ARS.

The reason for this is that both ServiceCenter and Remedy are usually customized to meet the needs of each site, and it is not possible for the gateway to automatically adjust itself to whatever customization may have been performed over the years. Therefore you will most likely have to adjust or extend the mapping by modifying the scripting code delivered with this gateway, unless it is implemented between an unmodified copy of ServiceCenter and ARS.

TOC PREV NEXT INDEX

Contacting Peregrine Systems

Contact one of the Peregrine Systems Customer Support offices listed here if you have questions about or problems with ServiceCenter systems.

North and South America

To get help immediately, call Peregrine Customer Support at:

(800) 960-9998 or **(619) 794-7402**

For ServiceCenter questions or information, send a fax or e-mail to:

Fax: **(619) 794-6028**
E-mail: **support@peregrine.com**
Hours: 5:00 A.M. to 5:30 P.M. PST, Monday through Friday

Send materials that Peregrine Systems Customer Support requests to:

Peregrine Systems, Inc.
ATTN: Customer Support
3611 Valley Centre Drive
San Diego, CA 92130

Europe

Great Britain	**Peregrine Systems Ltd.** **1st Floor** **Ambassador House** **Paradise Road** **Richmond, Surrey, Great Britain, TW9 1SQ** Phone: **+44 (0) 181-332-9666** Fax: **+ 44 (0) 181-334 5890** E-mail: **uksupport@peregrine.com** Hours: 08:00h to 18:00h GMT, Monday through Friday
France	**Peregrine Systems** **Tour Franklin-La Défense 8** **92042 Paris La Défense Cedex, France** Phone (International Toll Free): **+33 (0)800 505 100** Fax: **+33 (0)1 47 73 11 61** E-mail: **frsupport@peregrine.com** Hours: 08:00h to 18:00h, Monday through Friday
Germany	**Peregrine Systems GmbH** **Burohaus Atricom** **Lyoner Strasse 15,**

60528 Frankfurt, Germany

Phone: **+49-(0)69-66-80-260**
Fax: **+49-(0)69-66-80-2626**
Hours: 08:00h to 17:00h, Monday through Friday

Denmark

Peregrine Systems A/S
Naverland 2, 12 SAL
DK-2600 Glostrup
Denmark

Phone: **+45-4-346-7676**
Fax: **+45-4-346-7677**
Hours: 08:30h to 17:00h, Monday through Friday

Holland/
Netherlands/
Benelux

Peregrine Systems BV
Botnische Golf 9a
3446 CN Woerden
the Netherlands

Phone: **+31-348-437070**
Fax: **+31-348-437080**
Hours: 08:30h to 17:30h, Monday through Friday

Asia-Pacific

E-mail: **apsupport@peregrine.com**
Phone (U.S.): **619-794-7402**
Phone (Australia): **800-146-849**
Phone (Hong Kong): **800-90-8056**
Phone (Singapore): **800-1300-949-948**
Phone (Japan): **0044-221-22795**

Note: Only the Peregrine Systems European Customer Support staff is multilingual and can provide technical support to customers in their native language.

Chapter 1 Introduction

Overview

SCAuto for Remedy ARS is a bi-directional gateway between ServiceCenter and Remedy's Action Request System (ARS), providing continuous, automatic exchange of information between the two systems on a 24x7 basis. Running as a Windows NT service, it uses state-of-the-art Component Object Model (COM) objects and ActiveX scripting to provide maximum flexibility and extensibility.

Both ServiceCenter and Remedy ARS have a C/C++ application programming interface (API). But because both products are highly customizable, no two implementations of ServiceCenter or Remedy ARS are necessarily the same. Therefore, to create a generalized gateway that can be easily customized to integrate the two products is a challenging task. Fortunately, recent advances in technology have provided several tools that facilitate this integration:

- Object models such as COM, which permit traditional APIs to be encapsulated into easy to use objects
- OLE Automation, which defines dynamic COM interfaces to permit scripting languages to use COM objects
- Scripting languages such as VBScript and ECMAScript (standardized JScript), which provide rapid application development

SCAuto for Remedy ARS leverages these technologies to provide a robust gateway which takes full advantage of the power of the ServiceCenter and Remedy C/C++ APIs, without introducing the complexity and inflexibility normally associated with C/C++ applications.

Theory of Operation

SCAuto for Remedy ARS is designed to exchange information between the two service desk products. It works by alternatively polling each service desk product for information at a user-defined interval and then converting the information received into a transaction for the other service desk. The conversion is performed by simple assignment of property values between a pair of OLE Automation objects in JScript code.

Unlike other schemes for interoperability between service desks, SCAuto for Remedy ARS does not impose a predefined generalized data model and then require each service desk product to implement that model. In the author's experience, the latter design tends to suffer from the following problems:

- It is difficult to get the various service desk products to support the generalized data model.
- When service desk products do support the model, they tend to support a relatively modest subset of the model.
- As a result, the nature of data communication between the service desk products may be of limited bandwidth.

By contrast, SCAuto for Remedy ARS uses a more flexible approach, whereby data representing a transaction is extracted from a service desk product by an object which knows how to communicate with that service desk. The data in the object may then be manipulated as needed using scripting language, and assigned directly to the properties of another object which knows how to communicate with the other service desk product. In this way, a powerful, extensible gateway between two service desks can be implemented in a relatively short and simple piece of scripting code.

SCAuto for Remedy ARS customizes ServiceCenter to produce outbound event messages whenever a problem ticket is opened, updated or closed. Each event message is then retrieved by the gateway and converted into an OLE Automation object known as an *SCEvent*. Each SCEvent object has a set of properties, including an EvType (event type) and a collection of EvField (event field) objects. For problem tickets, the event type property indicates whether a problem open, update or close has occurred, and each EvField object contains the data for a particular field of the problem ticket. The SCEvent object is then assigned to an *REvent* object, which is a Peregrine-provided OLE Automation object which knows how to communicate with ARS using the Remedy API. An analogous operation is performed in the other direction to communicate ARS information to ServiceCenter.

Delivered Functionality

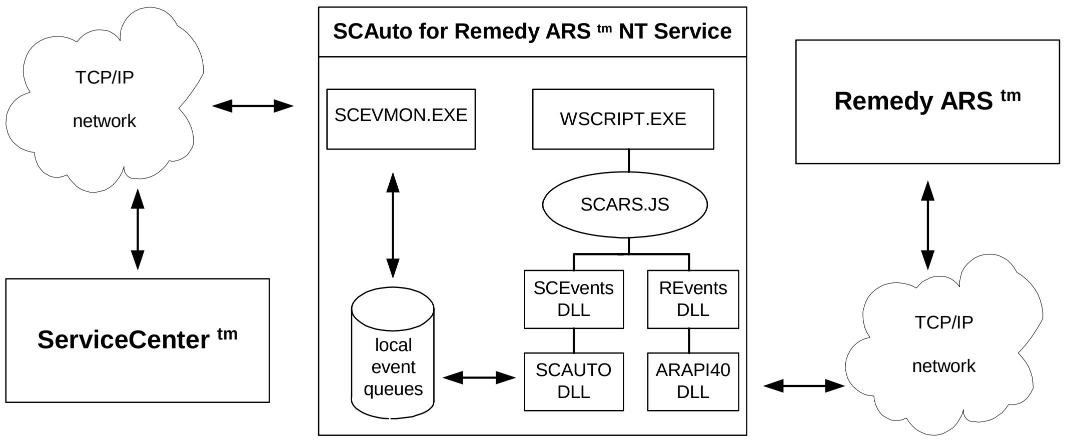
As delivered, the gateway performs a bi-directional exchange of trouble-ticket information, converting ServiceCenter Problem Management documents into Remedy ARS trouble-tickets, and vice-versa. Each new trouble-ticket created in ARS becomes a new problem document in ServiceCenter, and each new problem document created in ServiceCenter Problem Management becomes a new Remedy trouble-ticket. Likewise, problem updates and closes are communicated back and forth.

Extensibility

Although SCAuto for Remedy ARS is delivered with a script that demonstrates integration of problem information, the gateway is not limited to exchange of problem information. Any ServiceCenter application can be integrated with any Remedy application which can be manipulated via the ARS API. Each application to be integrated requires development of another script similar to the one provided. The gateway can run any number of additional scripts, simply by extending the scars.cfg file as discussed in Chapter 3.

Architecture

The diagram below illustrates the architecture of SCAuto for ARS.



Note the following:

- There are three major components—ServiceCenter, Remedy ARS and SCAuto for Remedy ARS (the gateway between the two).
- Each component can be installed on the same or a different server.
- Communication between the SCAuto for Remedy ARS gateway and ServiceCenter is based on the SCAuto API, which uses TCP (see chapter 5).
- Communication between the SCAuto for Remedy ARS gateway and Remedy ARS is based on the Remedy API, which uses RPC over TCP.
- ServiceCenter can be running on any supported OS platform (Windows NT, UNIX or MVS.
- Remedy ARS can be running on any supported OS platform (Windows NT or UNIX).
- The SCAuto for Remedy ARS gateway requires a Windows NT server platform.

Product Components

SCAuto for Remedy ARS provides a set of OLE Automation objects which provide a simple way to access the full functionality of ServiceCenter and Remedy ARS from VB, VBScript, or JScript (ECMAscript) programming, eliminating the need for laborious C or C++ programming. These objects are then exploited by a simple JScript routine (source code provided) which implements a mapping between the default ServiceCenter Problem Management and Remedy ARS Help Desk schemas.

SCAuto for Remedy ARS has a modular architecture which includes the following building blocks:

- **SCEVENTS.DLL -** an OLE Automation object which encapsulates SCAUTO.DLL, the ServiceCenter API, which uses TCP sockets to communicate with ServiceCenter.
- **REVENTS.DLL** - an OLE Automation object which encapsulates the Remedy API (ARAPI40.DLL and related DLLs). These use RPC to communicate with ARS.
- **SCRUTIL.DLL** - an OLE Automation object which provides utility functions for logging messages, obtaining run-time parameters, etc.
- **SCARS.JS -** a short ActiveX script written in JScript which implements the actual gateway, by continuously performing these steps:
  - Extract any waiting Problem Management events from ServiceCenter into a new SCEvent object
  - Map the SCEvent object data into a new REvent object
  - Create an ARS trouble ticket or ticket update using the REvent object
  - Extract any waiting trouble ticket or trouble ticket update from ARS using the REvent object
  - Map the REvent object data into a new SCEvent object
  - Create ServiceCenter Problem Management events from the SCEvent object
- **WSCRIPT.EXE** - The Microsoft Windows Scripting Host.
- **JSCRIPT.DLL** - Microsoft 5.0 ActiveX scripting engine.
- **SCEVMON.EXE -** an daemon which manages communication of ServiceCenter events between SCAuto for Remedy ARS and ServiceCenter. It provides event message queuing so that ServiceCenter can be started and stopped without affecting the operation of the gateway.
- **SCARSSRV.EXE** - A Windows NT service to run the above described components continuously in the background.

Customization

Customization of the SCAuto for Remedy ARS gateway is accomplished by modifying the SCARS.JS script to conform to your customized versions of ServiceCenter and Remedy ARS. Instructions for doing this are included in this guide. As delivered, SCARS.JS implements a mapping between the "out-of-the-box" ServiceCenter Problem Management application and the default Remedy ARS Help Desk Schema. The mapping is accomplished by series of simple assignments of ServiceCenter problem fields to Remedy ARS problem fields and vice-versa, using the SCEvent and REvent objects. The provided JScript routine is easily customized to conform to customized schemas simply by changing the names of the data fields to conform to your customized Remedy or ServiceCenter schema, or by adding new fields.

Extensibility

The SCAuto for Remedy ARS product is not limited to trouble ticketing. The SCEvent object supports creation of any kind of ServiceCenter event, which means it can communicate with all of the ServiceCenter applications. And the REvent object supports any Remedy ARS schema, thereby supporting Remedy applications other than trouble ticketing.

Chapter 2 Installation

Overview

To operate SCAuto for Remedy ARS, the following products must be installed:

- ServiceCenter
- Remedy ARS
- SCAuto for Remedy ARS
- Windows Script 5.0

This document provides installation instructions for only the last two items (SCAuto for Remedy ARS and the Windows Script 5.0). It is assumed the first two items (ServiceCenter and Remedy ARS) are already installed on your system. If not, be sure to install them before proceeding with the instructions provided in this document.

Installation Checklist

Verify the following items before proceeding:

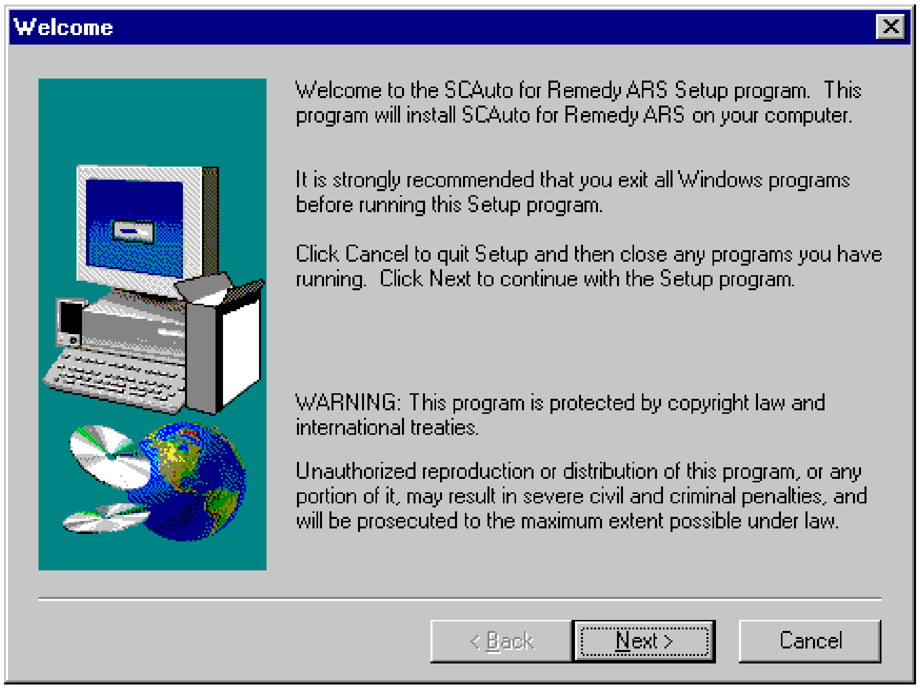
- ServiceCenter 2.1 SP3E2 or later is installed on your system.
- Remedy ARS 4.0 or later is installed on your system, or your earlier ARS system has been upgraded to 4.0.
- Remedy API materials from the Remedy 4.0 Action Request System CD have been installed on your system.
- ServiceCenter is properly configured to run Event Services, including the SCAUTOD server.
- You have collected the information needed to configure the gateway:
  - ServiceCenter hostname
  - TCP port number of the ServiceCenter SCAUTOD server
  - Remedy ARS server hostname
  - Remedy ARS userid and password to be used by the gateway

SCAuto for Remedy ARS

To install SCAuto for Remedy ARS:

1. Insert the SCAuto for Remedy ARS CD and run *setup.exe*.

   The Setup program runs until the Welcome screen is displayed.



Welcome to the SCAuto for Remedy ARS Setup program. This program will install SCAuto for Remedy ARS on your computer.

It is strongly recommended that you exit all Windows programs before running this Setup program.

Click Cancel to quit Setup and then close any programs you have running. Click Next to continue with the Setup program.

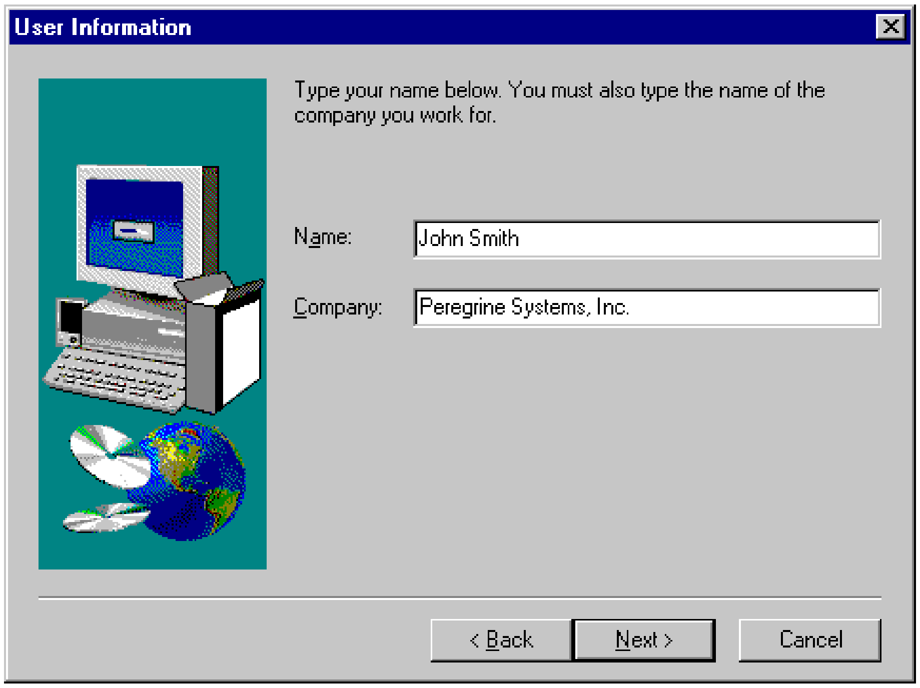
WARNING: This program is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

Figure 2-1 Welcome

2. Click on **Next** to continue.

The User Information screen is displayed.



Type your name below. You must also type the name of the company you work for.

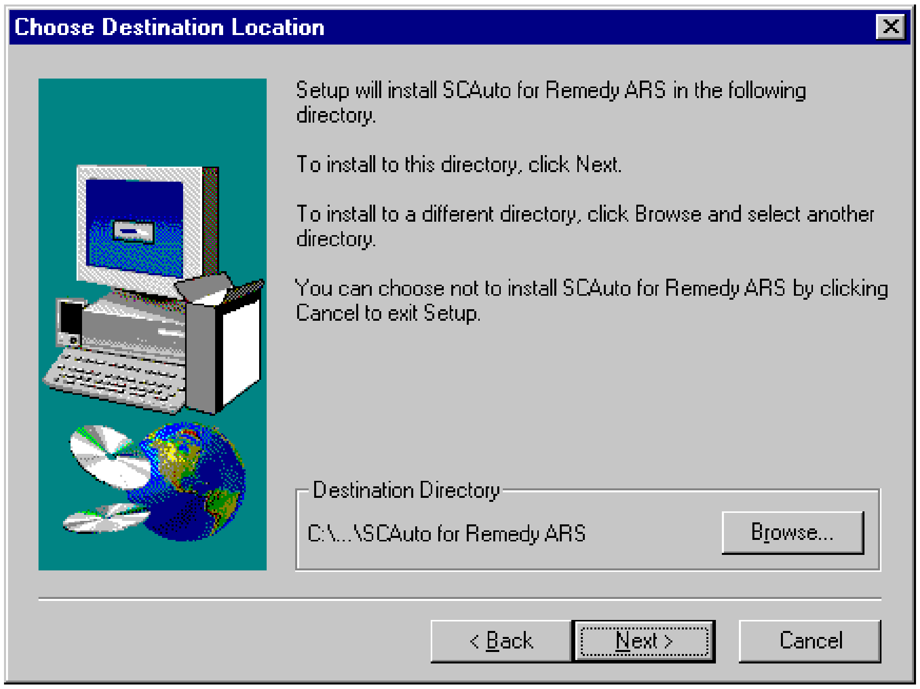
Name: John Smith

Company: Peregrine Systems, Inc.

Figure 2-2 User Information

3. Type your name and your company name, then click on **Next**.

The Choose Destination Location screen is displayed, which shows the directory to which SCAuto for Remedy ARS will be installed.



Setup will install SCAuto for Remedy ARS in the following directory.

To install to this directory, click Next.

To install to a different directory, click Browse and select another directory.

You can choose not to install SCAuto for Remedy ARS by clicking Cancel to exit Setup.

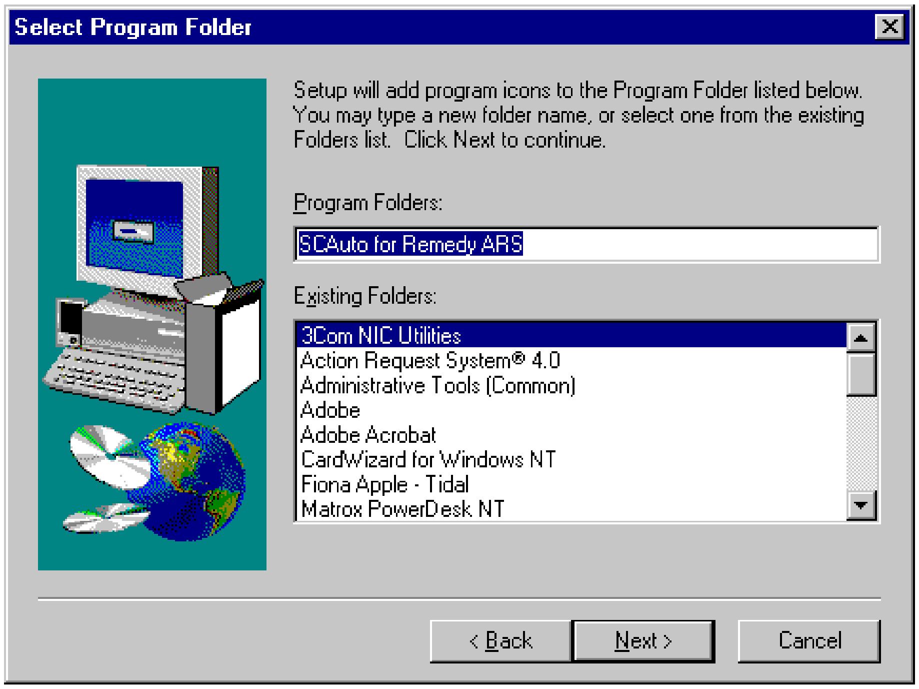
Destination Directory

C:\...\SCAuto for Remedy ARS

Figure 2-3 Destination Location

4. Ensure the Destination Directory is correct.
   - If so, click on **Next**.
   - If not, click on **Browse** and select a different directory. Then click on **Next**.

The Select Program Folder screen is displayed, where you select the program folder to which you want program icons added.



Select Program Folder

Setup will add program icons to the Program Folder listed below. You may type a new folder name, or select one from the existing Folders list. Click Next to continue.

Program Folders:

SCAuto for Remedy ARS

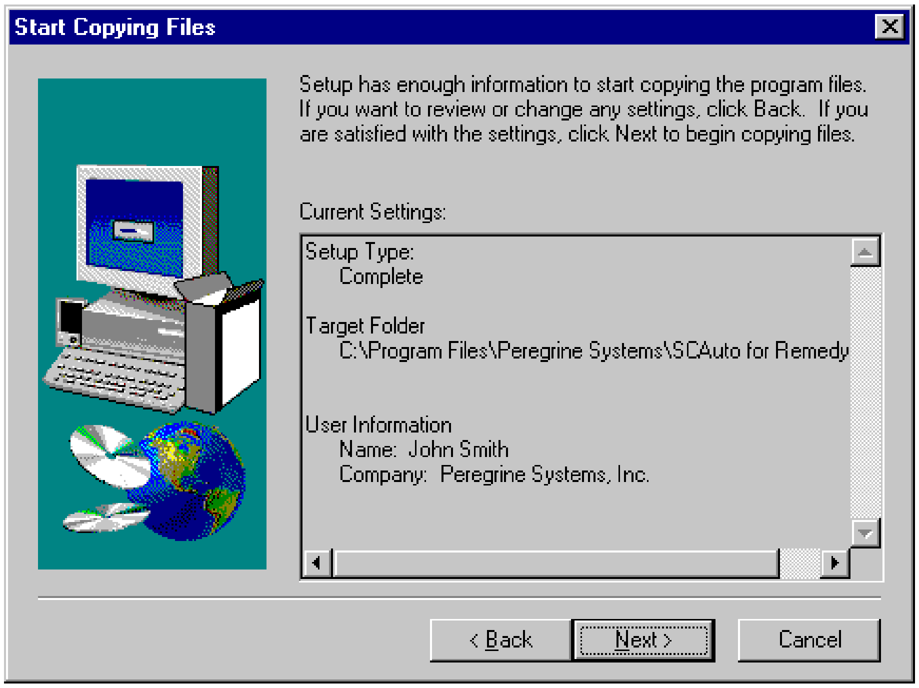
Existing Folders:

3Com NIC Utilities
Action Request System® 4.0
Administrative Tools (Common)
Adobe
Adobe Acrobat
CardWizard for Windows NT
Fiona Apple - Tidal
Matrox PowerDesk NT

Figure 2-4 Select Program Folder

5. Select a program folder for your SCAuto for Remedy ARS system, then click on **Next**.

The Start Copying Files screen is displayed, indicating that Setup is ready to start copying the SCAuto for Remedy ARS program files.


Start Copying Files

Setup has enough information to start copying the program files. If you want to review or change any settings, click Back. If you are satisfied with the settings, click Next to begin copying files.

Current Settings:

Setup Type:
    Complete

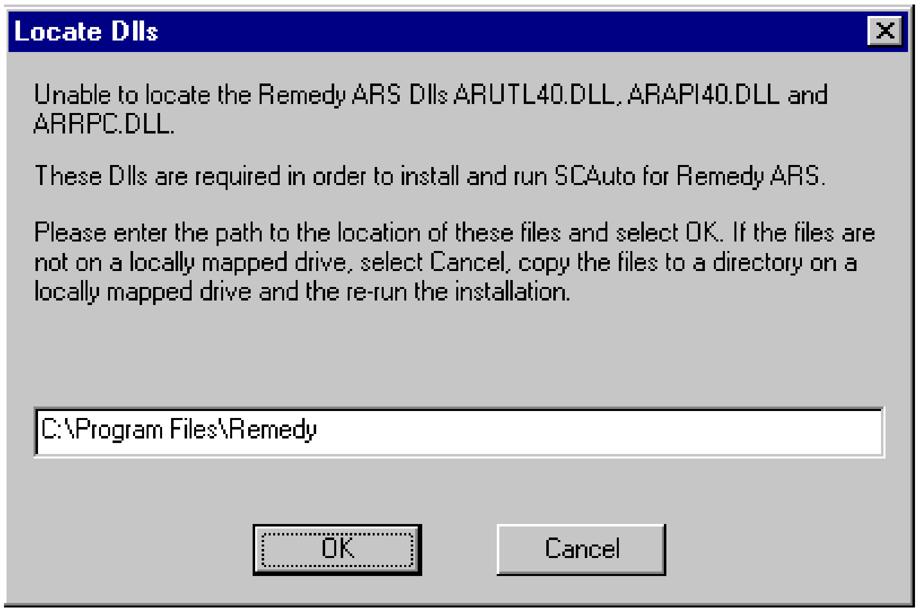
Target Folder
    C:\Program Files\Peregrine Systems\SCAuto for Remedy

User Information
    Name: John Smith
    Company: Peregrine Systems, Inc.

Figure 2-5 Start Copying Files

6. After verifying that the current settings displayed are correct, click **Next** to start copying the files.

If you are installing SCAuto for Remedy ARS onto a system that does not have Remedy ARS installed, Setup will display the Locate Dlls dialog and ask you for the location of the Remedy ARS Dlls ARUTL40.DLL, ARAPI40.DLL and ARRPC40.DLL. These Dlls are required in order for SCAuto for Remedy ARS to be able to communicate with Remedy ARS.



Unable to locate the Remedy ARS Dlls ARUTL40.DLL, ARAPI40.DLL and ARRPC.DLL.

These Dlls are required in order to install and run SCAuto for Remedy ARS.

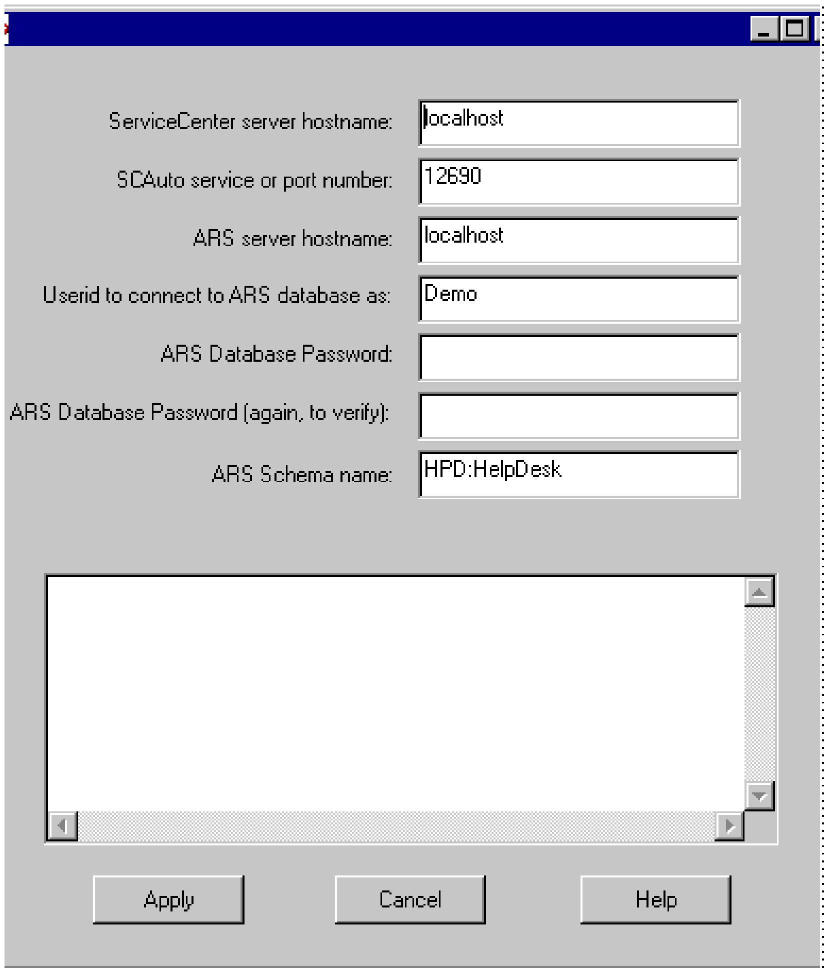
Please enter the path to the location of these files and select OK. If the files are not on a locally mapped drive, select Cancel, copy the files to a directory on a locally mapped drive and the re-run the installation.

C:\Program Files\Remedy

Figure 2-6 Locate Dlls

7. Enter the path to the Remedy Dlls and click on **OK**, or
8. If the Dlls are not available on this system or on a drive mapped to this system, click on **Cancel**. Then copy the files to a location accessible to the local system and re-run the SCAuto for Remedy ARS installation.

After all files are installed on your system, a dialog is displayed allowing you to edit your scars.ini file settings. Refer to the provided help or the section on the scars.ini file in Chapter 3 of this guide for further information.



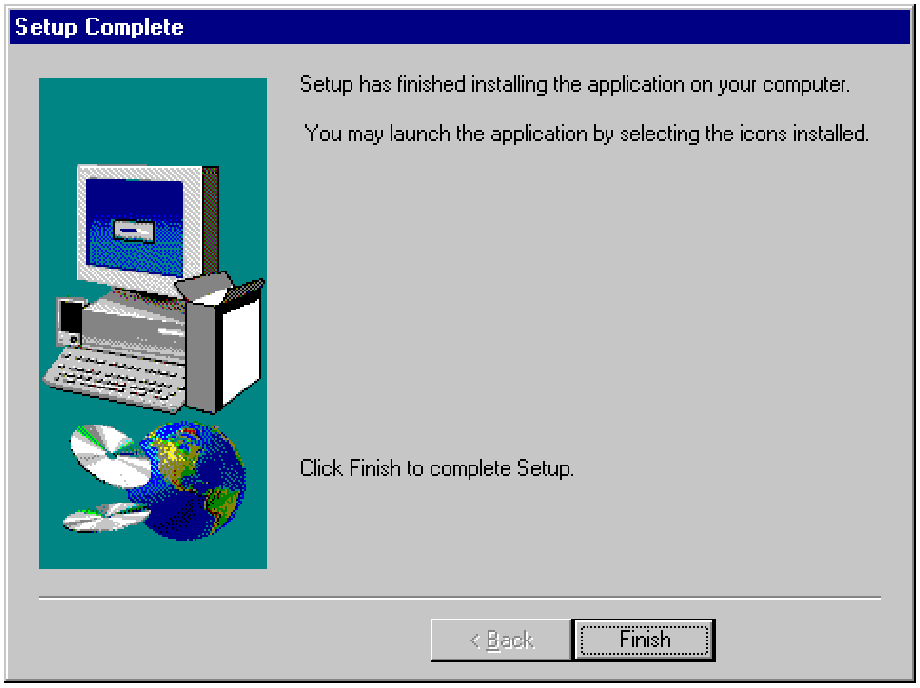
ServiceCenter server hostname:	localhost
SCAuto service or port number:	12690
ARS server hostname:	localhost
Userid to connect to ARS database as:	Demo
ARS Database Password:	
ARS Database Password (again, to verify):	
ARS Schema name:	HPD:HelpDesk

Figure 2-7 Configuration Screen

9. Ensure all settings are correct, then click on **Next**.

Note: These settings can be changed later by selecting **Programs>SCAuto for Remedy ARS>Configure SCAuto for Remedy ARS** from your Windows Start menu.

The Setup Complete screen is displayed when Setup has finished installing SCAuto for Remedy ARS on your computer.


Setup Complete

Setup has finished installing the application on your computer.

You may launch the application by selecting the icons installed.

Click Finish to complete Setup.

Figure 2-8 Setup Complete

10. Click on **Finish** to exit.

    SCAuto for Remedy ARS is now installed.

11. After finishing the installation and configuration of all components, please refer to Chapter 7 of this guide for further information on operating your SCAuto for Remedy ARS system.

Note: If you already have your own custom script, you will want to copy this into your installation\bin directory.

Windows Script 5.0

SCAuto for Remedy ARS requires Windows Script 5.0, which includes Windows Scripting Host and the 5.0 scripting engines, including JScript 5.0.

To install Windows Script 5.0:

1. Access the web site http://www.microsoft.com/msdownload/vbscript/scripting.asp
2. Despite the *vbscript* in the URL, this takes you to the Windows Script 5.0 page, which invites you to download **ste50en.exe** (900K) which is the US English version, or an equivalent localized version, such as **ste50de.exe**, or **ste50fr.exe**. The download includes Windows Script 1.0, JScript 5.0, VBScript 5.0 and other required files.
3. Download the appropriate Windows Script 5.0 file and install it. You must be running version 5.0 or later of the Windows Scripting engines to successfully run the Microsoft JScript code provided with SCAuto for Remedy ARS.
4. While you are here, you may wish to download the free script debugger, unless you have Visual Studio Professional or Enterprise Edition, which includes an enhanced debugger.
5. We also recommend you download the Microsoft documentation package for JScript 5.0

Chapter 3 Configuration

Overview

To configure SCAuto for Remedy ARS, you may need to make changes to the following files:

scars.ini

Changes to this file are rarely needed once a few key parameters are correctly set. These include the identities of the ServiceCenter and Remedy servers, and the userid and password required for connection to the Remedy database. The latter parameters should have been correctly set when you ran the configuration dialog at the end of the installation process. If you need to change those, it is recommended you rerun the configuration dialog rather than hand-edit the scars.ini file. However, you may need to modify the scars.ini file by hand for certain things, such as activating debug logging. A discussion of the required scars.ini file parameters follows. A complete list of .ini parameters is also given in Appendix A.

scars.cfg

Changes to the this file should not be necessary. An exception might be the specification of some transient NT command you wish executed at service start-up, or specification of additional scripts to be run as part of the gateway.

scars.ini File

The following parameters in the scars.ini file may need to be configured:

sceventserver:localhost.12690

The *sceventserver* parameter defines the TCP/IP hostname and TCP port number of the ServiceCenter SCAUTOD server. The two values must be specified together, separated by a period. The SCAUTOD TCP port number must not be confused with the ***other two TCP port numbers*** used by ServiceCenter (express client server and full client server). The value *localhost* in this example indicates that the ServiceCenter server is running on the same machine as the gateway. The *12690* value is the default TCP port used by the SCAUTOD server in ServiceCenter.

scauto_polling_delay:30000

The *scauto_polling_delay* parameter specifies the interval in milliseconds at which the scars.js script will poll the ServiceCenter and Remedy COM objects for new transactions to be processed.

ars_server:

The *ars_server* parameter defines the TCP/IP hostname of the Remedy ARS server.

ars_username:

The *ars_username* parameter defines the user id for SCAuto for Remedy ARS to use when logging in to Remedy ARS.

ars_password:

The *ars_password* parameter defines the password for SCAuto for Remedy ARS to use when logging in to Remedy ARS. This value is stored in encrypted form and should not be edited by hand.

ars_schema:

The *ars_schema* parameter defines the ARS schema to be used for updating the Remedy ARS database. This is a required parameter. All interaction with the Remedy ARS database is performed in terms of a particular schema. The default schema delivered with the SCAuto for Remedy ARS gateway is for the standard Remedy Help Desk application without customization applied.

Important: The following parameters in the scars.ini file do not need to be modified. These should only be changed at the direction of Peregrine support personnel.

log:scars.log

Indicates that the name of the message log file is to be "scars.log".

scevmon_sleep_interval:

Specifies the delay in seconds between polls of ServiceCenter.

scevents:(pmo,pmu,pmc):

Lists the event types which will be retrieved from or sent to ServiceCenter. Parentheses and commas must be specified as shown.

debugscautoevents:0

If 1, additional messages are logged in the indicated log file.

event_map_dir:EventMap

Specifies the directory (relative to installation directory) where event maps are kept.

killableprocesses:1

If 1, indicates that the SCARSSRV service should permit the NT Task Manager application to terminate its processes.

stopeventname:SCAutoARS.StopEvent

Name of Win32 event which is signalled at service shutdown.

sessid:SC_ARS

Unique identifier given to this instance of SCAuto for Remedy ARS.

scars.cfg File

The scars.cfg file is read by the SCAuto for Remedy ARS service manager program (SCARSSRV.EXE) at startup, and controls the commands and processes executed by the service. Each line in the file must be a valid Windows NT command.

There are two types of commands that can be specified:

- Transient NT commands, such as might be issued from an NT command line. These must be prefixed with *CMD /c*. Once execution of the command finishes, it terminates. These can be used to perform maintenance tasks (such as moving or renaming a log file) each time the service is started.
- Long-running commands, which execute until the service is shut down. The long running commands relevant to SCAuto for Remedy ARS are discussed below.

Transient NT Commands

As distributed, no transient NT commands are specified in the scars.cfg file.

Long-running Commands

The following long-running commands are relevant to SCAuto for Remedy ARS:

SCEVMON.EXE

scevmon.exe is a background process which is responsible for communication with ServiceCenter. It runs until the service is shut down. It makes a TCP connection with the SCAUTOD server component of ServiceCenter, using the hostname and port information supplied via the **sceventserver** parameter of the **scars.ini** file. There should be no need to modify this line of scars.cfg. If you do, SCAuto for Remedy ARS will probably not operate correctly.

WSCRIPT.EXE

This command invokes the Microsoft scripting host to execute the **scars.js** script. The JScript code in **scars.js** implements the actual logic of the gateway. Most of the customization changes you make to the gateway will be changes to scars.js, rather than to scars.cfg or scars.ini.

Chapter 4 Customizing ServiceCenter

Overview

In order for SCAuto for Remedy ARS to work properly, some customization must be performed to ServiceCenter. This customization adjusts what ServiceCenter normally does in response to inbound problem event messages and alters the way outbound problem event responses are generated in reply. It also ensures that problem events are sent out when problem tickets are manually opened, updated or closed by ServiceCenter operators.

Experienced ServiceCenter consultants and administrators will be familiar with the steps described in this chapter. Some might decide an alternative customization approach is more appropriate, perhaps because of substantial customization already performed to the ServiceCenter system at their site. If so, please first implement the customization described below on a freshly installed, unmodified ServiceCenter system, both to verify the operation of the gateway and to ensure you fully understand the objectives of the customization (described below) **before** designing any alternative approach. Keep in mind it is more difficult for Peregrine to provide assistance if you radically depart from the methods described here.

Customization Objectives

The changes to ServiceCenter described in the following sections are intended to accomplish the bulleted list of requirements given below. Any alternative customization approach that is adopted must satisfy each of these requirements, unless you are removing a major function from the gateway, for example, making the gateway function in a uni-directional manner rather than a bi-directional manner. Another example would be customizing the gateway to work with a different ServiceCenter application, and therefore a different set of events. Such significant changes should probably only be done by an experienced consultant.

Also note that the discussion below assumes you are familiar with ServiceCenter facilities such as Event Services and Format Control. If you are not familiar with these facilities, implementation of this product is probably not a good time to approach them for the first time. Seek assistance from a qualified consultant or obtain training from Peregrine Systems.

- Ensure that for each and every *pmo* (Problem Open) event sent to ServiceCenter by SCAuto for Remedy ARS, a new problem ticket is opened, and no inbound *pmo* events are ever transformed into a *pmu*. The usual behavior of the *pmo* event is to first try to find a matching problem ticket and update that ticket. That logic is not appropriate for this gateway. Each *pmo* from the SCAuto ARS gateway must open a new ticket. However, a unilateral change to the *pmo* event registration expression to always open a new problem for every *pmo* is probably not appropriate, because other SCAuto applications may depend on the other behavior. For example, if a network management platform sends repeated events about the same symptom on the same node, you usually want these to update the same ticket, not open thousands of new tickets. Problem events from the SCAuto ARS gateway can be distinguished by the *evuser* field of the event message. For SCAuto ARS, the evuser value is *SCAutoARS*. This value is assigned to the event when it is created in scars.js. The customization steps described in this chapter update the event registration expressions for *pmo* events to check to see if SCAuto ARS created the event, and if so, always open a new ticket.
- Ensure that *pmo* (Problem Open), *pmu* (Problem Update), and *pmc* (Problem Close) events sent from the gateway include a new field that carries the Remedy ARS problem ticket number in them, and that this ARS ticket number is stored in the *problem* and *probsummary* table when the event is processed.
- Ensure that when a *pmu* or *pmc* event arrives from the gateway, it can update or close the correct ServiceCenter problem ticket, even if it does NOT contain the ServiceCenter problem number. This requirement arises because a Remedy ticket might be created and then immediately updated again before the corresponding problem has been opened in ServiceCenter. Therefore a *pmu* may need to be sent to ServiceCenter before the ServiceCenter problem number is known. The customization shown later in this chapter ensures that ServiceCenter problems can be located by Remedy ticket number. The event registration expressions for *pmu* and *pmc* are customized in a way that allows the ServiceCenter problem to be updated or closed to be found using the Remedy ticket number contained in the *pmu* or *pmc*.
- Ensure that when a human operator or background ServiceCenter process creates, updates or closes a problem ticket, a corresponding *pmo, pmu* or *pmc* event is written to the eventout file if you wish that action to be propagated to Remedy ARS. The customization described in this chapter

shows how this can be done using Format Control, on a per-category basis. A different but perfectly valid approach might use the ServiceCenter assignment group concept instead, causing events to be written to eventout whenever a ticket was assigned to a particular assignment group. The key is to ensure an appropriate event is written to eventout when you wish the gateway to propagate information to Remedy ARS.

- Ensure that no duplicate events are written to eventout. Some customization approaches may cause more than one *pmo*, *pmu*, or *pmc* event to be written to eventout. This is not acceptable. Duplicate output events will confuse the gateway and may cause it to fail. For example, if both Format Control and Event Registration both execute logic which writes a *pmo*, *pmu*, or *pmc* event to the eventout file, you will get duplicate outbound events. To avoid this problem, the customization approach shown later in this chapter suppresses generation of response events to eventout when processing inbound events from SCAutoARS, preferring to have Format Control expressions cause the output event to be written. Other approaches are possible.
- Finally, the customization must implement some reasonable policy for sharing of problem tickets between ServiceCenter and Remedy ARS. By "policy", we mean answers to questions like these:
  - Which tickets should be shared between the two systems? All tickets? Only certain categories of problems? Only certain assignment groups?
  - Should tickets be allowed to be updated or closed in either system? Or should control and ownership of a ticket pass back and forth between the systems?

The customization described in this chapter implements total bi-directional ticket sharing. Any ticket opened, updated or closed in either system causes the same action to be performed in the other system. The customization is shown for just one problem category, the *default* category. Assignment group is not used to control whether a ticket goes to Remedy or not. And tickets can be freely updated in either system. No mechanism is implemented to "lock" a ticket after it has been sent to the other system. These choices do not represent a value judgement as to the "right" way to do things. Each installation contemplating the use of this gateway in production should decide for itself what policies to use. A requirements analysis process should be carried out to decide these questions before applying any customization to a production system.

Format Control

For every problem open (pmo), problem update (pmu) and problem close (pmc) created in ServiceCenter, you must ensure an outbound message is generated in the eventout message queue.

For each problem category you want to be sent to the event out queue, perform the steps that follow. For demonstration purposes, we will illustrate how to change the *default* problem category to do this. In a production installation, you may wish to customize all existing problem categories or set up a new, special problem category to be used for propagation of problem data to ARS.

Note: Version 2.1 of ServiceCenter is shown. If you are running a different version, such as 1.4 or 3, and are unfamiliar with the process illustrated here, please consult your ServiceCenter documentation.

1. Select the Utilities tab from ServiceCenter's main screen.

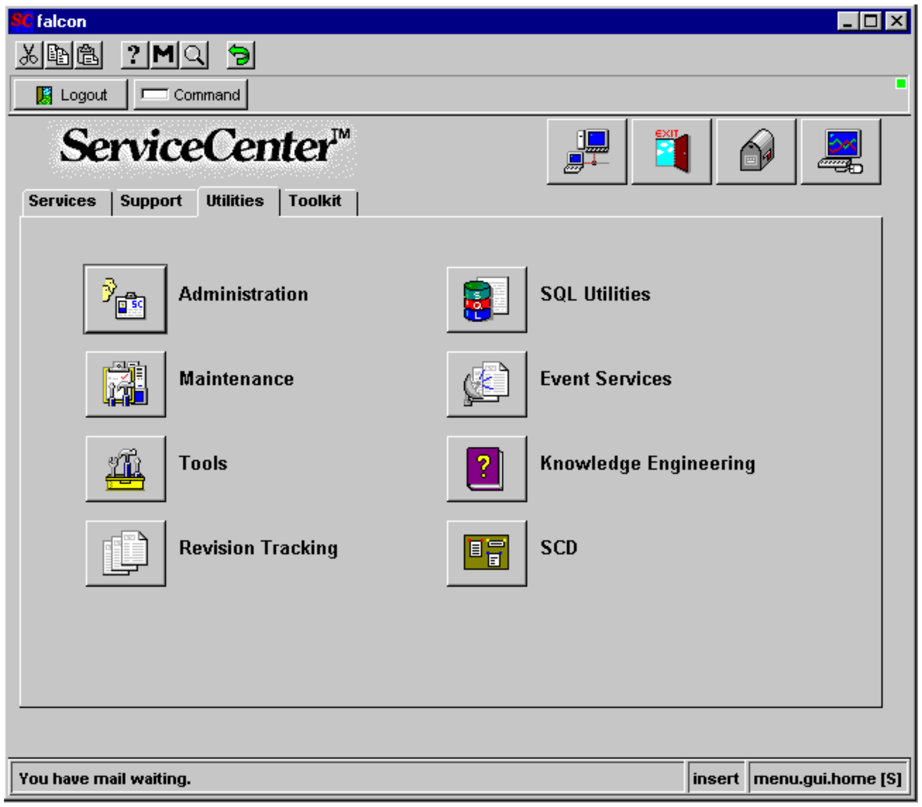


Figure 4-1 Utilities Tab

2. Click on Tools.

The Tools menu is displayed.

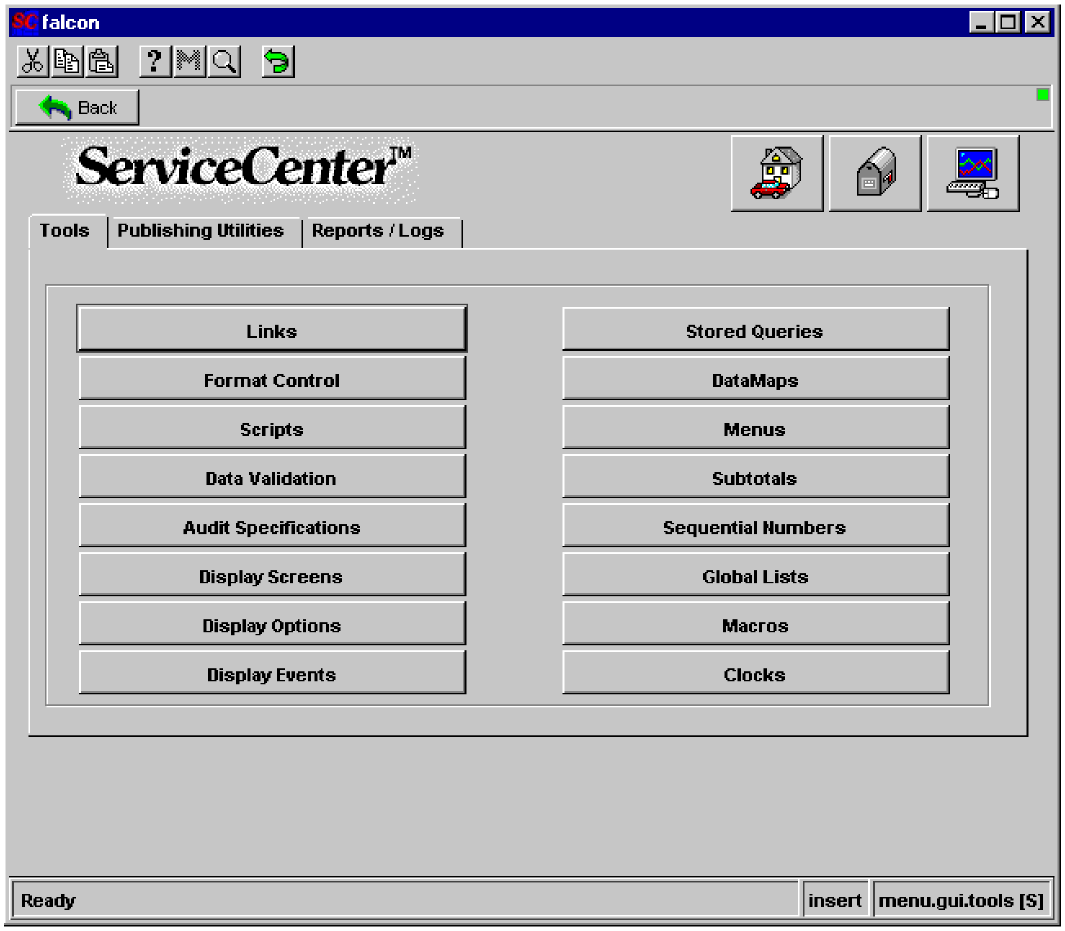


Figure 4-2 Tools Menu

3. Click on Format Control.

   The Format Control screen is displayed.

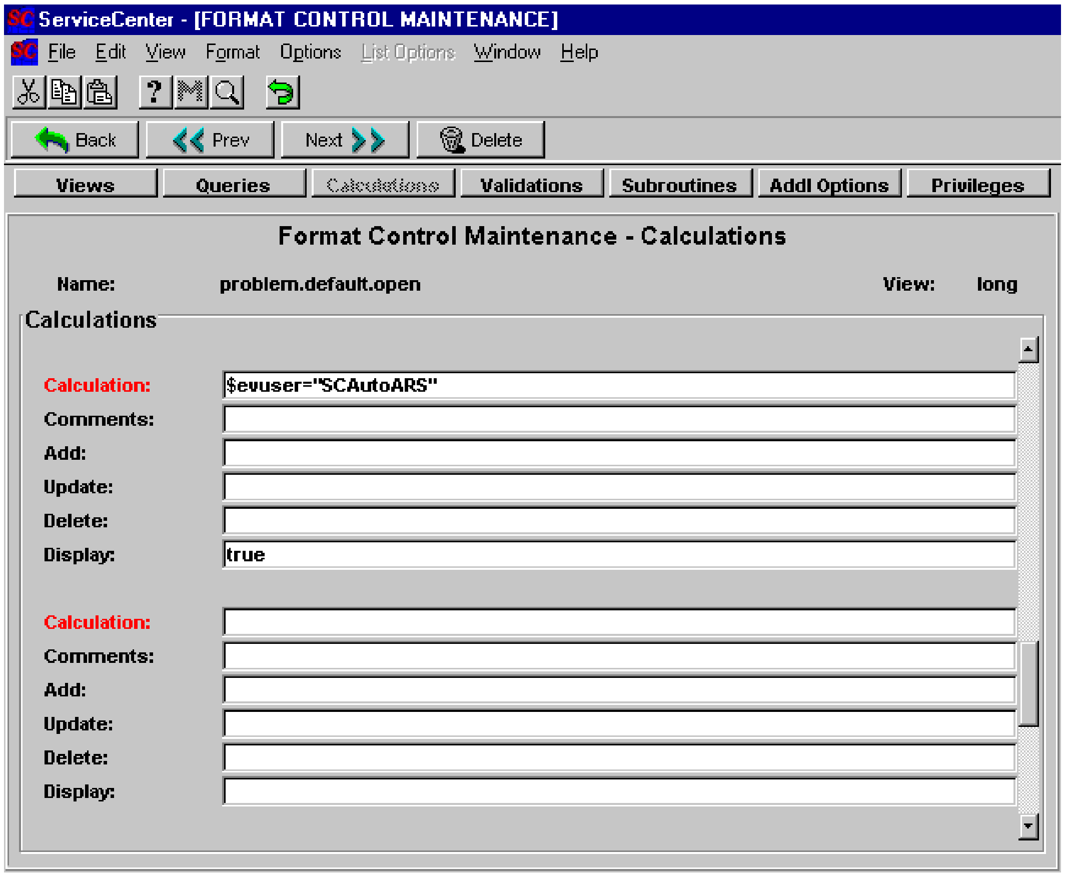
4. Type **problem.default.open** in the Name field, then click on **Search**.

   The problem.default.open Main Information screen is displayed.

5. Click on Calculations.

6. Click on **Options>Show Expanded Form**.

The form is displayed.


ServiceCenter - [FORMAT CONTROL MAINTENANCE]
File Edit View Format Options List Options Window Help
Back
Prev
Next
Delete
Views
Queries
Calculations
Validations
Subroutines
Addl Options
Privileges

Format Control Maintenance - Calculations

Name: problem.default.open View: long

Calculations

Calculation:	$evuser="SCAutoARS"
Comments:	
Add:	
Update:	
Delete:	
Display:	true
Calculation:	
Comments:	
Add:	
Update:	
Delete:	
Display:	

Figure 4-3 Format Control Calculation

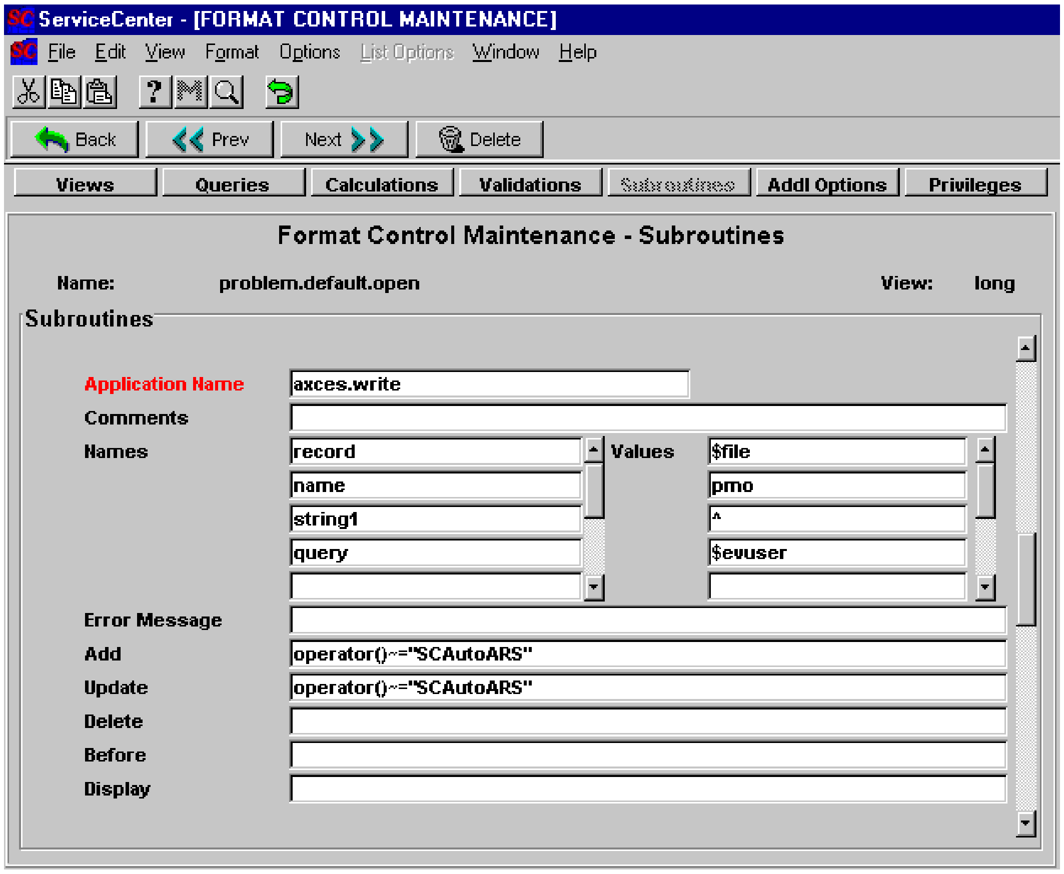
7. Scroll down to an empty calculation section and enter the calculation **$evuser=”SCAutoARS”** onto the Calculation line. Then input the word **true** on the Display line.

8. Click on Subroutines.

   The form is displayed.

9. Scroll down to an empty line and fill in the following information:

Application Name:	axces.write		
Names:	record	**Values:**	$file
	name		pmo
	string1		^
	query		$evuser
Add:	operator()~="SCAutoARS"		
Update:	operator()~="SCAutoARS"		


ServiceCenter - [FORMAT CONTROL MAINTENANCE]
File Edit View Format Options List Options Window Help
Back
Prev
Next
Delete
Views
Queries
Calculations
Validations
Subroutines
Addl Options
Privileges

Format Control Maintenance - Subroutines

Name: problem.default.open View: long

Subroutines

Application Name	axces.write		
Comments			
Names	record	Values	$file
	name		pmo
	string1		^
	query		$evuser
Error Message			
Add	operator()~="SCAutoARS"		
Update	operator()~="SCAutoARS"		
Delete			
Before			
Display			

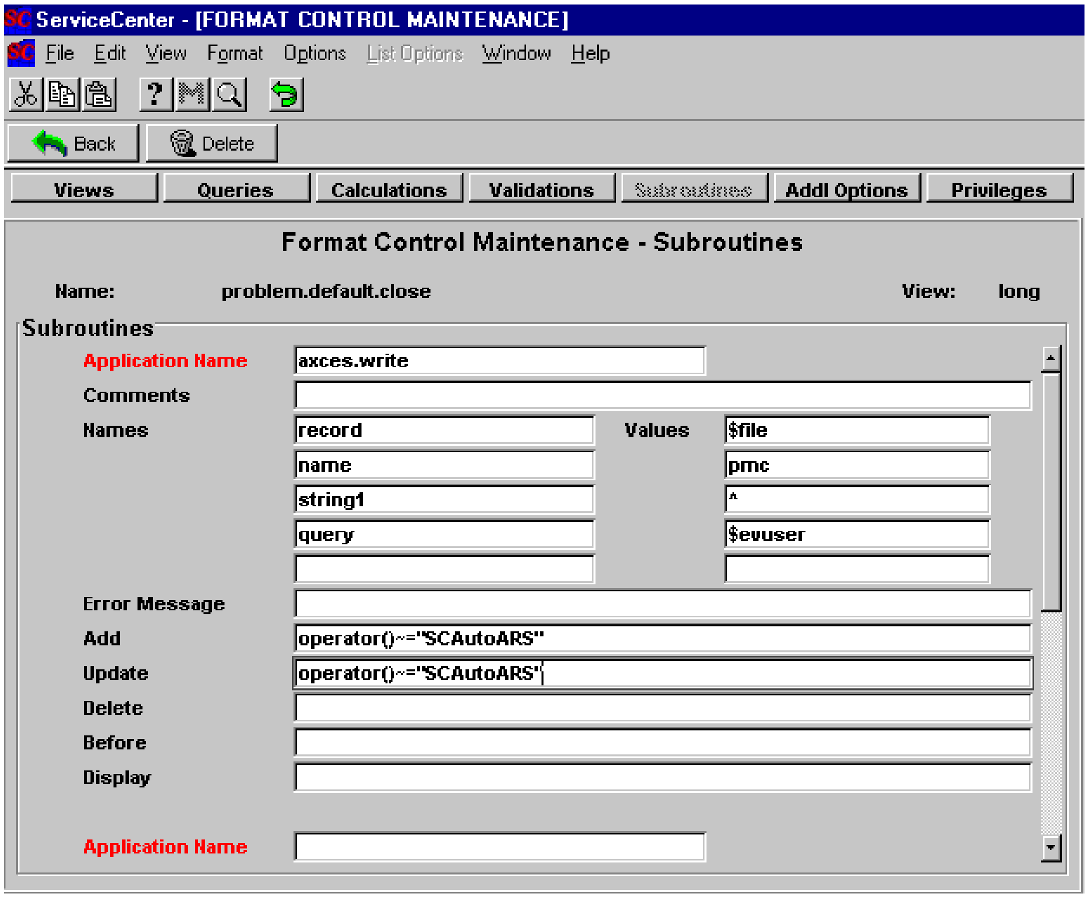
Figure 4-4 problem.default.open

You must now apply the same changes to **problem.default.close**.

10. Repeat step 4 through step 9, this time for problem.default.close. In step 9, fill in the following information:

Application Name: axces.write

Names:	record	**Values:**	$file
	name		pmc
	string1		^
	query		$evuser
Add:	operator()~=”SCAutoARS”		
Update:	operator()~=”SCAutoARS”		


ServiceCenter - [FORMAT CONTROL MAINTENANCE]
File Edit View Format Options List Options Window Help
Back
Delete
Views
Queries
Calculations
Validations
Subroutines
Addl Options
Privileges

Format Control Maintenance - Subroutines

Name: problem.default.close View: long

Subroutines

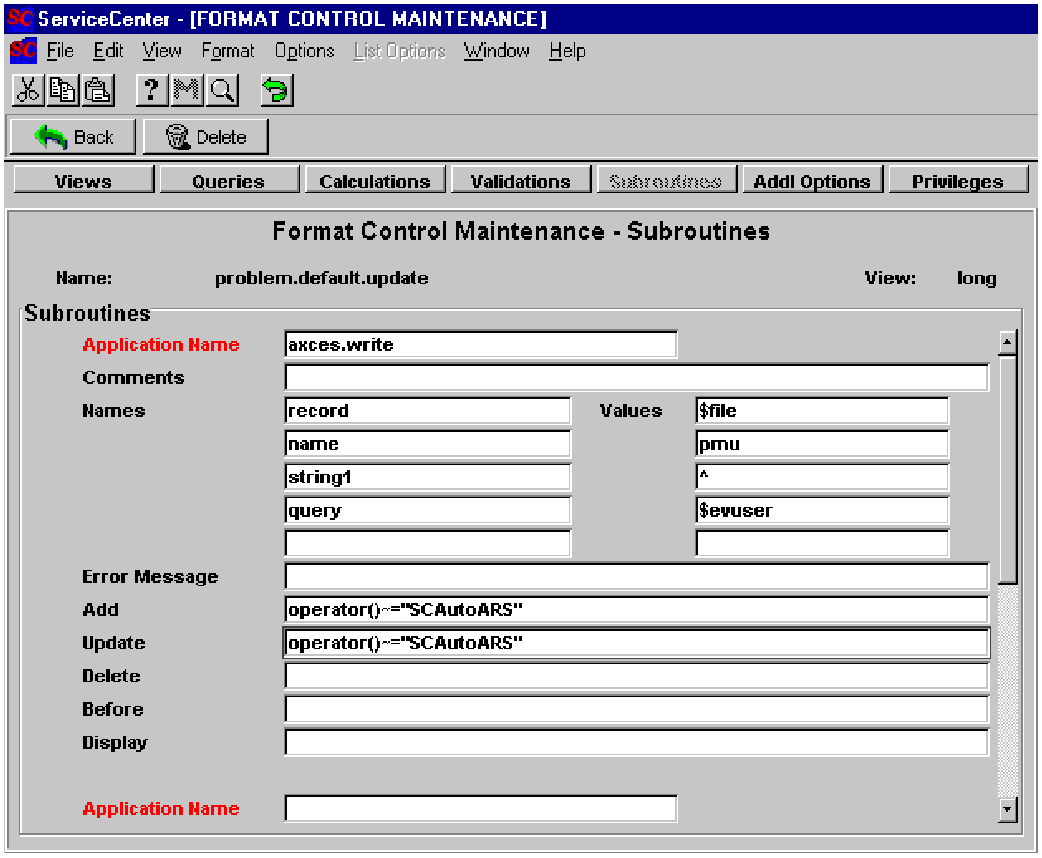
Application Name	axces.write		
Comments			
Names	record	Values	$file
	name		pmc
	string1		^
	query		$evuser
Error Message			
Add	operator()~="SCAutoARS"		
Update	operator()~="SCAutoARS"		
Delete			
Before			
Display			
Application Name			

Figure 4-5 problem.default.close

You must now apply the same changes to **problem.default.update**.

11. Repeat step 4 through step 9, this time for problem.default.update. In step 9, fill in the following information:

Application Name:	axces.write		
Names:	record	**Values:**	$file
	name		pmu
	string1		^
	query		$evuser
Add:	operator()~="SCAutoARS"		
Update:	operator()~="SCAutoARS"		


ServiceCenter - [FORMAT CONTROL MAINTENANCE]
File Edit View Format Options List Options Window Help
Back
Delete
Views
Queries
Calculations
Validations
Subroutines
Addl Options
Privileges

Format Control Maintenance - Subroutines

Name: problem.default.update View: long

Subroutines

Application Name	axces.write		
Comments			
Names	record	Values	$file
	name		pmu
	string1		^
	query		$evuser
Error Message			
Add	operator()~="SCAutoARS"		
Update	operator()~="SCAutoARS"		
Delete			
Before			
Display			
Application Name			

Figure 4-6 problem.default.update

12. Finally, return to the ServiceCenter's main menu, ensuring that you save your changes when asked.

Database Dictionary

The **scars.js** script delivered with SCAuto for Remedy ARS assumes you have made the following changes to the problem database, to add a field called **remedy.no** to hold the ARS problem ticket number. This is the recommended approach. Although the existing field **reference.no** could serve this purpose, that field is often used by other Peregrine SCAuto adapters, for example, SCAuto for HP/IT Operations, ServiceCenterPlus for Tivoli, and SCAuto for Unicenter TNG all use this field to correlate ServiceCenter problem tickets with event console messages.

1. Select the Toolkit tab from ServiceCenter's main menu.

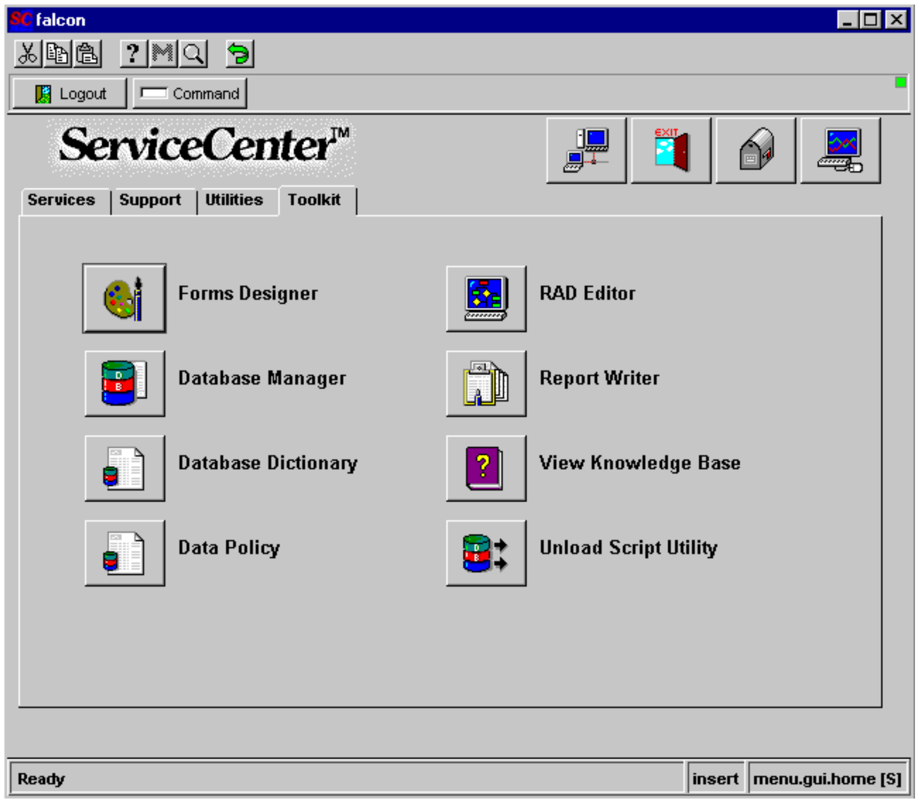
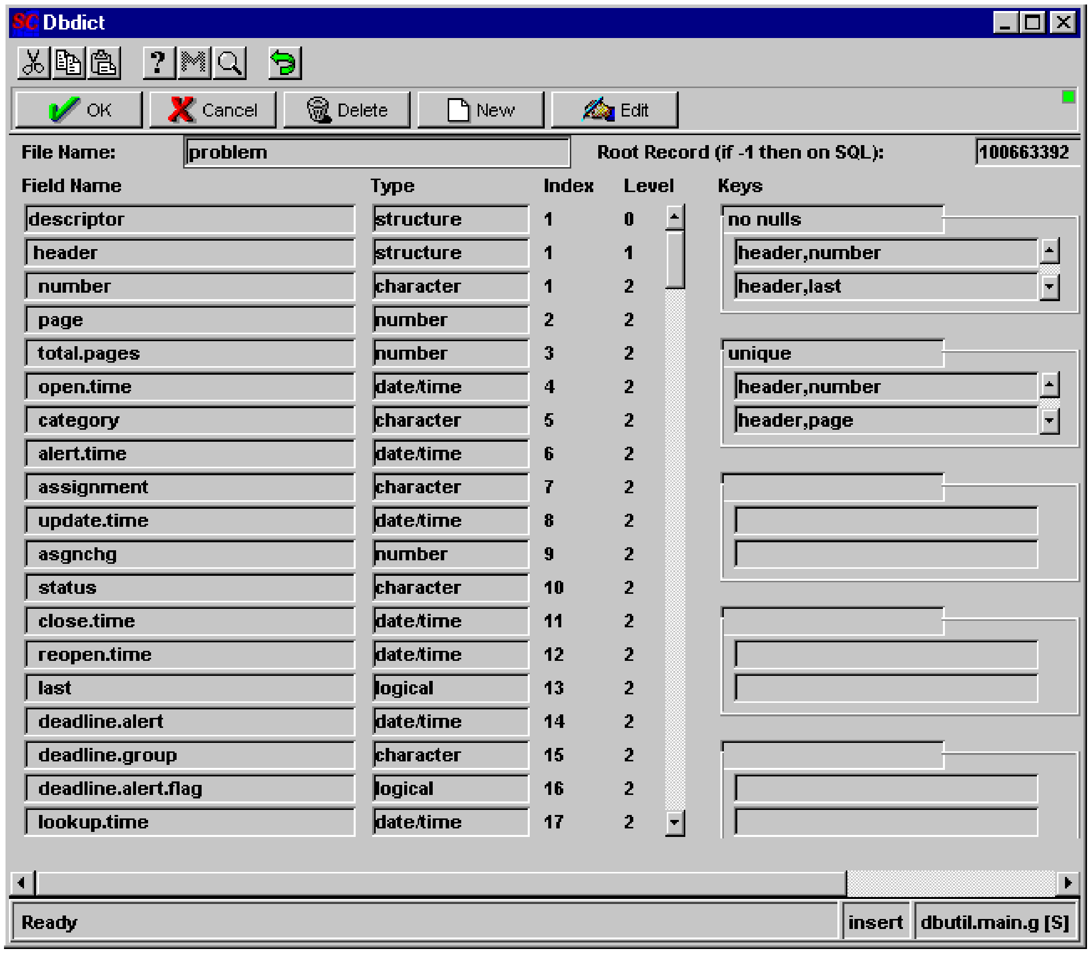


Figure 4-7 Toolkit

2. Click on Database Dictionary.
3. Type *problem* in the File Name field, then click on the **Search** button.

The problem's dictionary record is displayed.



File Name: problem   Root Record (if -1 then on SQL): 100663392

Field Name	Type	Index	Level
descriptor	structure	1	0
header	structure	1	1
number	character	1	2
page	number	2	2
total.pages	number	3	2
open.time	date/time	4	2
category	character	5	2
alert.time	date/time	6	2
assignment	character	7	2
update.time	date/time	8	2
asgnchg	number	9	2
status	character	10	2
close.time	date/time	11	2
reopen.time	date/time	12	2
last	logical	13	2
deadline.alert	date/time	14	2
deadline.group	character	15	2
deadline.alert.flag	logical	16	2
lookup.time	date/time	17	2

Keys
no nulls
header,number
header,last
unique
header,number
header,page

Ready   insert   dbutil.main.g [S]

Figure 4-8 Problem

You now add new fields to the header structure of the problem dictionary.

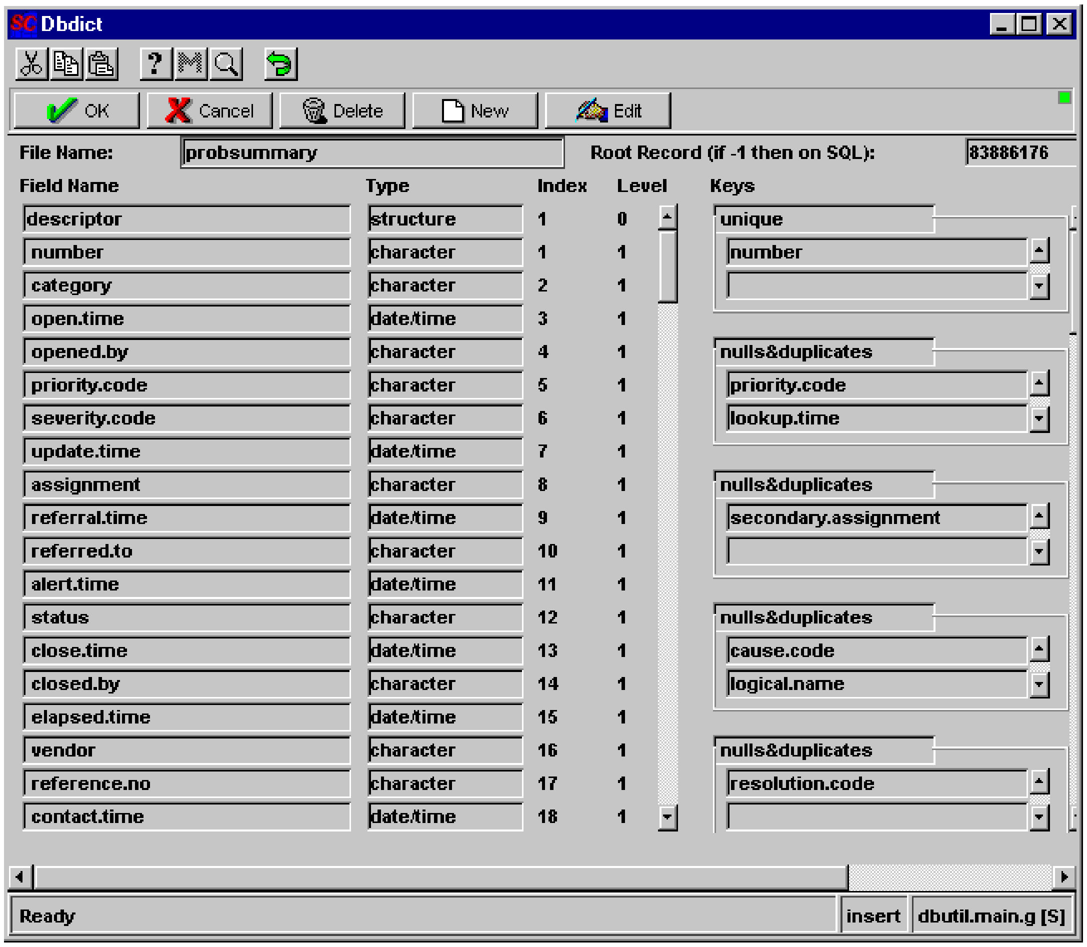
4. Click in the header box and click on **New**.

   A screen is displayed that allows you to add fields to the header structure.

5. Add the following fields:

Name	**Type**
remedy.no	character

6. Repeat this process (step 3 through step 5) to add the same fields to the **probsummary** dictionary record. This time, place the fields in the descriptor structure instead of the header structure.



File Name: probsummary
Root Record (if -1 then on SQL): 83886176

Field Name	Type	Index	Level
descriptor	structure	1	0
number	character	1	1
category	character	2	1
open.time	date/time	3	1
opened.by	character	4	1
priority.code	character	5	1
severity.code	character	6	1
update.time	date/time	7	1
assignment	character	8	1
referral.time	date/time	9	1
referred.to	character	10	1
alert.time	date/time	11	1
status	character	12	1
close.time	date/time	13	1
closed.by	character	14	1
elapsed.time	date/time	15	1
vendor	character	16	1
reference.no	character	17	1
contact.time	date/time	18	1

Keys

unique	number
nulls&duplicates	priority.code, lookup.time
nulls&duplicates	secondary.assignment
nulls&duplicates	cause.code, logical.name
nulls&duplicates	resolution.code

Ready | insert | dbutil.main.g [S]

Figure 4-9 probsummary

Links

To ensure that the values stored in the problem are searchable, they need to be copied to the probsummary record. This is done in the link record.

1. Select Utilities from ServiceCenter's main screen.

   The Utilities menu is displayed.

2. Click on Tools.

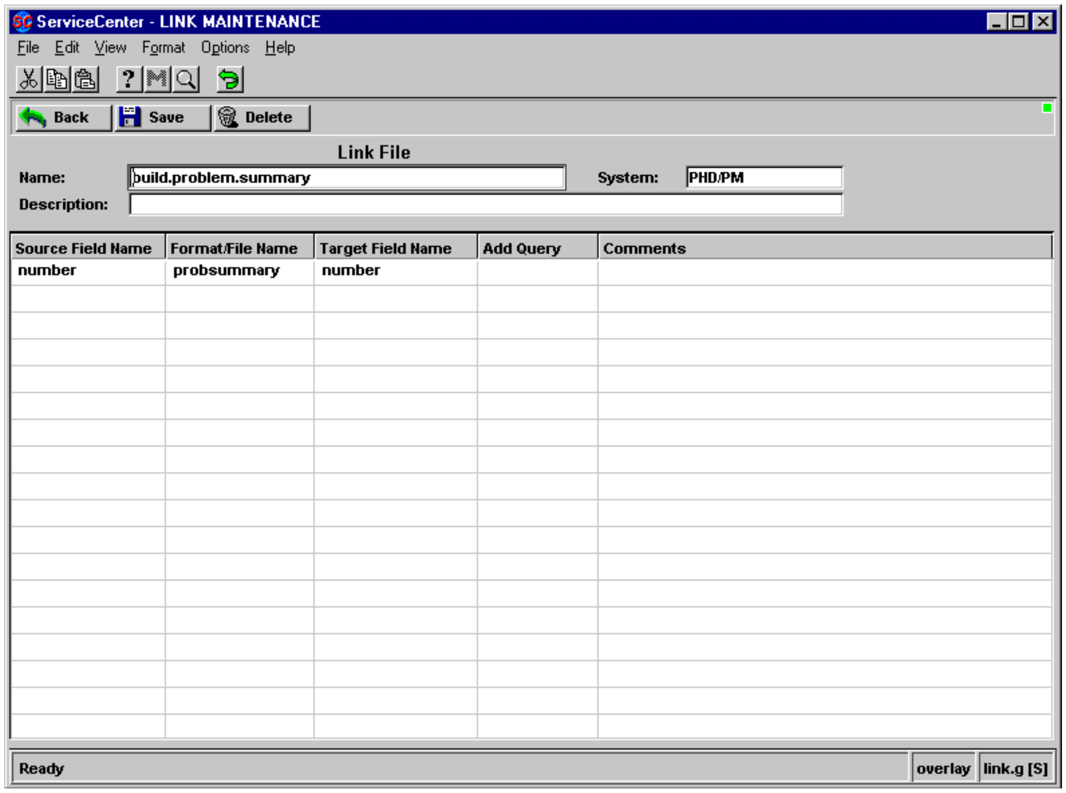
   The Tools menu is displayed.

3. Click on Links.

   The Link Manager form is displayed.

4. Name the form **build.problem.summary**.

5. Click on **Search**.


ServiceCenter - LINK MAINTENANCE
File Edit View Format Options Help
Back
Save
Delete
Link File

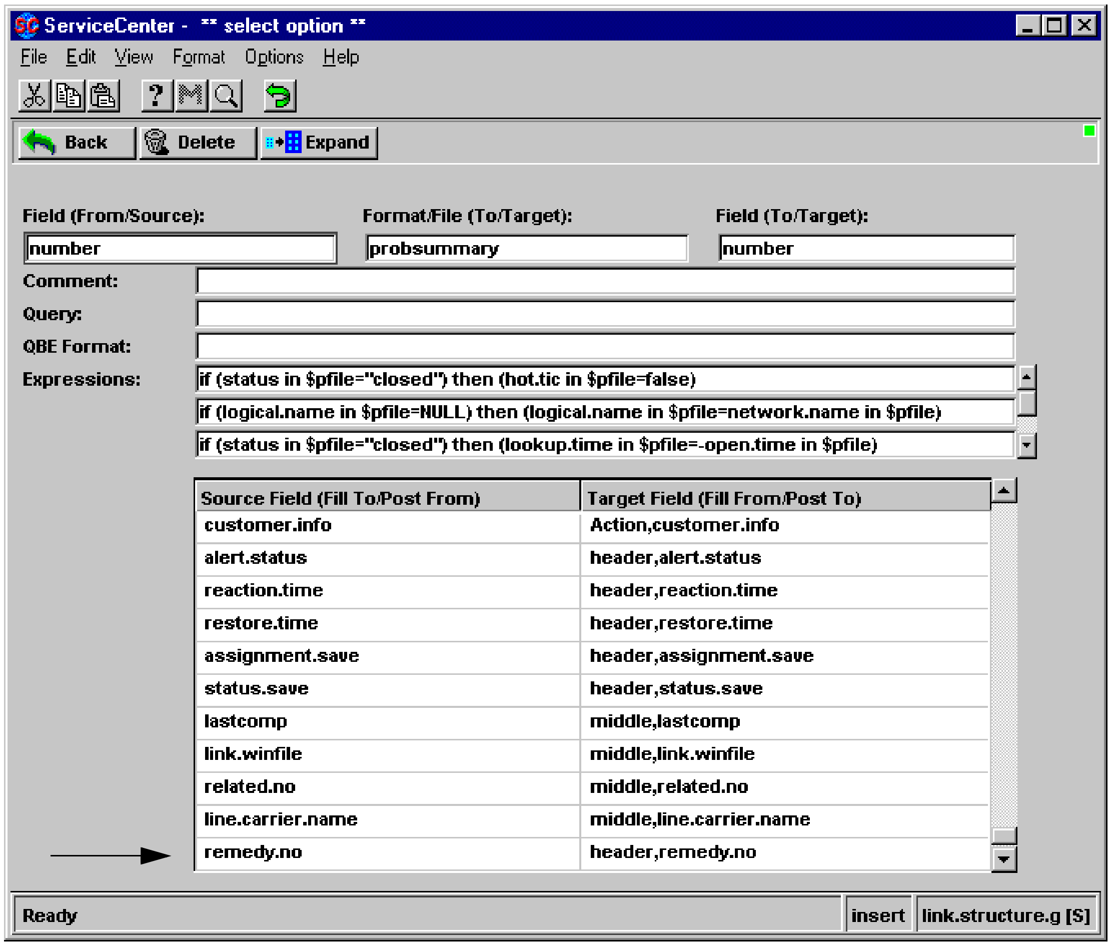
Name:	build.problem.summary	System:	PHD/PM
Description:			

Source Field Name	Format/File Name	Target Field Name	Add Query	Comments
number	probsummary	number		

Ready | overlay | link.g [S]

Figure 4-10 Link File

6. Click anywhere on the *number* record.
7. Click on Options>Select Line.


ServiceCenter - ** select option **
File Edit View Format Options Help
Back
Delete
Expand

Field (From/Source):	number
Format/File (To/Target):	probsummary
Field (To/Target):	number
Comment:	
Query:	
QBE Format:	
Expressions:	if (status in $pfile="closed") then (hot.tic in $pfile=false)
	if (logical.name in $pfile=NULL) then (logical.name in $pfile=network.name in $pfile)
	if (status in $pfile="closed") then (lookup.time in $pfile=-open.time in $pfile)

Source Field (Fill To/Post From)	Target Field (Fill From/Post To)
customer.info	Action,customer.info
alert.status	header,alert.status
reaction.time	header,reaction.time
restore.time	header,restore.time
assignment.save	header,assignment.save
status.save	header,status.save
lastcomp	middle,lastcomp
link.winfile	middle,link.winfile
related.no	middle,related.no
line.carrier.name	middle,line.carrier.name
remedy.no	header,remedy.no

Ready | insert | link.structure.g [S]

Figure 4-11 Number

8. Scroll to the bottom of the list of fields (see arrow above) and add the following:

Source Field	**Target Field**
remedy.no	header,remedy.no

9. Finally, return to the ServiceCenter's main menu, ensuring that you save your changes when asked.

Event Maps

Any field you wish to map in the JScript must appear in the appropriate event maps for both input and output.

1. Select the Utilities tab from ServiceCenter's main menu.

   The Utilities Menu is displayed.

2. Click on Event Services.

   The Event Services menu is displayed.

3. Click on the Administration tab.

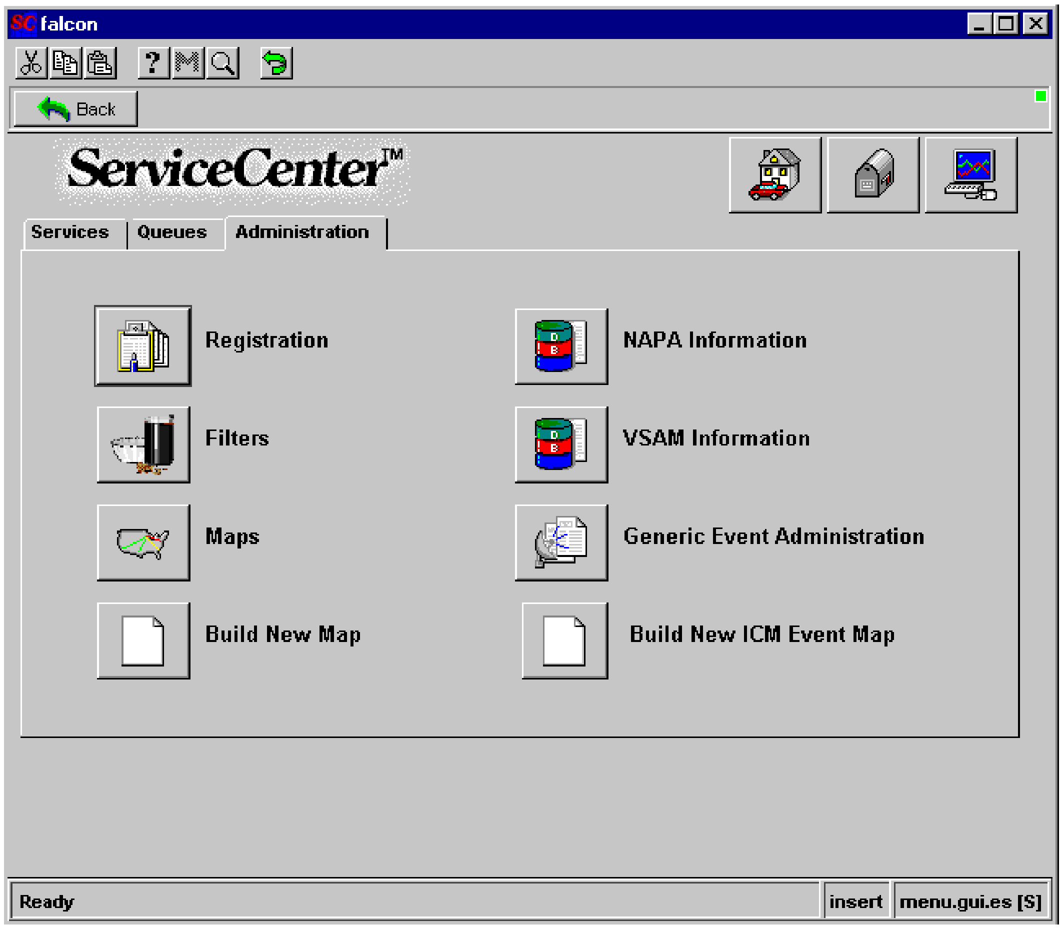


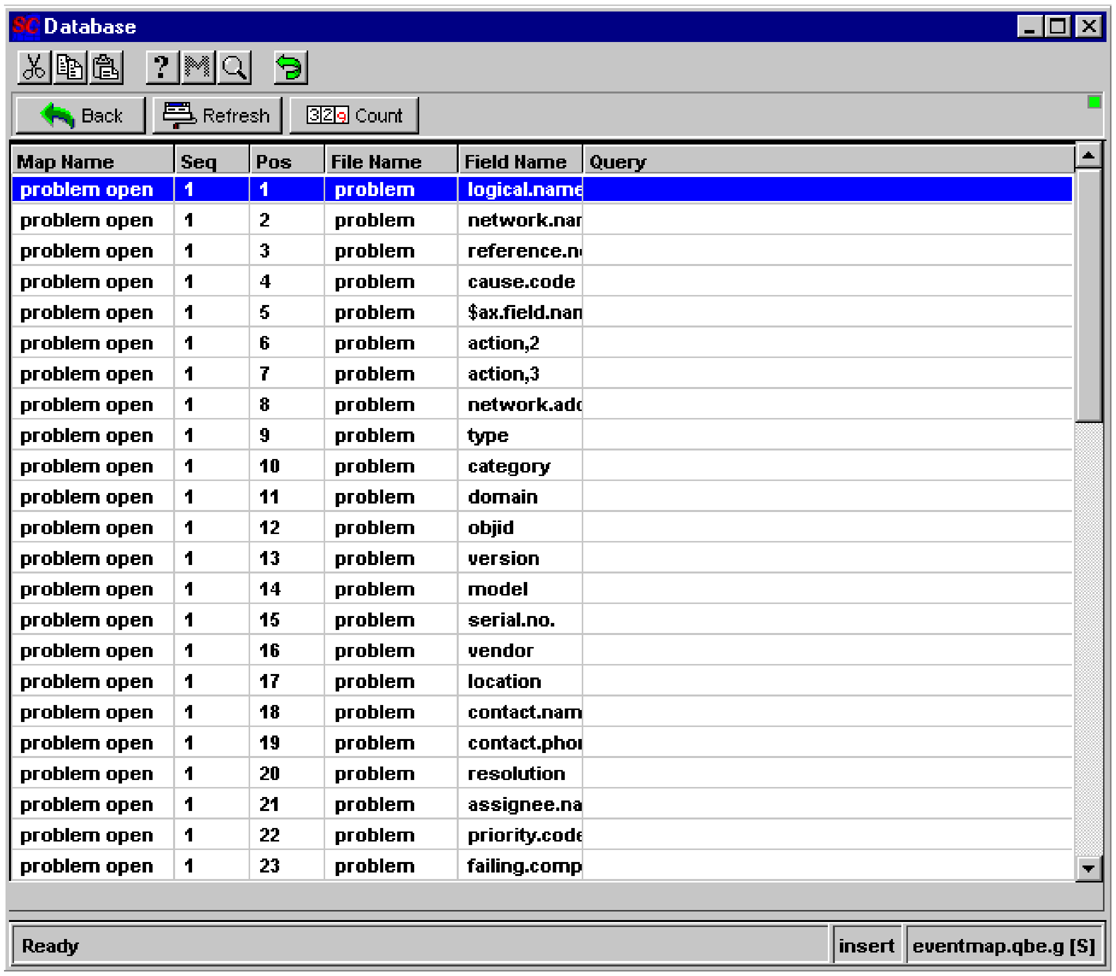
Figure 4-12 Administration

4. Click on Maps.

   An Event Map form is displayed.

5. Type *problem open* in the Map Name field, then click on **Search**.

   The problem open (pmo) map is displayed.



Database

Back | Refresh | Count

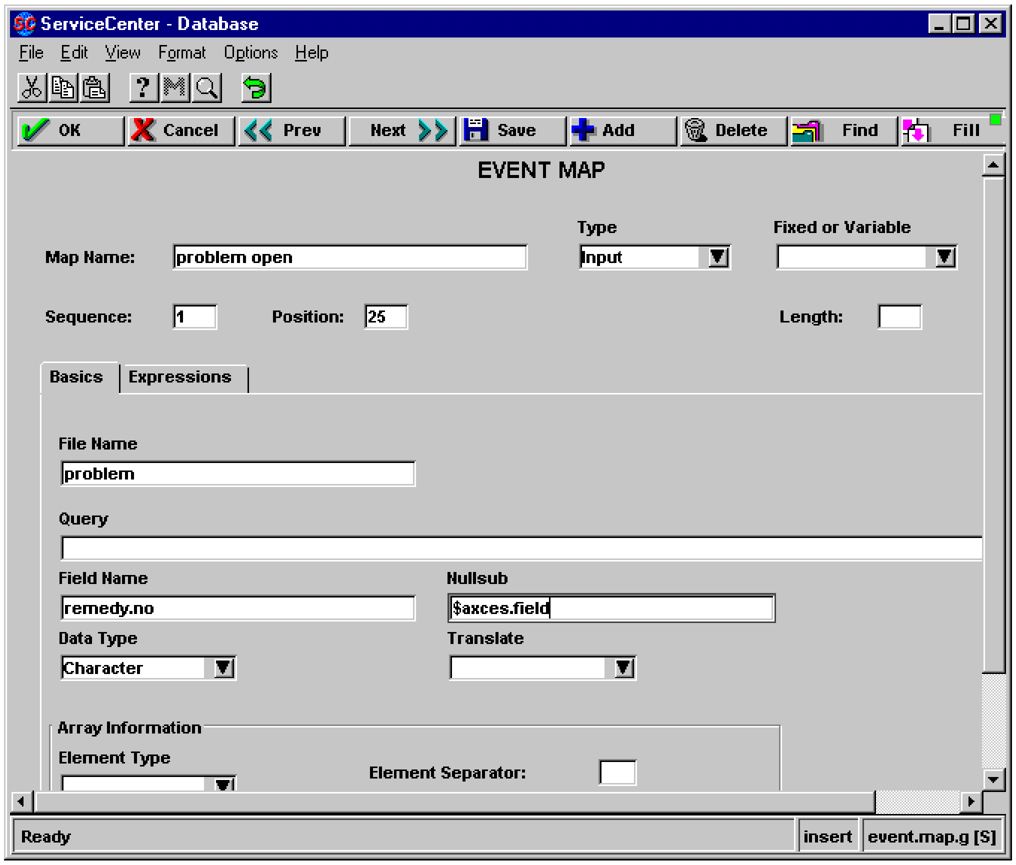
Map Name	Seq	Pos	File Name	Field Name	Query
problem open	1	1	problem	logical.name	
problem open	1	2	problem	network.nar	
problem open	1	3	problem	reference.n	
problem open	1	4	problem	cause.code	
problem open	1	5	problem	$ax.field.nan	
problem open	1	6	problem	action,2	
problem open	1	7	problem	action,3	
problem open	1	8	problem	network.adc	
problem open	1	9	problem	type	
problem open	1	10	problem	category	
problem open	1	11	problem	domain	
problem open	1	12	problem	objid	
problem open	1	13	problem	version	
problem open	1	14	problem	model	
problem open	1	15	problem	serial.no.	
problem open	1	16	problem	vendor	
problem open	1	17	problem	location	
problem open	1	18	problem	contact.nam	
problem open	1	19	problem	contact.phoı	
problem open	1	20	problem	resolution	
problem open	1	21	problem	assignee.na	
problem open	1	22	problem	priority.code	
problem open	1	23	problem	failing.comp	

Ready | insert | eventmap.qbe.g [S]

Figure 4-13 problem open (pmo) map

6. Scroll down to the last line with a File Name of problem. This will be the last field for problem open (pmo) - Input.
7. Click on the **problem open** button to bring up the event map.

The problem open input event map is displayed. Note that the Type is Input.


ServiceCenter - Database
File Edit View Format Options Help
OK Cancel Prev Next Save Add Delete Find Fill

EVENT MAP

Map Name: problem open
Type: Input
Fixed or Variable:
Sequence: 1
Position: 25
Length:

Basics | Expressions

File Name: problem
Query:
Field Name: remedy.no
Nullsub: $axces.field
Data Type: Character
Translate:

Array Information
Element Type:
Element Separator:

Ready | insert | event.map.g [S]

Figure 4-14 problem open (pmo) Input Event Map

8. Edit the Field Name and Data Type one at a time and edit the position to the next number. Use the following values:

Field Name	**Data Type**
remedy.no	character

Note: The field positions for problem close (pmc) Output and problem open (pmo) Output are different than for problem update (pmu) Output. It is entirely possible for the field positions to be different on your setup. problem open (pmo) - Input

9. Click on **Add**.

Note: Be careful to use the **Add** button, not the **Save** button. Using **Save** will overwrite the record you were editing.

10. Press cancel to return to the problem open (pmo) map.
11. Scroll down and select the last *problem open*. This should have no filename and should have a Type of *Output* once selected.
12. Edit the Position to the next number.
13. Edit the Field Name fields. You must input a data type. Use the following values:

    problem open (pmo) - Output

Field Name	**Data Type**
remedy.no	character

14. Repeat the process (step 6 through step 13) for *problem update* and *problem close*. Use the same values. They may end up as different position numbers.

    You should add the following fields to both your input and output maps as well, if they are not already there.

    - full.name
    - brief.description
    - severity.code
    - problem.status
    - status

    To do so:

15. Repeat the process for each problem open - input, problem open - output, problem update - input, problem update - output, problem close - input and problem close - output. Prior to adding the new field, full.name or any field to a map, you should always look to ensure that it does not already exist.

Forms Designer

After adding the field *remedy.no*, you may want to display this field in various forms within ServiceCenter Problem Management.

1. Click on the Toolkit tab from ServiceCenter's main menu.

   The Toolkit menu is displayed.

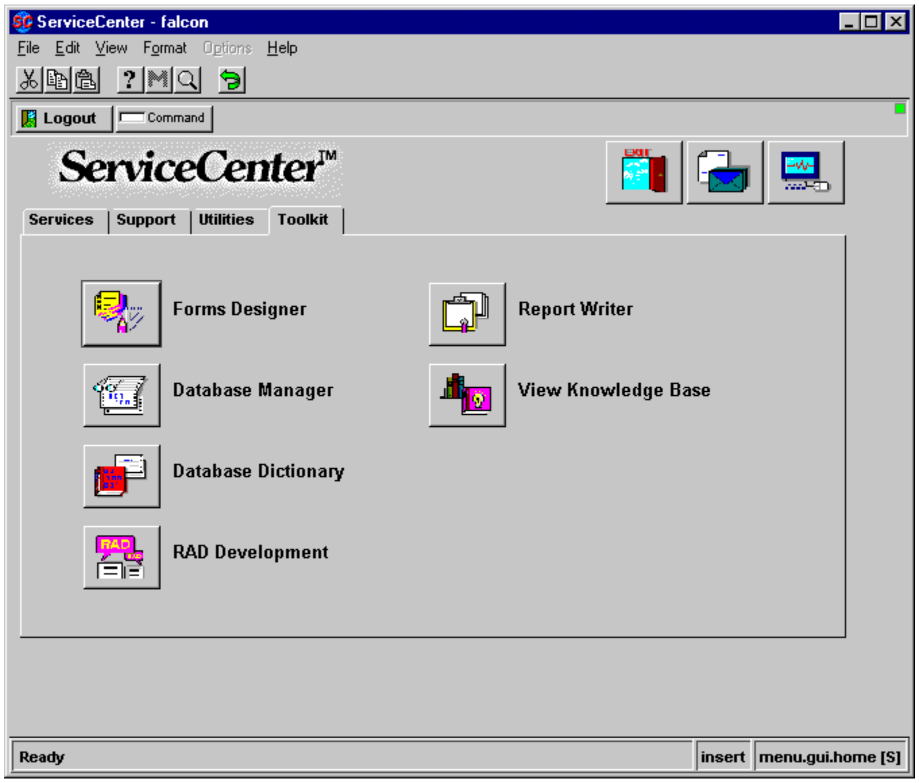
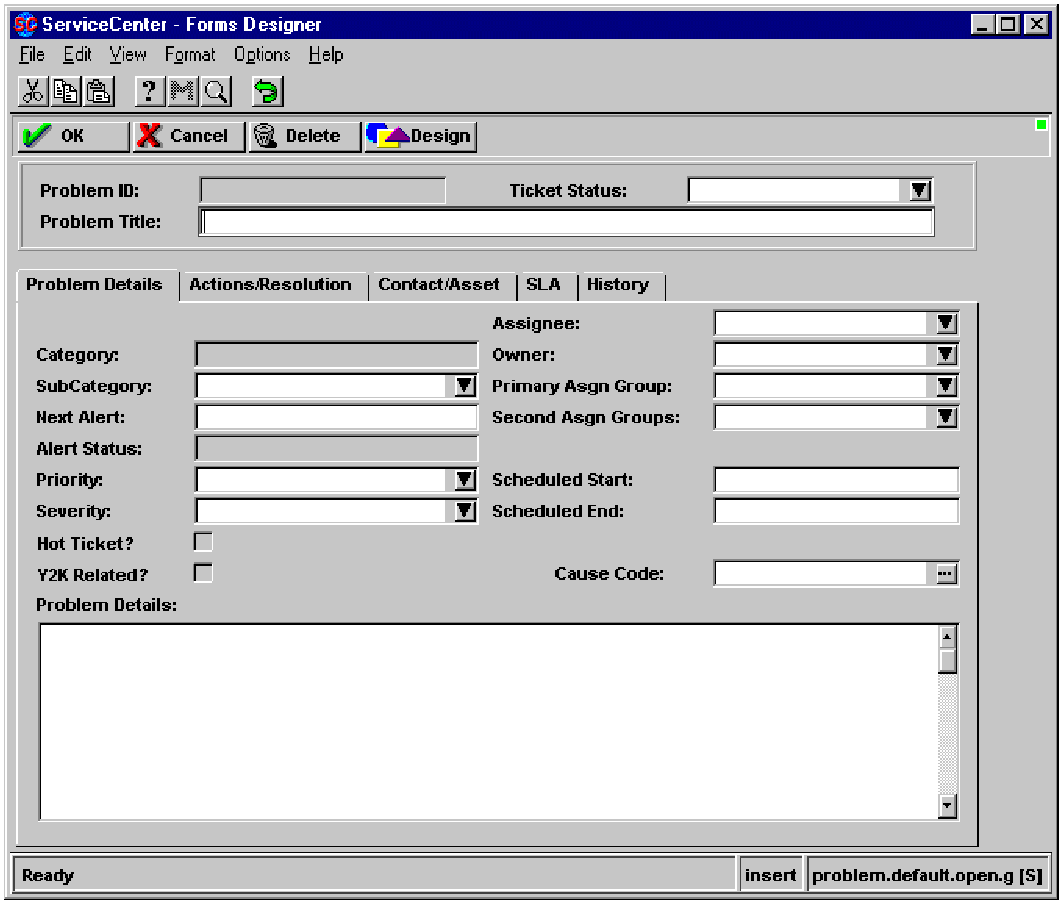


Figure 4-15 Toolkit

2. Click on Forms Designer.
3. Type the name of the form you want to edit. In this example, we are using *problem.default.open.g*. The g stands for GUI (Graphical User Interface).
4. Press **Enter**.

The problem.default.open.g form is displayed.



Problem ID:
Ticket Status:
Problem Title:

Category:
SubCategory:
Next Alert:
Alert Status:
Priority:
Severity:
Hot Ticket?
Y2K Related?
Problem Details:

Assignee:
Owner:
Primary Asgn Group:
Second Asgn Groups:
Scheduled Start:
Scheduled End:
Cause Code:

Ready | insert | problem.default.open.g [S]

Figure 4-16 problem.default.open.g form

5. Click on the **Design** button to bring up your design tools.

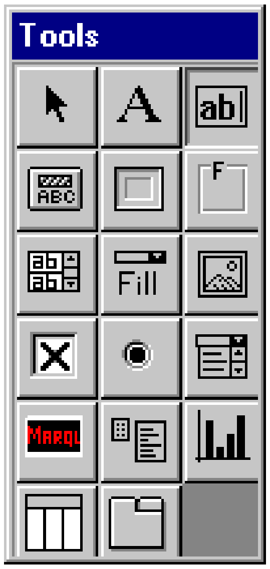
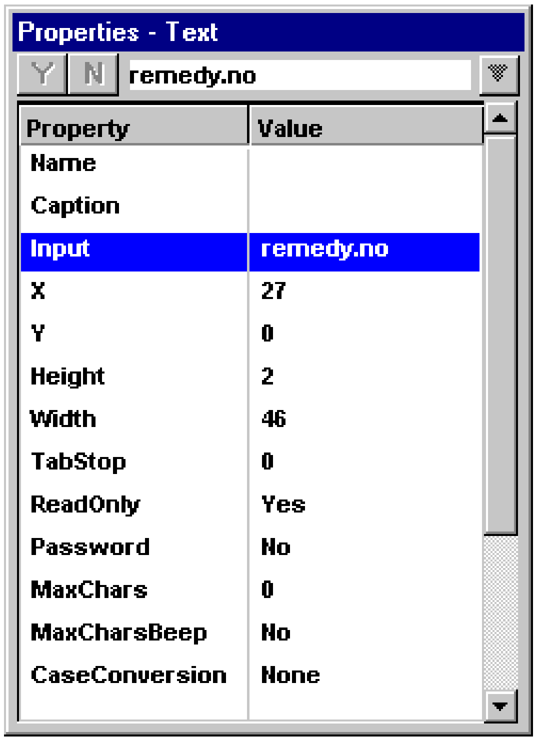


Figure 4-17 Tools

6. Click on the Text tool.
7. Click on your form where you would like the new item placed.
8. Adjust the size and position until you are satisfied.

9. Type a value in the Properties box for *Input*. This tells the Forms Designer where to get the data for the field. You may also want to make the field Read Only.


Properties - Text

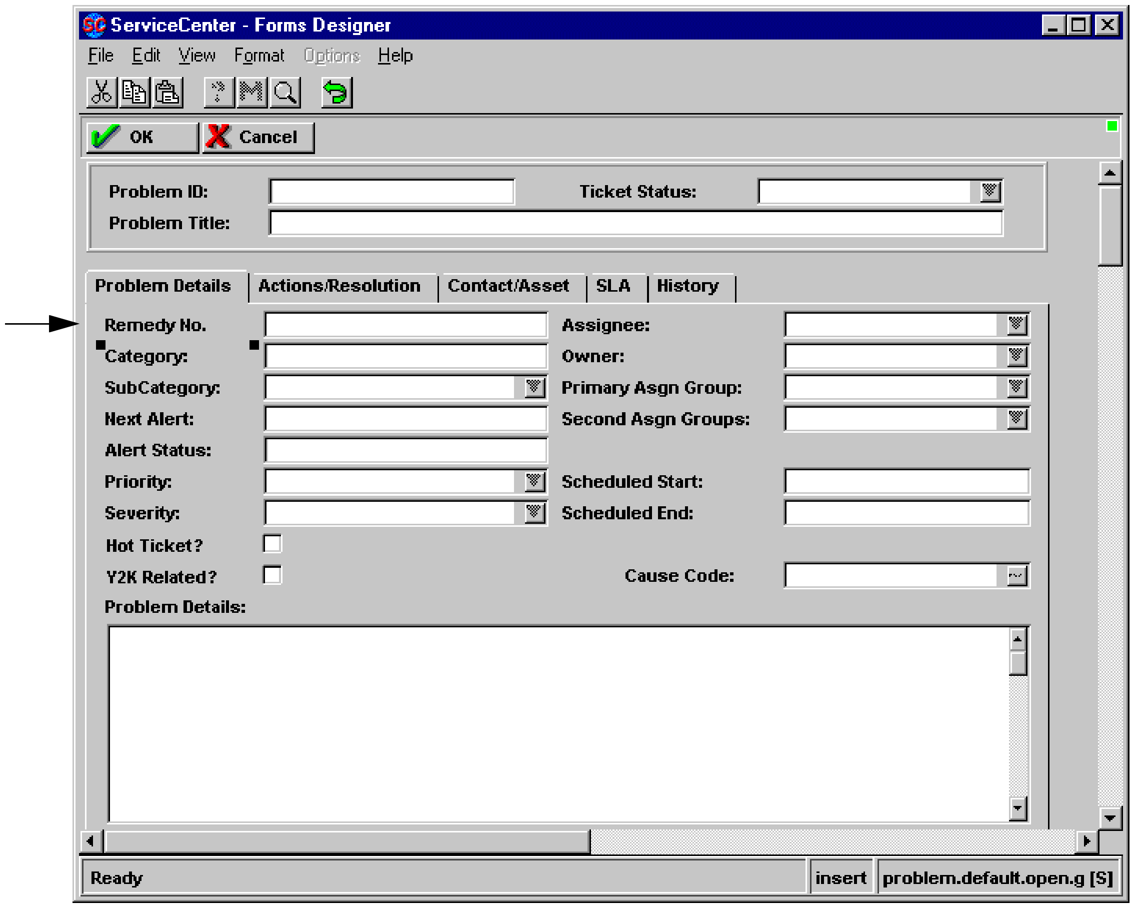
remedy.no

Property	Value
Name	
Caption	
Input	remedy.no
X	27
Y	0
Height	2
Width	46
TabStop	0
ReadOnly	Yes
Password	No
MaxChars	0
MaxCharsBeep	No
CaseConversion	None

Figure 4-18 Properties

10. Click on the Text tool to add a label for our new field.
11. Click on your form and place the field where you want.
12. Fill in the Caption in the Properties box.

This should give you a final form like this. The arrow below shows the addition of the **Remedy No.** field.



ServiceCenter - Forms Designer

File Edit View Format Options Help

OK Cancel

Problem ID: Ticket Status:
Problem Title:

Problem Details | Actions/Resolution | Contact/Asset | SLA | History

Remedy No.		Assignee:	
Category:		Owner:	
SubCategory:		Primary Asgn Group:	
Next Alert:		Second Asgn Groups:	
Alert Status:			
Priority:		Scheduled Start:	
Severity:		Scheduled End:	
Hot Ticket?	☐		
Y2K Related?	☐	Cause Code:	

Problem Details:

Ready | insert | problem.default.open.g [S]

Figure 4-19 Finished Form

13. Repeat this process for each form in which you want the field to be displayed, such as **problem.default.close.g** and **problem.default.update.g**.

Event Registration Expressions

Event registration expressions control the processing which occurs when a particular kind of inbound or outbound event is created in ServiceCenter. For example, this is where one specifies the RAD application which is to be executed when a particular kind of event is received in the **eventin** file, and details such as whether a response event should be written to the **eventout** file.

In this section, we discuss the strategy for processing problem management events exchanged between ServiceCenter and Remedy ARS via the gateway. Note that the instructions that follow make significant modifications to the way ServiceCenter processes problem events. If you have other SCAuto applications which produce inbound problem events or process outbound problem events, it is especially important that changes be tested to ensure they do not impact these other applications.

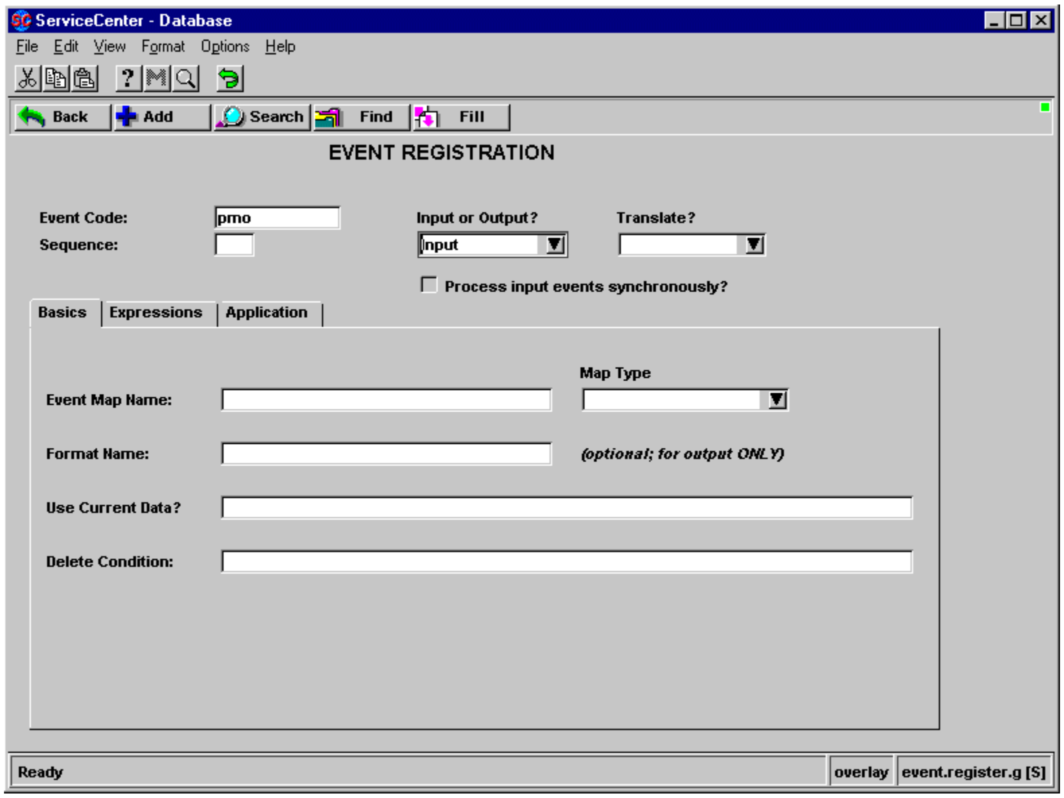
The approach taken here is to make changes to the event registration records which are conditioned upon the creator of the event being *SCAutoARS*. The intent is to suppress automatic generation of output events in response to inbound (pmu and pmc) events, since the Format Control changes previously made will generate the desired outbound (pmu and pmc) events, regardless of whether a human operator or SCAutoARS caused the Problem Management code to be executed. An alternative approach might be to create a new set of events similar to *pmo*, *pmu* and *pmc*, for the private use of the SCAutoARS application, thereby avoiding the possibility of any conflicts with other applications.

1. Select the Utilities tab from ServiceCenter's main menu.

   The Utilities menu is displayed.
2. Click on Event Services.

   The Event Services menu is displayed.
3. Click on the Administration tab.
4. Click on Registration.

The Event Registration form is displayed.


ServiceCenter - Database
File Edit View Format Options Help
Back Add Search Find Fill
Basics Expressions Application

EVENT REGISTRATION

Event Code: pmo
Sequence:
Input or Output? Input
Translate?
Process input events synchronously?

Event Map Name:
Map Type
Format Name: (optional; for output ONLY)
Use Current Data?
Delete Condition:

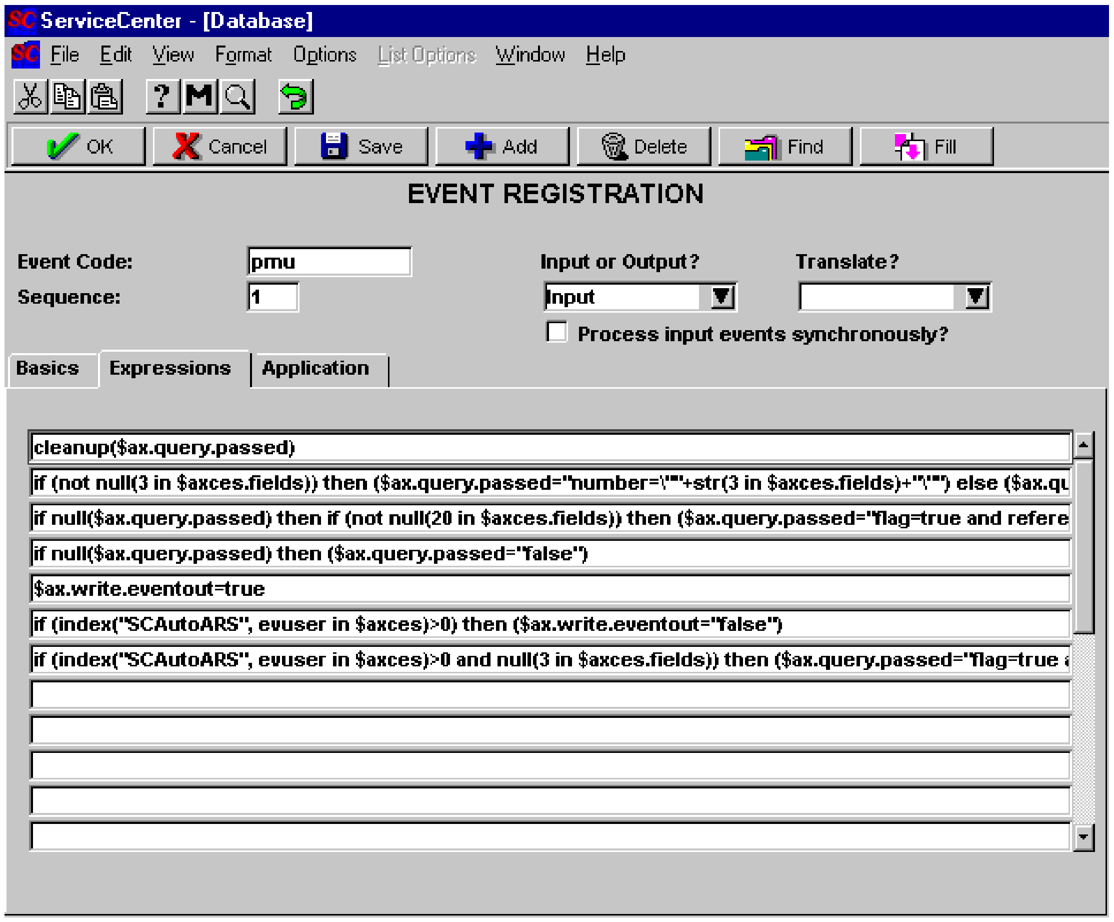
Ready | overlay | event.register.g [S]

Figure 4-20 Event Registration

5. Type *pmo* in the Event Code field.
6. Select *Input* from the Input or Output list box.
7. Click on **Search**.

   A screen is displayed that lists the records with a problem open (pmo) type.
8. Select the Expressions tab.
9. To ensure you always get an event written for all SCAutoARS open events, add the following two lines:

```
$ax.write.eventout=true
if (index("SCAutoARS", evuser in $axces)>0) then ($ax.write.eventout="true")
```


ServiceCenter - [Database]
File Edit View Format Options List Options Window Help
OK Cancel Save Add Delete Find Fill
EVENT REGISTRATION
Basics Expressions Application

Event Code: pmu
Sequence: 1
Input or Output? Input
Translate?
Process input events synchronously?

```
cleanup($ax.query.passed)
if (not null(3 in $axces.fields)) then ($ax.query.passed="number=\""+str(3 in $axces.fields)+"\"") else ($ax.qu
if null($ax.query.passed) then if (not null(20 in $axces.fields)) then ($ax.query.passed="flag=true and refere
if null($ax.query.passed) then ($ax.query.passed="false")
$ax.write.eventout=true
if (index("SCAutoARS", evuser in $axces)>0) then ($ax.write.eventout="false")
if (index("SCAutoARS", evuser in $axces)>0 and null(3 in $axces.fields)) then ($ax.query.passed="flag=true
```

Figure 4-21 pmo Expressions

10. To ensure that all pmo events open a new problem ticket (and hence do not update an existing one), we must also add the following line:

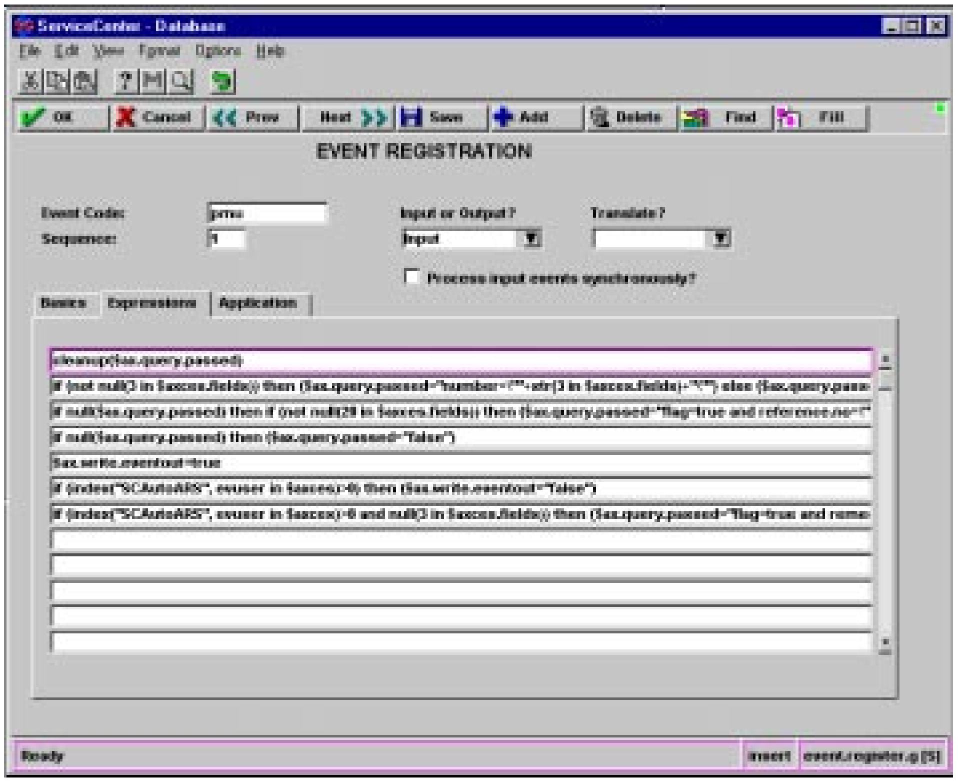
    if (index("SCAutoARS", evuser in $axces)>0) then ($ax.query.passed="false")

11. Click on the Application tab and edit the parameter values of the write eventout line to say $ax.write.eventout. This will set boolean1 to the value produced in the expressions.

12. Repeat this process for each problem update (pmu) and problem close (pmc) using the following three lines in expressions instead of the three used for problem open (pmo). Please note that *$ax.write.eventout* is now false because we do not want update and close events reflected back out.

```
$ax.write.eventout=true
if (index("SCAutoARS", evuser in $axces)>0) then ($ax.write.eventout="false")
if (index("SCAutoARS", evuser in $axces)>0 and null(3 in $axces.fields)) then
($ax.query.passed="flag=true and remedy.no=\""+str(evusrseq in $axces)+"\"")
```

Note: Due to space constraints on this page, there appears to be four lines of expressions in the text shown above. This is not the case—there are only three lines. The last line (that begins with **($ax.query.passed ......**) is a continuation of the previous line (that begins with **if (index ("SCAutoARS" ......**). Be sure to enter this information as a single line of text, not as two separate lines.



EVENT REGISTRATION

Event Code: pmu
Sequence: 1
Input or Output ? Input
Translate ?
Process input events synchronously?

```
cleanup($ax.query.passed)
if (not null(3 in $axces.fields)) then ($ax.query.passed="number=\""+str(3 in $axces.fields)+"\"") else ($ax.query.pass
if null($ax.query.passed) then if (not null(28 in $axces.fields)) then ($ax.query.passed="flag=true and reference.no=\""
if null($ax.query.passed) then ($ax.query.passed="false")
$ax.write.eventout=true
if (index("SCAutoARS", evuser in $axces)>0) then ($ax.write.eventout="false")
if (index("SCAutoARS", evuser in $axces)>0 and null(3 in $axces.fields)) then ($ax.query.passed="flag=true and reme
```

Figure 4-22 pmu Expressions

You have now finished the necessary modifications to ServiceCenter.

Chapter 5 Customizing Remedy ARS

Overview

In order for SCAuto for Remedy ARS to work properly, some customizations must be made. These changes allow:

- Remedy ARS to store the ServiceCenter ticket number
- SCAuto for Remedy ARS to tell when there is a new or modified Remedy ticket that needs to be passed on to ServiceCenter

Forms

Three fields need to be added to the schema you are using:

- ServiceCenter Ticket Number
- SCNew
- SCUpdated

This is usually the HPD:HelpDesk (used for demonstration purposes) or some other derivative.

ServiceCenter Ticket Number

The first field that must be added will hold the ServiceCenter Ticket Number. To add this field:

1. Run the Remedy Administrator and login.

A Server Window is displayed.

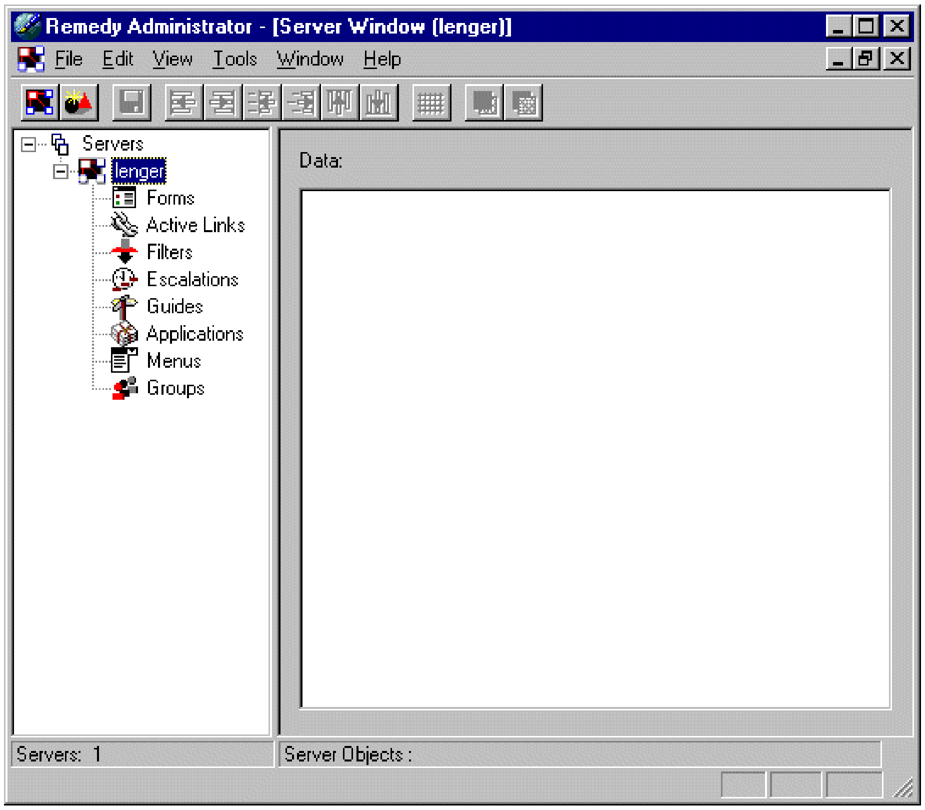


Figure 5-1 Server Window

2. Select Forms.

   A list of forms is displayed.

3. Select the form for your schema.

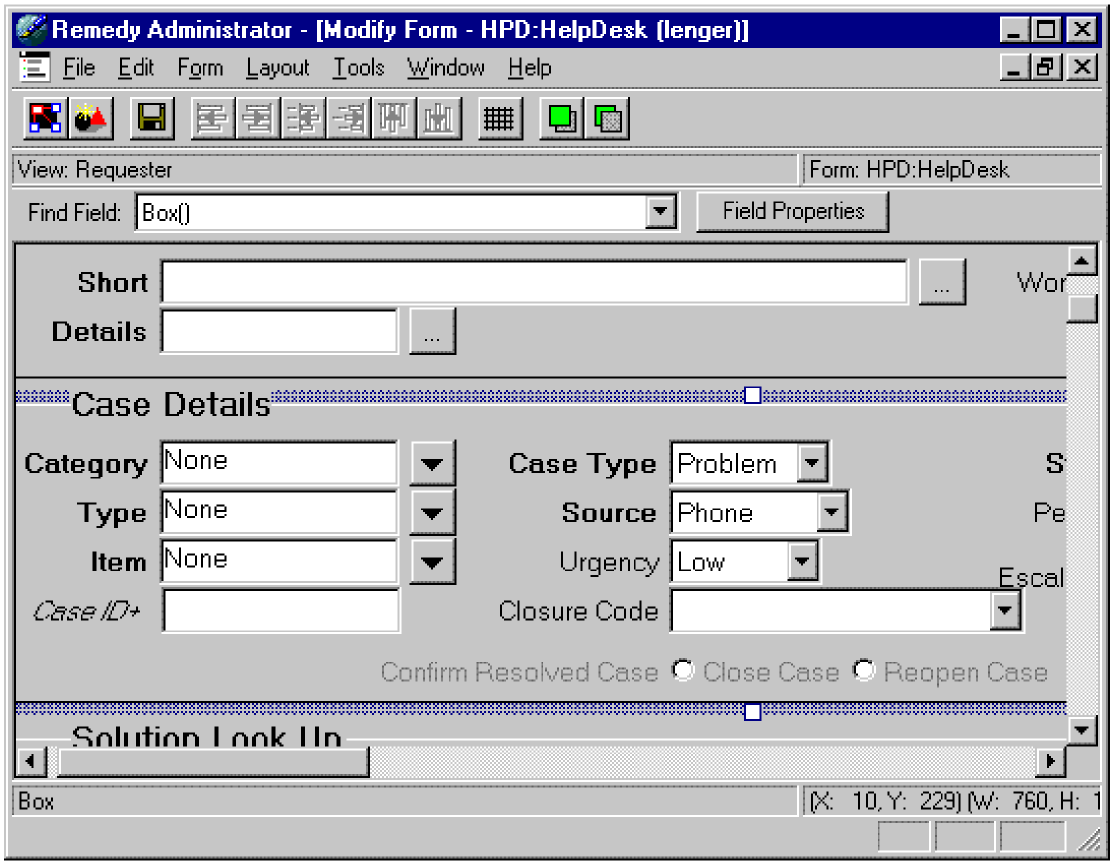


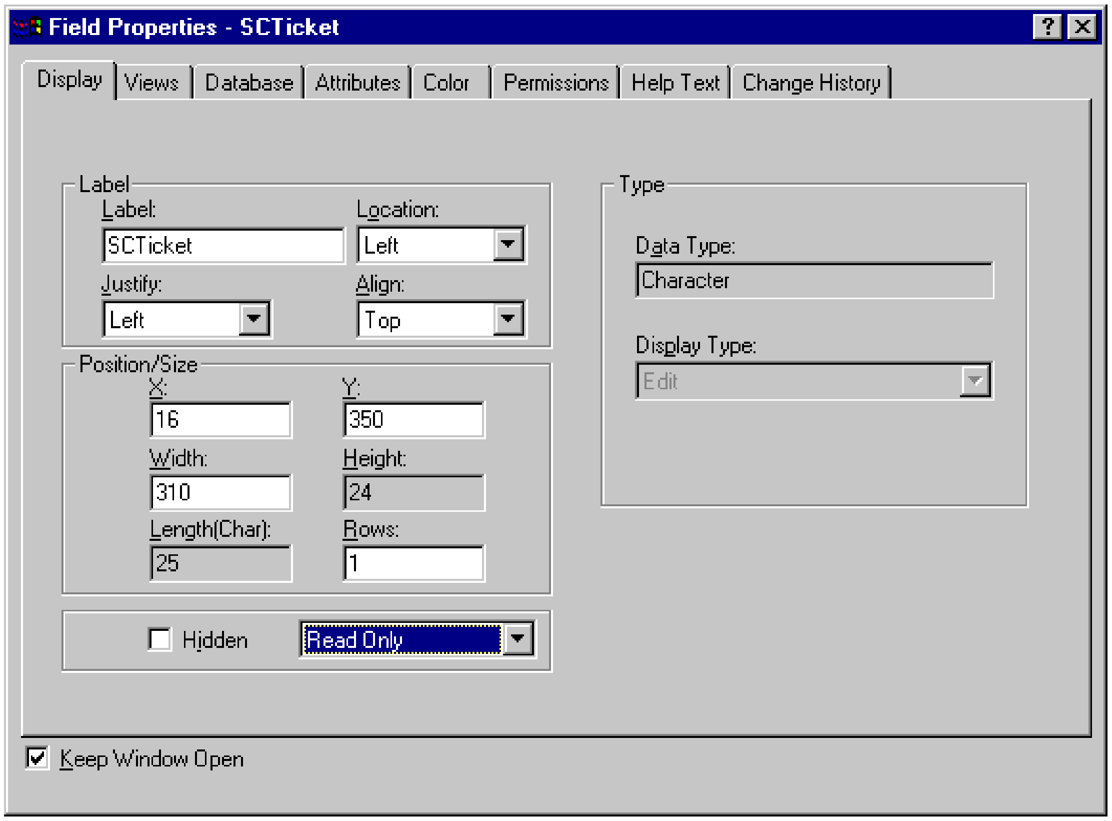
Figure 5-2 Modify Form

4. Click on **Form>Create a New>Character Field** from the top menu of the Modify form.

   A new character field appears on the form.

5. Double-Click on the new field to modify it's various attributes.

6. Click on the Display tab.


Field Properties - SCTicket

Display | Views | Database | Attributes | Color | Permissions | Help Text | Change History

Label
Label: SCTicket
Location: Left
Justify: Left
Align: Top

Position/Size
X: 16
Y: 350
Width: 310
Height: 24
Length(Char): 25
Rows: 1

☐ Hidden
Read Only

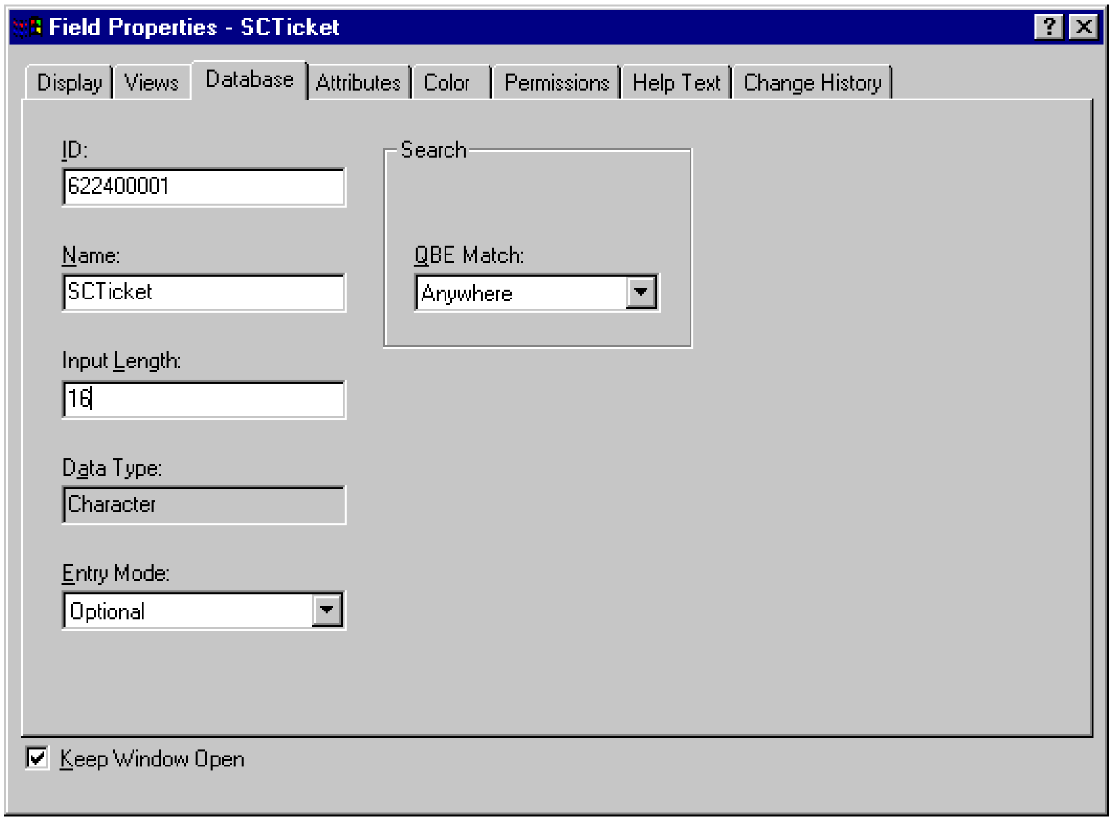
Type
Data Type: Character
Display Type: Edit

☑ Keep Window Open

Figure 5-3 Field Properties - Display

7. Change the label to *SCTicket* and make the field Read Only.

8. Click on the Database tab.



ID: 622400001

Name: SCTicket

Input Length: 16

Data Type: Character

Entry Mode: Optional

Search

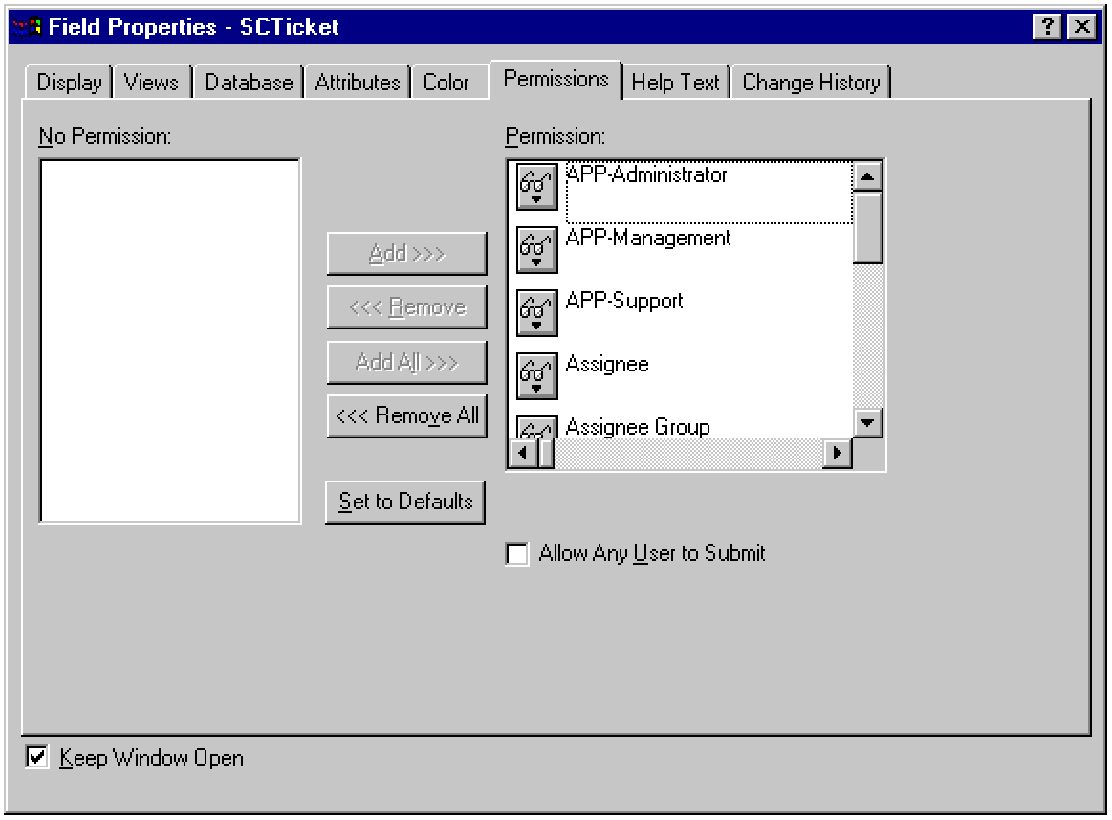
QBE Match: Anywhere

☑ Keep Window Open

Figure 5-4 Field Properties - Database

9. Enter an ID for the field. This can be any currently unused ID number.
10. Ensure that the Name is SCTicket. This is the actual name that this field will be referred by in the JScript later.
11. Change the Input Length to a number large enough to store the ServiceCenter ticket number.

12. Click on the Permissions tab.


Field Properties - SCTicket

No Permission:

Permission:

APP-Administrator
APP-Management
APP-Support
Assignee
Assignee Group

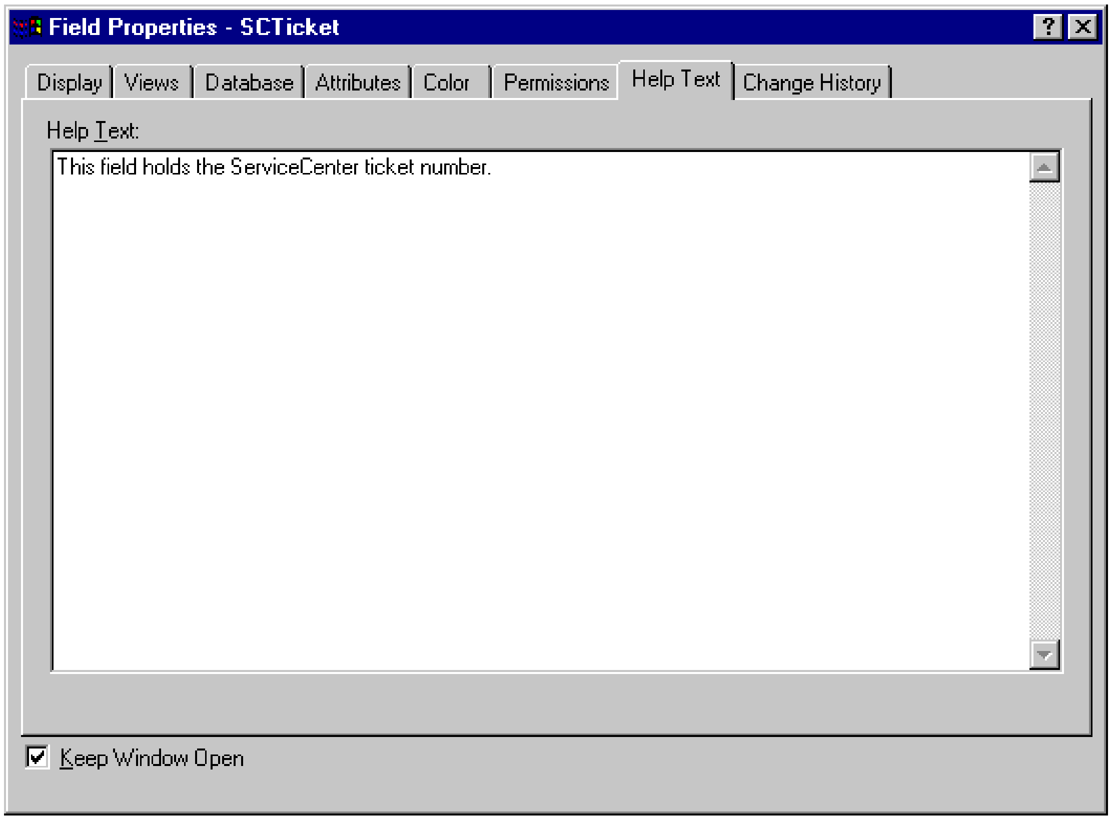
☐ Allow Any User to Submit

☑ Keep Window Open

Figure 5-5 Field Properties - Permission

13. Click on *Add All* to ensure that anyone that the system that adds the ticket can modify the SCTicket field. You may wish to modify these permissions to better suit your needs.

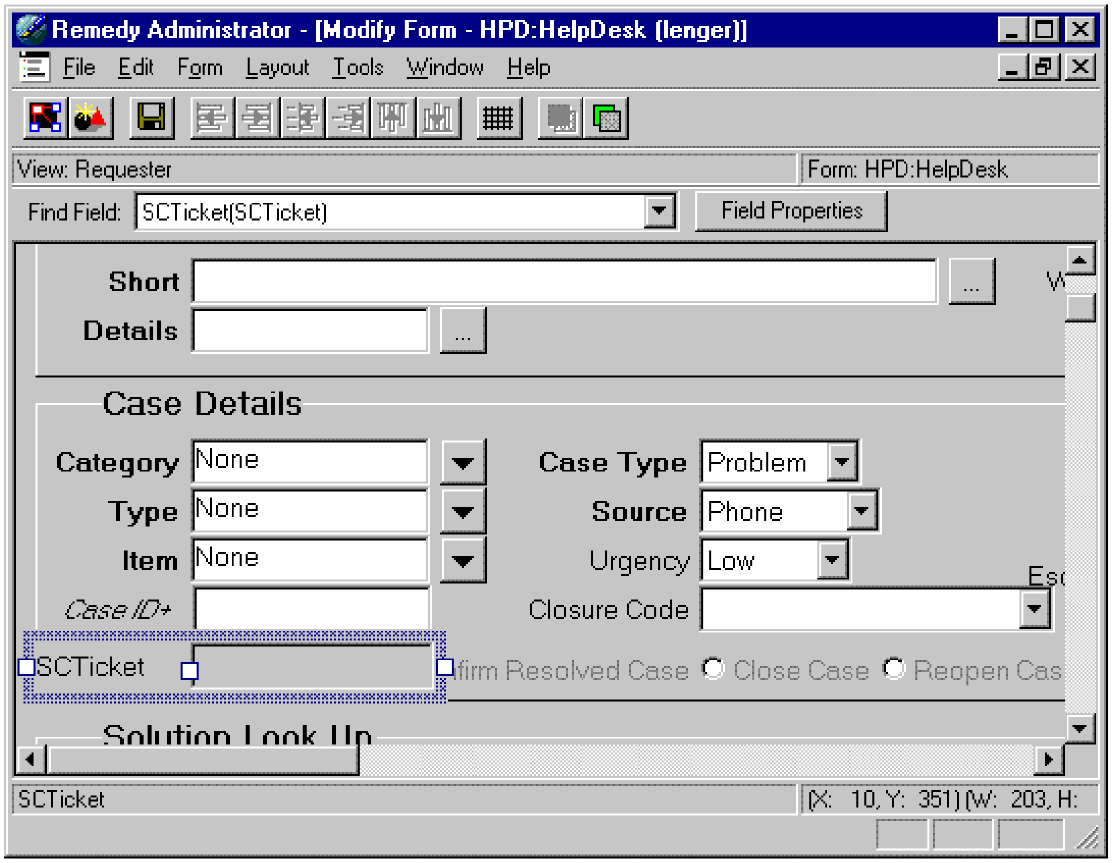
14. Click on the Help Text tab.



This field holds the ServiceCenter ticket number.

Figure 5-6 Field Properties - Help Text

15. Enter some help text as shown for the SCTicket field.
16. Click on the X (top corner of the screen) to close the Field Properties dialog when you are finished.


Remedy Administrator - [Modify Form - HPD:HelpDesk (lenger)]
File Edit Form Layout Tools Window Help
Field Properties
Solution Look Up

View: Requester
Form: HPD:HelpDesk
Find Field: SCTicket(SCTicket)

Short
Details

Case Details
Category None
Type None
Item None
Case ID+
SCTicket
Case Type Problem
Source Phone
Urgency Low
Closure Code
firm Resolved Case Close Case Reopen Cas

SCTicket
(X: 10, Y: 351) (W: 203, H:

Figure 5-7 Field Properties - SCTicket

17. Now you can adjust the size and position of your new SCTicket field until it is placed to your satisfaction.

SCNew

Once you have finished with the SCTicket field, you need to add an additional field that is used only for control by the SCAuto for Remedy ARS system. This field is SCNew. It keeps track of whether a ticket created on the Remedy ARS system is a new ticket.

To add the SCNew field:

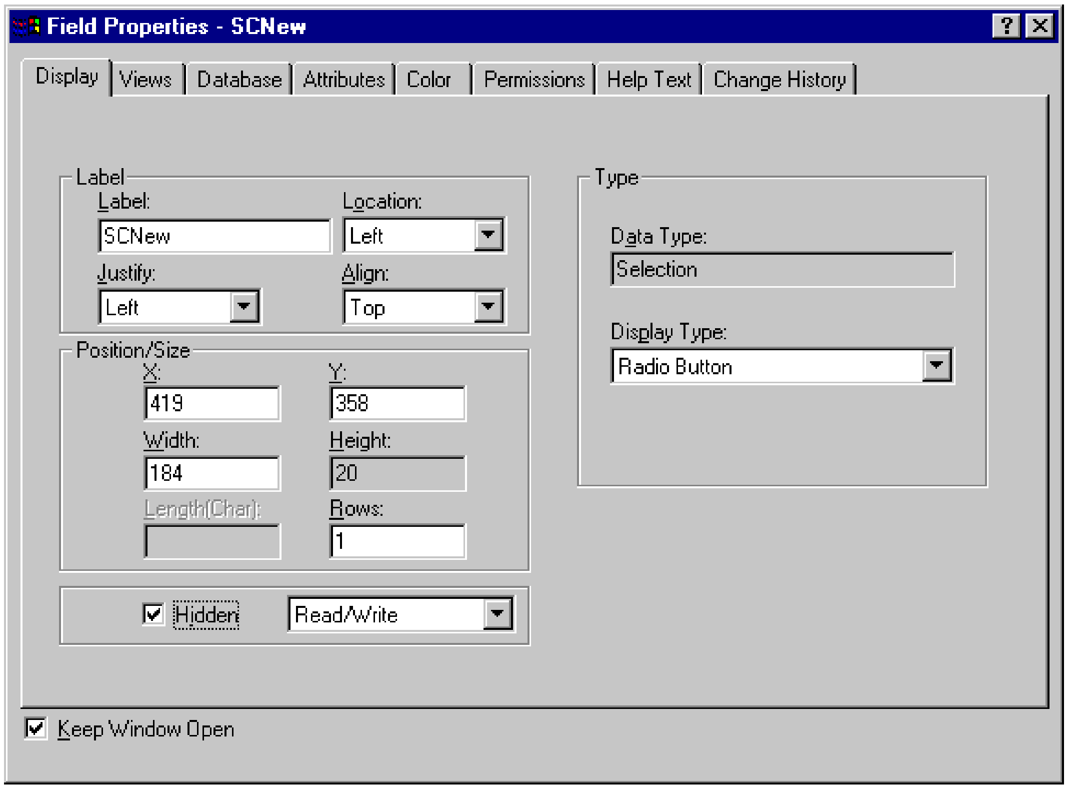
1. Click on **Form>Create a New>Radio Button Field**.

   A screen showing the radio button field is displayed.

2. Double-click on the newly created field to edit its properties.

   A screen is displayed with the field properties for SCNew.

3. Click on the Display tab.

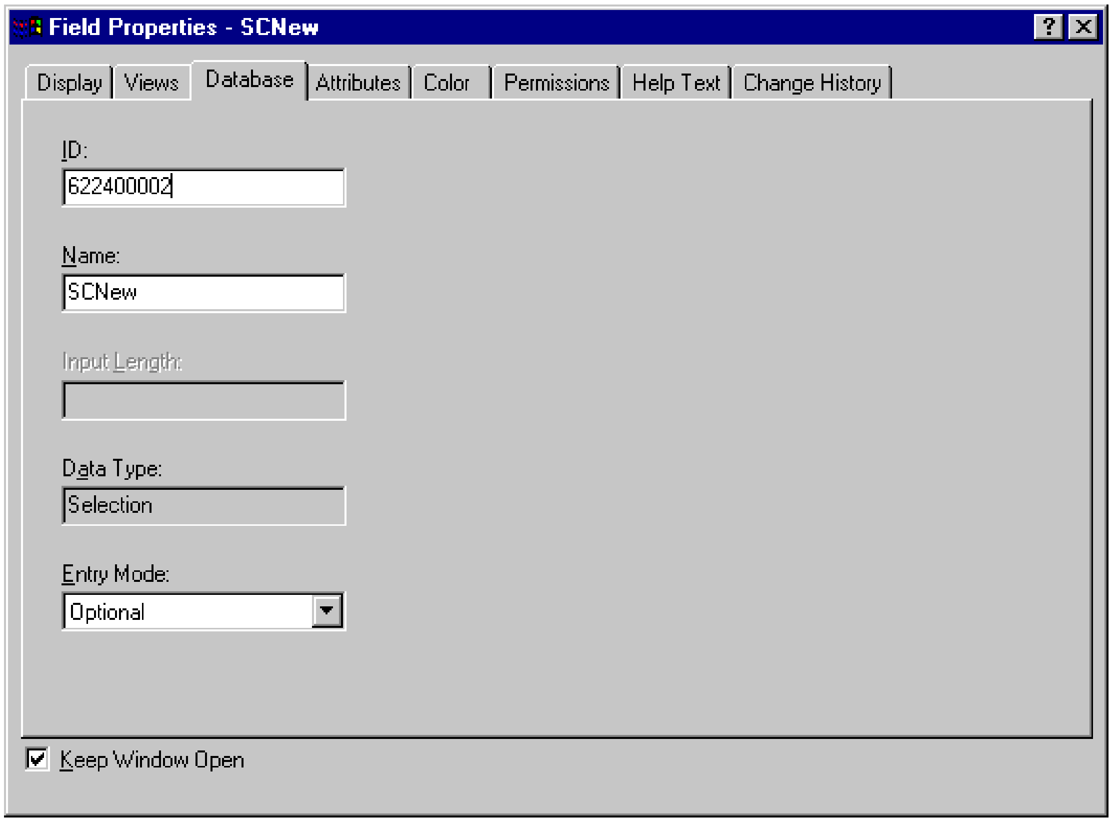


Label: SCNew
Location: Left
Justify: Left
Align: Top
Data Type: Selection
Display Type: Radio Button
X: 419
Y: 358
Width: 184
Height: 20
Length(Char):
Rows: 1
☑ Hidden
Read/Write
☑ Keep Window Open

Figure 5-8 Field Properties - Display

4. Modify the label to read SCNew.
5. Check the hidden box. This field will be hidden because it is only used for control purposes.

6. Click on the Database tab.



ID: 622400002

Name: SCNew

Input Length:

Data Type: Selection

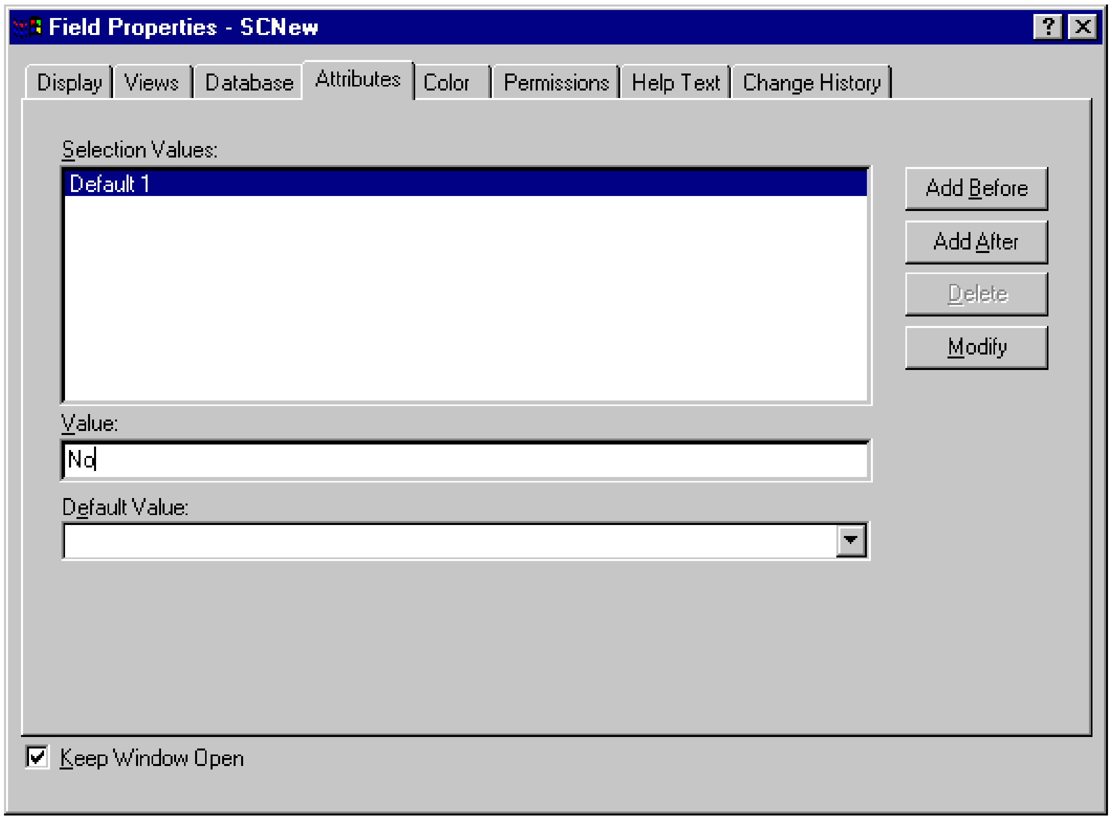
Entry Mode: Optional

☑ Keep Window Open

Figure 5-9 Field Properties - Database

7. Type an ID for the field and make certain it's name is SCNew. This is important since the database name is how SCAuto for ARS will later find this field. SCAuto for ARS does not find fields based on the label name—only the Database name.

8. Click on the Attributes tab. Here we must put in appropriate selection values for this field.



Field Properties - SCNew

Selection Values:
Default 1

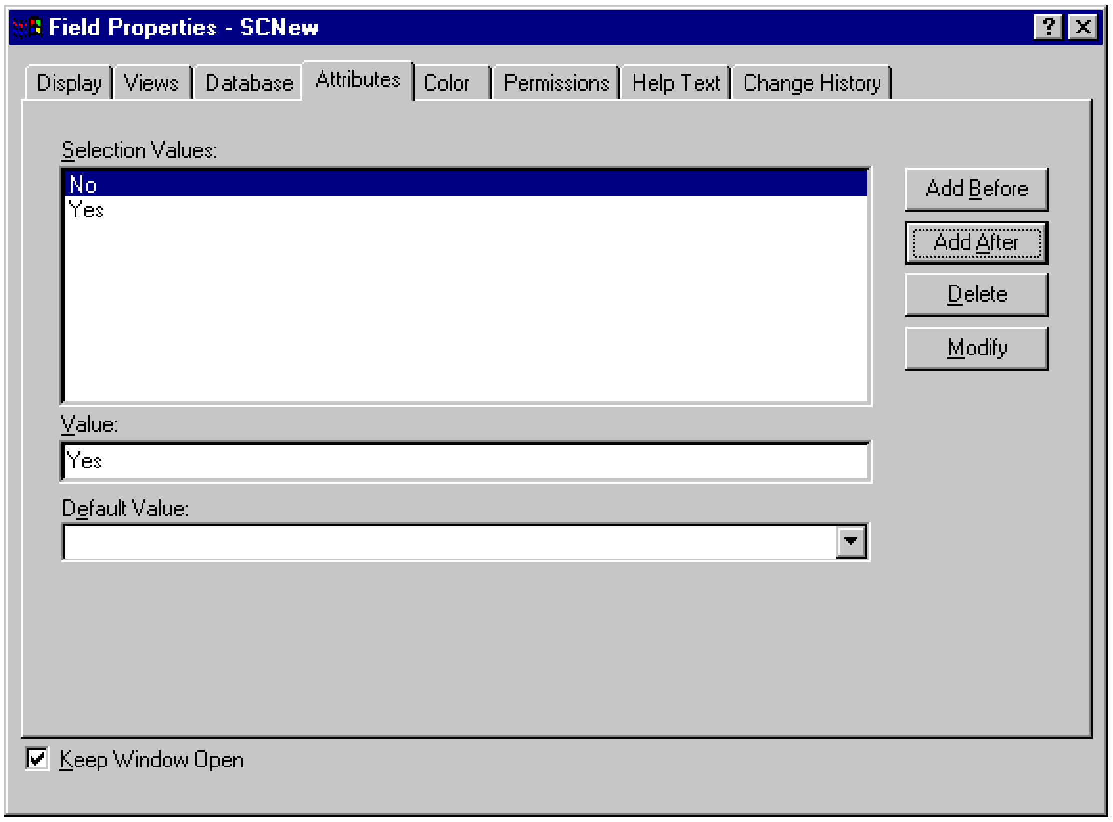
Value: No

Default Value:

Figure 5-10 Field Properties - Attributes

9. Type **No** in the Value field.
10. Click on Modify. This changes the original value of Default 1 to No.

The screen is updated.



Selection Values:
No
Yes

Value: Yes

Default Value:

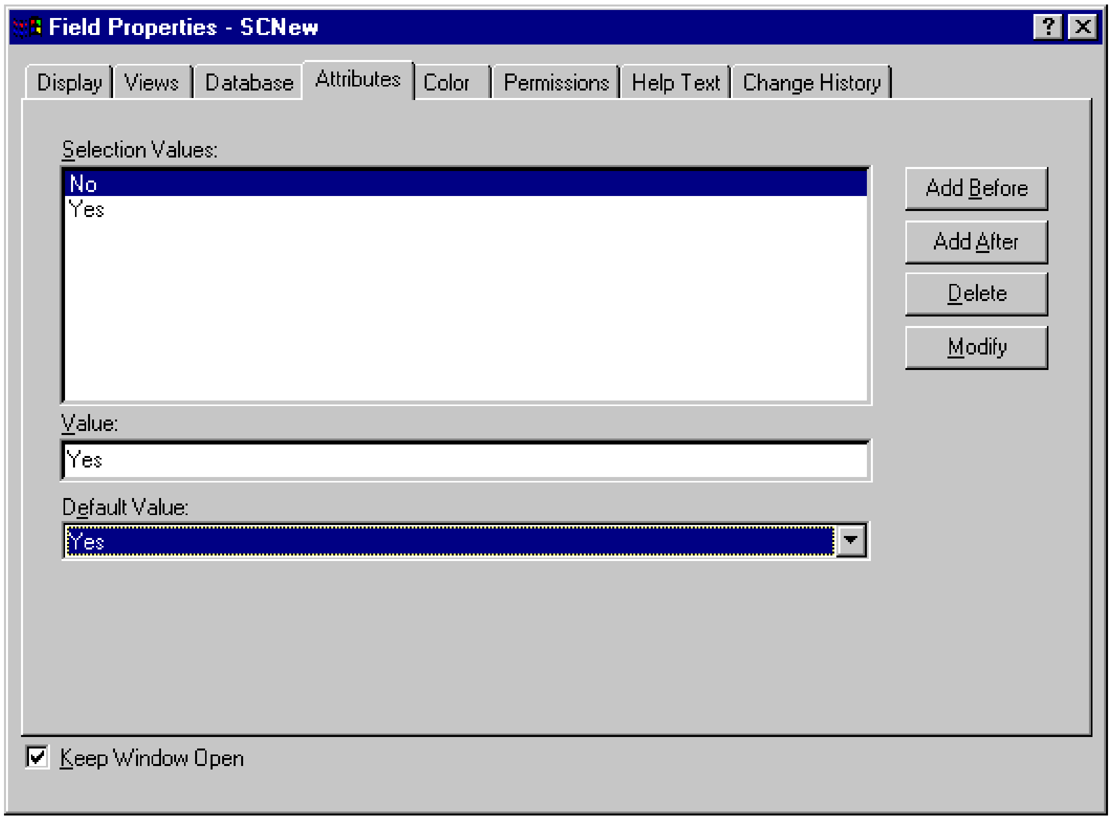
☑ Keep Window Open

Figure 5-11 Field Properties - Attributes

11. Type **Yes** in the value field.
12. Click on Add After.

    This will result in selections of No and Yes for this field. It is important that No be the first selection and Yes the second. This will give No the value of 0 and Yes the value of 1. Or for programming purposes, zero is False and any non-zero value is True. This will make the JScript much easier to deal with.

The screen is updated.


Field Properties - SCNew

Selection Values:
No
Yes

Value: Yes

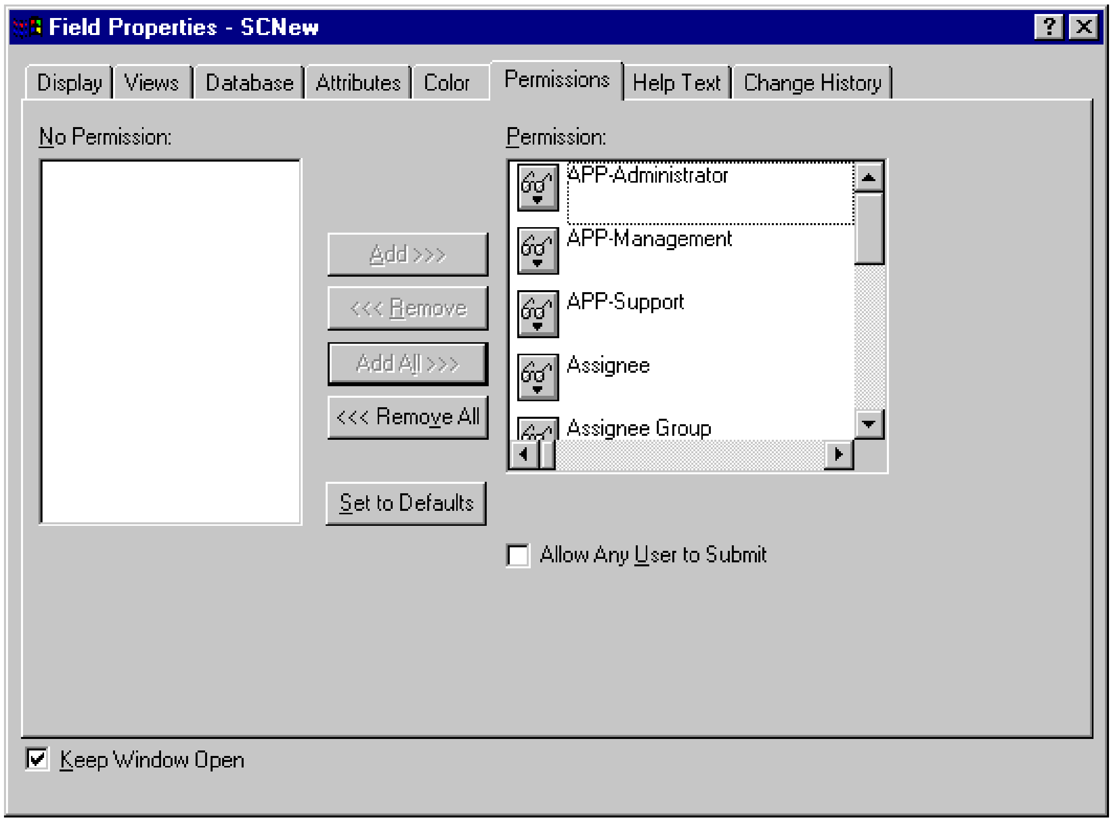
Default Value: Yes

☑ Keep Window Open

Figure 5-12 Field Properties - Attributes

13. Select a default value of Yes for this field. This ensures a newly created ticket is flagged as being a new ticket.

14. Click on the Permissions tab. Just as with the SCTicket field, you must give permissions so that SCAuto for Remedy ARS can modify this field.



Permission:
APP-Administrator
APP-Management
APP-Support
Assignee
Assignee Group

☐ Allow Any User to Submit

☑ Keep Window Open

Figure 5-13 Field Properties - Permissions

15. Click on Add All.
16. Give the appropriate permissions. You will not input any help text for this field as it will be hidden anyway.
17. Click the X (in the upper corner of the screen) when finished.
18. Place this field where appropriate on your form.

SCUpdated

Now you need to add an another field that is used only for control by the SCAuto for Remedy ARS system. This field is SCUpdated. It keeps track of whether a ticket has yet been updated on the ServiceCenter system after creation and modification.

To add the SCUpated field:

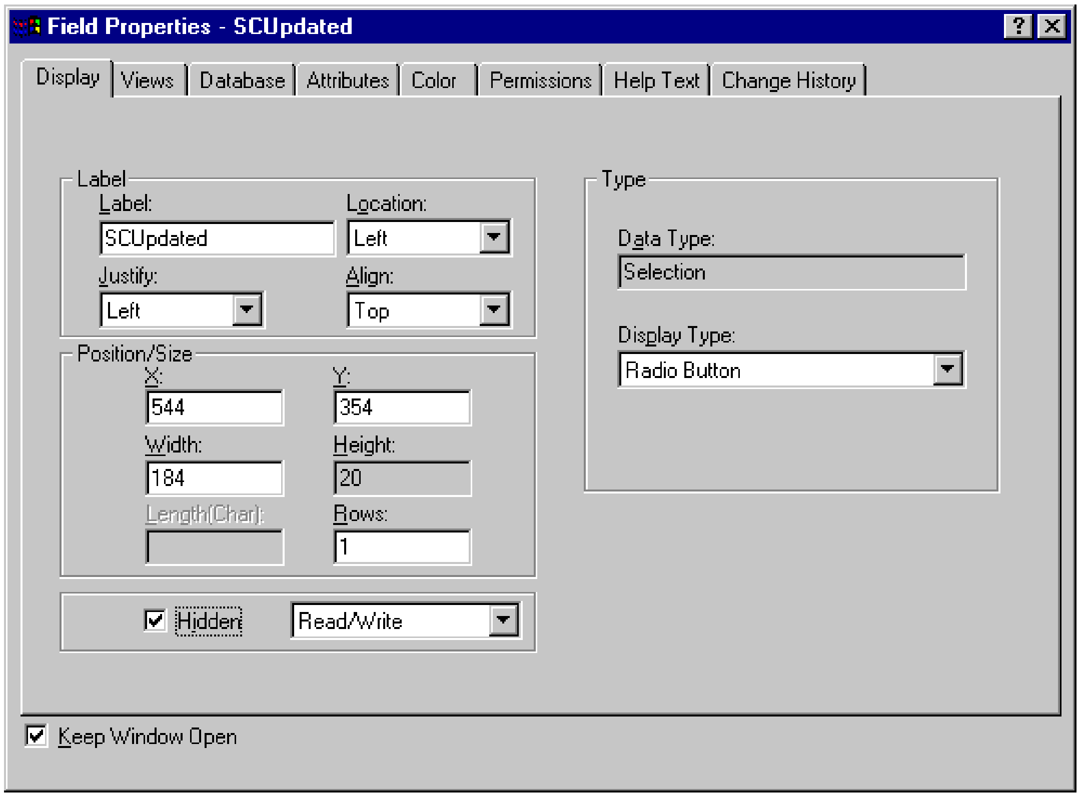
1. Click on **Form>Create a New>Radio Button Field**.

   A screen showing the radio button field is displayed.

2. Double-click on the newly created field to edit its properties.

   A screen is displayed with the field properties for SCUpdated.

3. Click on the Display tab.



Label
Label: SCUpdated
Location: Left
Justify: Left
Align: Top

Position/Size
X: 544
Y: 354
Width: 184
Height: 20
Length(Char):
Rows: 1

☑ Hidden
Read/Write

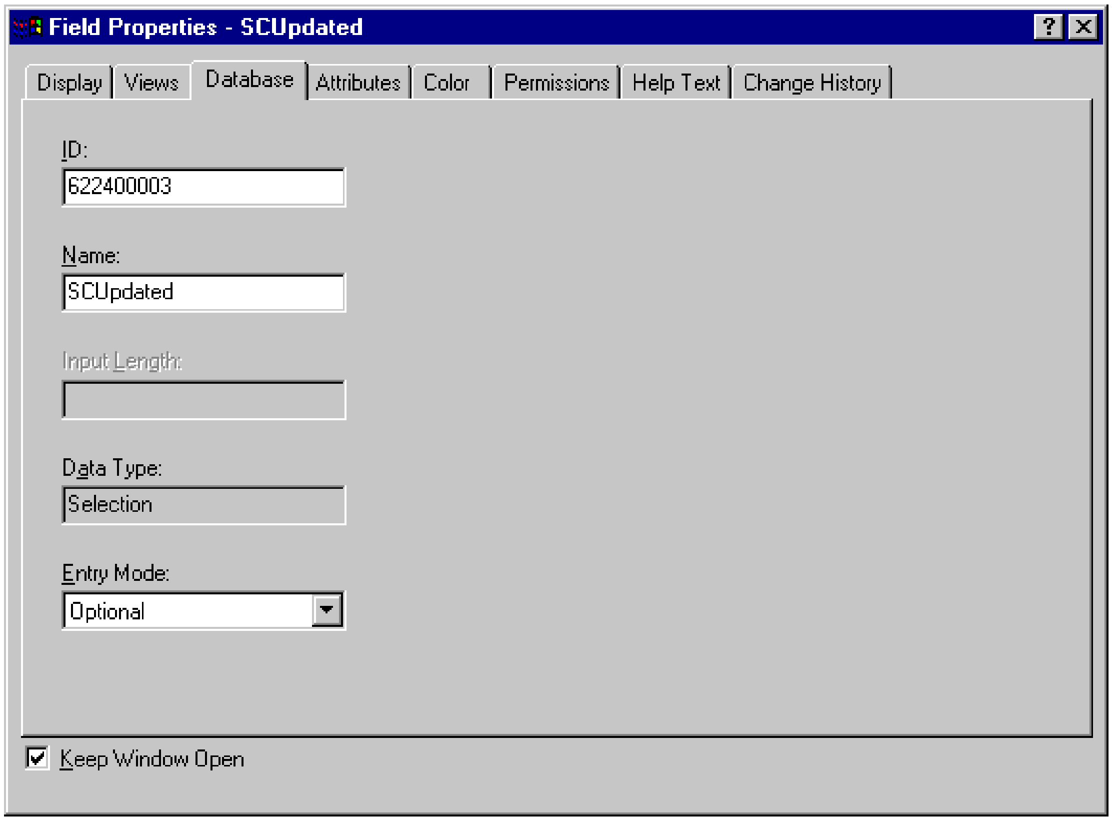
Type
Data Type: Selection
Display Type: Radio Button

☑ Keep Window Open

Figure 5-14 Field Properties - Display

4. Modify the label to read SCUpdated.
5. Check the hidden box. This field will be hidden because it is only used for control purposes.

6. Click on the Database tab.


Field Properties - SCUpdated

Display | Views | Database | Attributes | Color | Permissions | Help Text | Change History

ID: 622400003

Name: SCUpdated

Input Length:

Data Type: Selection

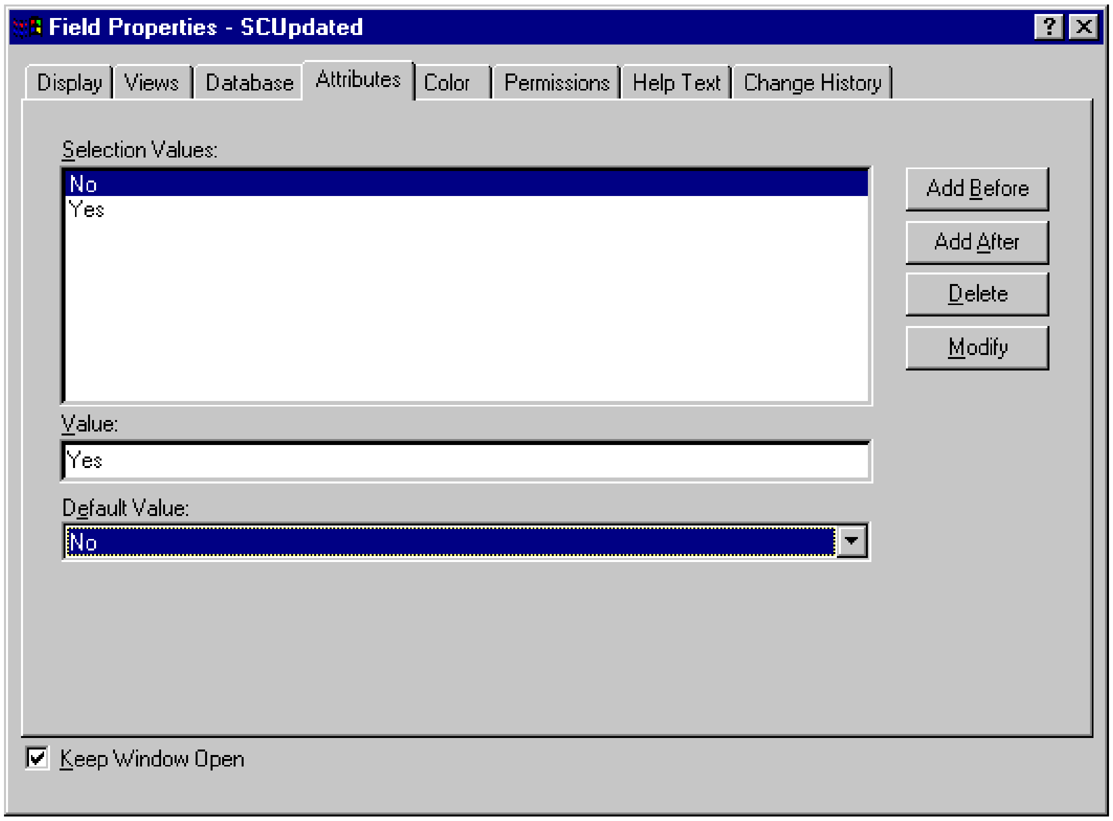
Entry Mode: Optional

[✓] Keep Window Open

Figure 5-15 Field Properties - Database

7. Type an ID for the field.
8. Type SCUpdated for the name. This name must be exact as SCAuto for Remedy ARS uses this field for control purposes.

9. Click on the Attributes tab. Here we must put in appropriate selection values for this field.



Selection Values:
No
Yes

Value: Yes

Default Value: No

☑ Keep Window Open

Figure 5-16 Field Properties - Attributes

10. Type **No** in the Value field.
11. Click on Modify.

    The screen is updated.
12. Type **Yes** in the value field.
13. Click on Add After.

    This will result in selections of No and Yes for this field. It is important that No be the first selection and Yes the second. This will give No the value of 0 and Yes the value of 1. Or for programming purposes, zero is False and any non-zero value is True.

The screen is updated.

14. Select a default value of No since this field indicates whether the ticket has been updated in ServiceCenter yet.
15. Click on the Permissions tab. You must give permissions so that SCAuto for Remedy ARS can modify this field.

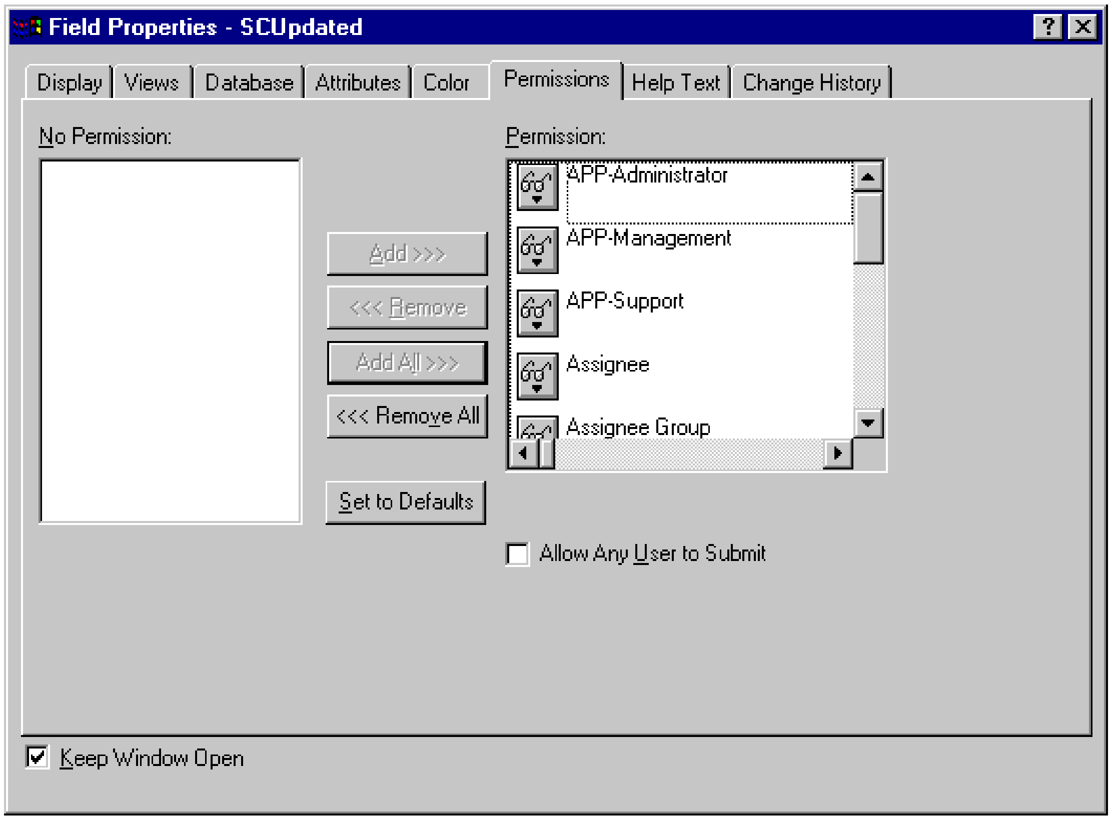


Figure 5-17 Field Properties - Permissions

16. Click on Add All.
17. Give the appropriate permissions. You will not input any help text for this field as it will be hidden anyway.
18. Click the X (in the upper corner of the screen) when finished.
19. Place this field where appropriate on your form.

Filters

The last modification you make to Remedy is adding a Filter to control the SCUpdated field.

To add a filter:

1. Return to the Server Window.

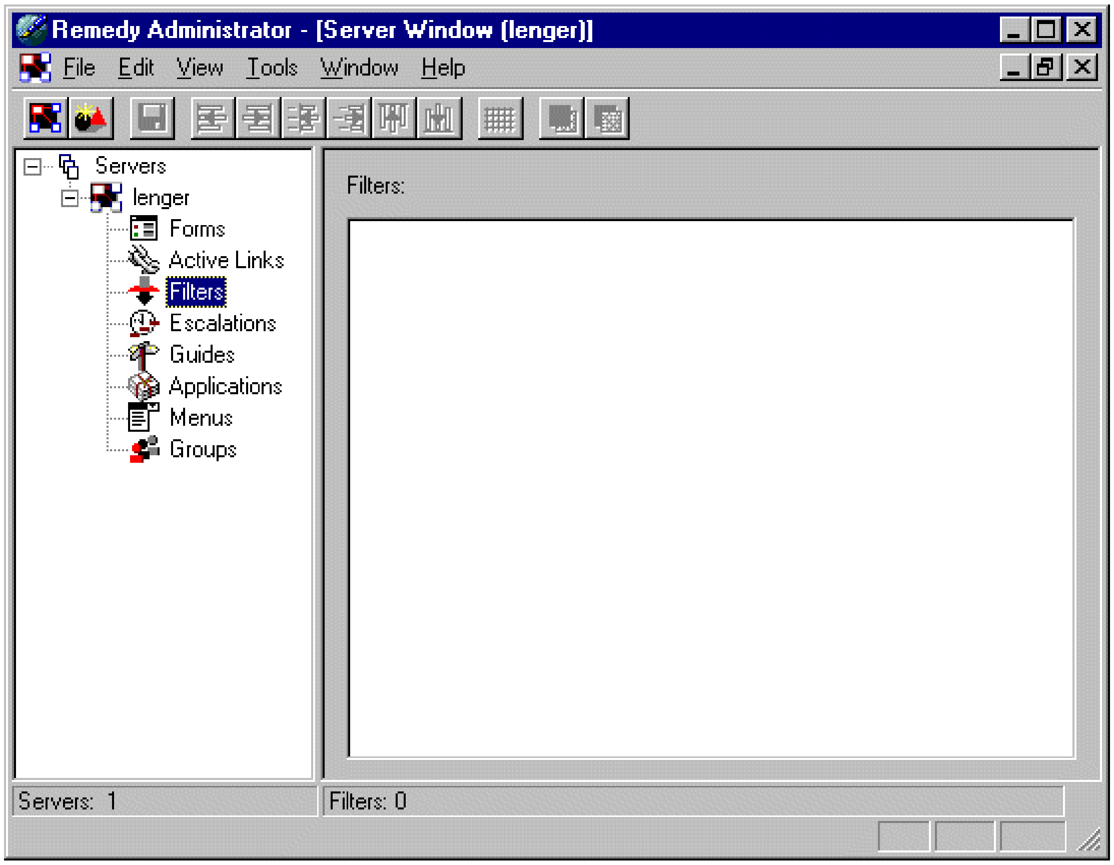
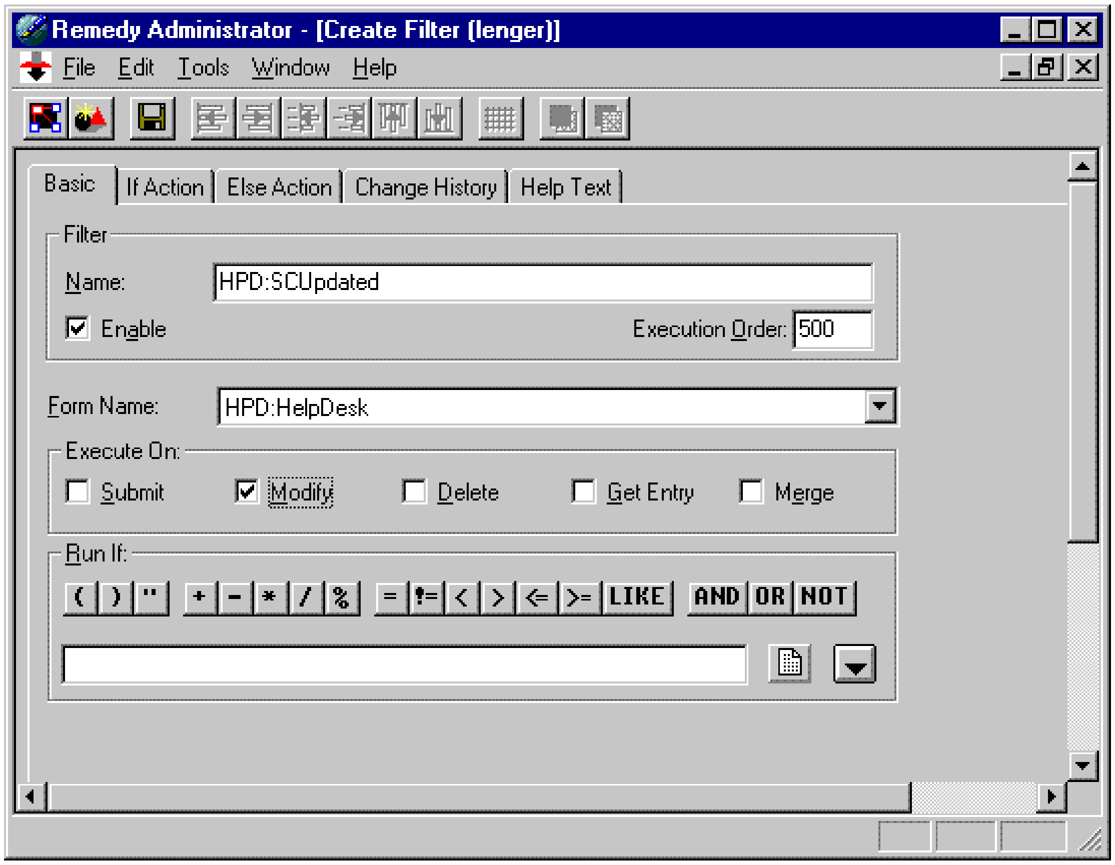


Figure 5-18 Field Properties - Server Window

2. Click on **File>New Server Object>Filter**.
3. Click on OK.

The Create Filter dialog box is displayed.

4. Click on the Basic tab.


Remedy Administrator - [Create Filter (lenger)]
File
Edit
Tools
Window
Help
Basic
If Action
Else Action
Change History
Help Text
Run If:
( ) " + - * / % = != < > <= >= LIKE AND OR NOT

Filter

Name: HPD:SCUpdated

☑ Enable Execution Order: 500

Form Name: HPD:HelpDesk

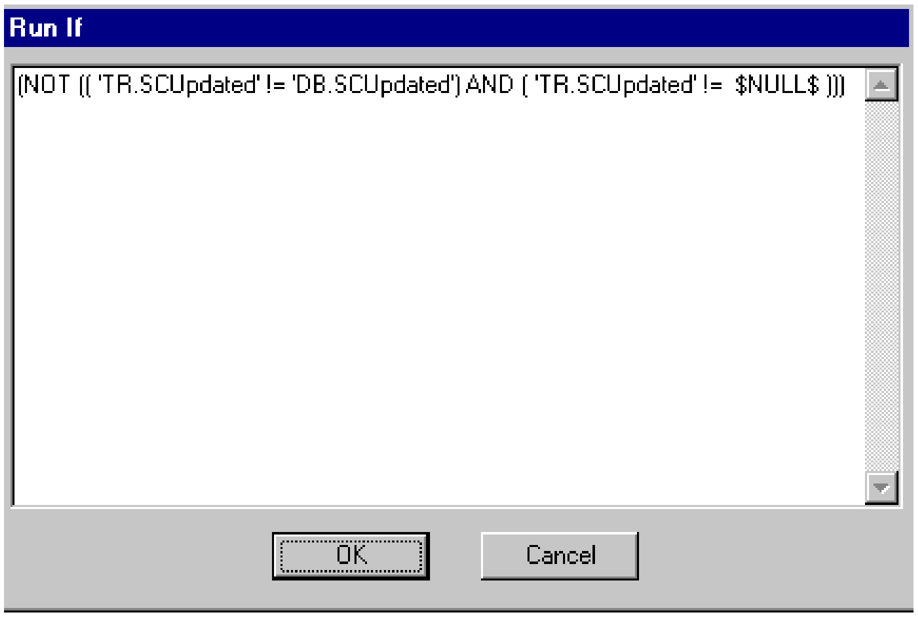
Execute On:

☐ Submit ☑ Modify ☐ Delete ☐ Get Entry ☐ Merge

Figure 5-19 Field Properties - Create Filter - Basic

5. Fill in the Name field with something that reflects both the name or your schema and the function of the field. For example, you may enter HPD:SCUpdated.
6. Select the appropriate form from the drop-down list.
7. Check the Modify box since this filter should run whenever a field is modified.

Now you have to add some conditional logic to the filter.

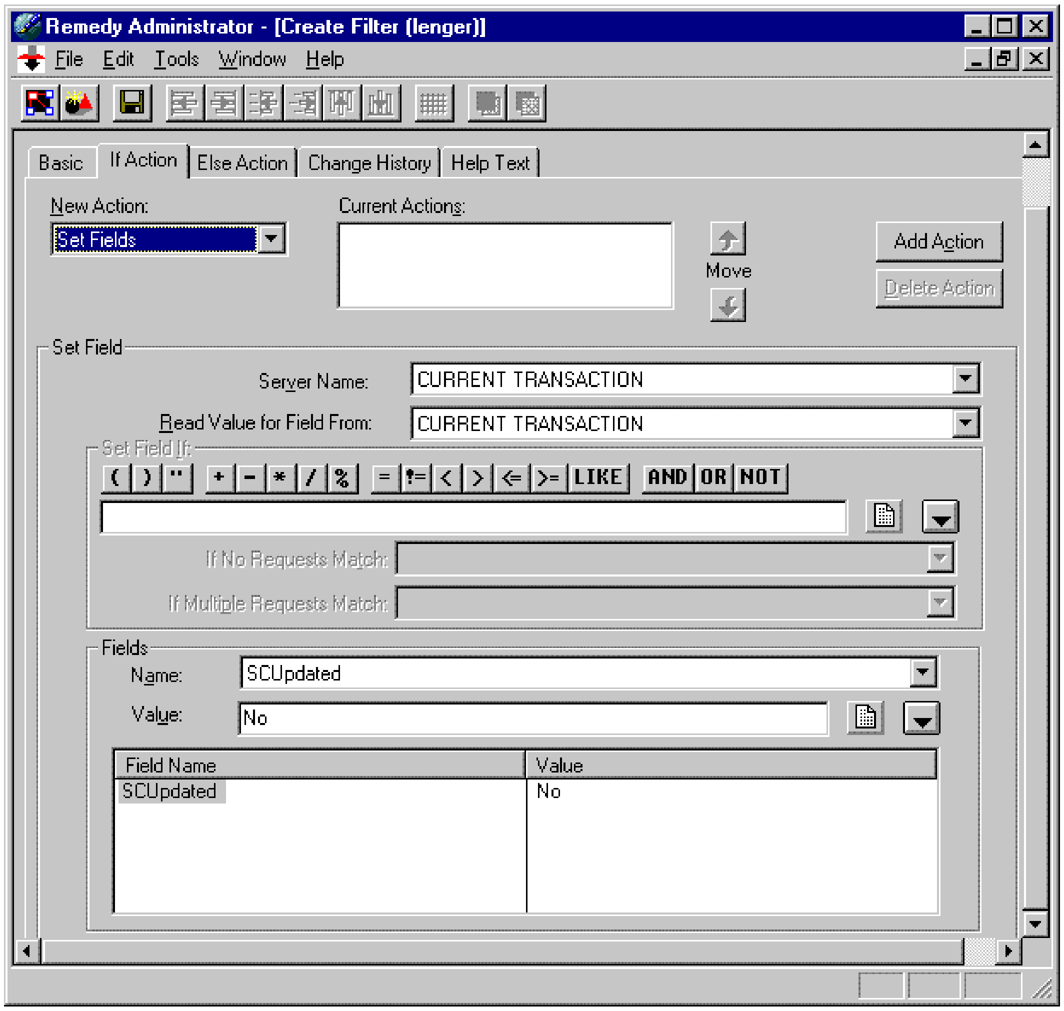


(NOT (('TR.SCUpdated' != 'DB.SCUpdated') AND ('TR.SCUpdated' != $NULL$)))

Figure 5-20 Field Properties - Create Filter - Run

8. Open the Run If dialog and input the statement shown in the graphic above. This ensures the filter will run only if some field other than the SCUpdated field is being modified.
9. Click on OK to close the dialog.

Click on the If Action tab.


Remedy Administrator - [Create Filter (lenger)]
File Edit Tools Window Help
Basic
If Action
Else Action
Change History
Help Text
Move
Add Action
Delete Action
Set Field
Set Field If:
( ) " + - * / % = != < > <= >= LIKE AND OR NOT
If No Requests Match:
If Multiple Requests Match:
Fields

New Action: Set Fields

Current Actions:

Server Name: CURRENT TRANSACTION

Read Value for Field From: CURRENT TRANSACTION

Name: SCUpdated

Value: No

Field Name	Value
SCUpdated	No

Figure 5-21 Field Properties - Create Filter - If Action

10. In the New Action field, select Set Fields from the drop down menu.
11. In the Name field in the Fields group, select the SCUpdated from the drop down menu.
12. In the Value field in the Field group, type No.
13. Select **File>Save Filter**.

This concludes the modifications to the Remedy ARS system.

Chapter 6 Scripting

Overview

This chapter explains the how Microsoft JScript 5.0 is used in the SCAuto for Remedy ARS gateway, and how other ActiveX scripting languages such as VBScript 5.0 may be used in addition to, or instead of JScript. It also discusses some of the logic of the provided script called **scars.js**, which is the main program of the gateway.

Windows Scripting Architecture and Terminology

The Microsoft architecture for ActiveX scripting languages in Windows is still fairly new, so it may help to briefly explain a few concepts and terminology.

Until very recently, the only native scripting language supported by the Windows operating system was the old MS-DOS® operating system command language used for batch files (.bat). This was a serious limitation of Windows. With the new Windows scripting architecture, users can take advantage of powerful scripting languages such as Visual Basic Scripting Edition and JScript. These scripting languages support object-oriented programming using OLE Automation objects (COM).

The Windows scripting architecture has three components:

ActiveX scripting hosts

These are programs such as the Microsoft-provided WSCRIPT.EXE and CSCRIPT.EXE. A scripting host is a program which can load an ActiveX scripting engine and a text file containing a script, and "feed" the script to the engine. A scripting host may also implement various objects and make these available to the script programmer.

ActiveX scripting engines

A scripting engine is a .DLL file which implements a particular scripting language. Microsoft currently provides two different scripting engines, one for JScript and another for VBScript. These are included in the Windows Script 5.0 distribution described in Chapter 2.

scripts written in one of the supported scripting languages

Using Windows scripting is as simple as writing a script in a supported language and invoking a scripting host such as wscript.exe to process the script. The script can use any OLE Automation objects which have been installed on the Windows platform where the script is running. You may be surprised at the number of such objects which can be found on any given Windows system.

Microsoft Scripting Engine Considerations

The SCAuto for Remedy ARS script **scars.js** uses Microsoft JScript 5.0, which is an ECMA262 compliant implementation of ECMA script.

Because it makes use of Microsoft extensions not yet in the ECMA specification, such as exception handling using **try...catch** notation, it is important to be sure that Microsoft JScript version 5.0 or later is installed on the NT platform where SCAuto ARS is running. Versions of JScript prior to 5.0 do not contain support for exception handling.

Although we chose to use JScript to implement SCAuto for Remedy ARS, the OLE Automation objects provided by Peregrine can be used just as easily from VBscript than from JScript. So if you have a major project to tackle, such as developing an additional script to integrate some other Remedy ARS application with ServiceCenter, you could do it in VBScript if you prefer.

Windows Scripting Host Considerations

No particular use is made of the special **wsh** objects or other Windows Scripting Host specific functionality in this release. **wscript.exe** is simply used to invoke and run the scars.js script. The command shell **cscript.exe** could just as easily be used instead.

The first release of SCAuto for Remedy ARS happens to use Windows Scripting Host 1.0, because WSH 2.0 was not yet generally available.

If you plan to do significant customization, you should investigate Windows Scripting Host 2.0, which supports multiple scripting languages in the same file, inclusion of scripts from other files, and other useful things.

Recommended Tools

There are a variety to tools available for free download from the Microsoft scripting web site referenced earlier in this book. These include a simple script debugger. However, it is recommended that you instead install the much more capable script debugger included in the Visual Studio Professional or Enterprise edition product.

No script customization should be attempted without first installing some sort of script debugger. If you don't have a version of Visual Studio which includes the enhanced script debugger, download the free script debugger from the Microsoft scripting web site.

TOC PREV NEXT INDEX

The SCARS.JS Script

This script functions as the main program of the SCAuto ARS gateway. It is executed by the command **wscript.exe scars.js** which appears in the scars.cfg file. The latter file is processed by the SCAuto ARS service control program, scarssrv.exe, at startup, and commands in the file are executed. You can make the gateway execute additional scripts by adding additional lines to the scars.cfg file to invoke wscript.exe for the new script.

The scars.js script is essentially an endless loop which runs until the SCAutoARS service is shut down, which causes scarssrv.exe to terminate the wscript process(es) that it started at startup. After some initialization code is executed to read parameters from scars.ini and create some OLE Automation objects, the script executes the two functions SC2ARS( ) and ARS2SC( ) continuously until service shutdown. The code below appears at the bottom of the scars.js file:

```
//=====================================================================
//
// Main program loop.
//
//=====================================================================

// Execute forever or until our service is shutdown.

while( true )
{
   SC2Ars( su );                  // Send a message from ServiceCenter to Remedy ARS .

   su.Sleep( ARSTimeout );        // Sleep.

   Ars2SC( su );                  // Send a message from Remedy ARS to ServiceCenter

   su.Sleep( ARSTimeout );        // Sleep.
}
```

The two functions SC2Ars and Ars2SC are basically exception-handling wrappers around two more detailed functions, *SC2ArsDetails* and *Ars2SCDetails.*

The mapping between the SC and Remedy objects occurs in the latter two functions—*SC2ArsDetails* and *Ars2SCDetails.*

Each of these functions has a marked section that indicates where the mapping of data is done. The other parts of the code are used for flow control and should normally not need to be modified.

The beginning of the sections for data mapping are marked with a comment block that looks like this:

```
////////////////////////////////////////////////////////////
//
// Do field mapping from ServiceCenter to Remedy in this section
//
////////////////////////////////////////////////////////////
```

The end of the sections for data mappings are marked with a comment block that looks like this:

```
////////////////////////////////////////////////////////////
//
// End of mapping from ServiceCenter to Remedy ARS
//
////////////////////////////////////////////////////////////
```

All mappings for data should occur between these two comment blocks and the two similar comment blocks in the other function.

Data may be assigned between objects using a simple syntax. The example shows how to map data from a Remedy ARS field to a ServiceCenter field. Specifically, the *remedy.no* field that we have added to ServiceCenter will be filled in with the Remedy ARS “Case ID” field.

```
s.EvField(“remedy.no”).Value = r.RField(“Case ID”).Value;
```

This next example demonstrates how to check to see if a field is filled in. If it is, we will simply map the data as demonstrated above, otherwise we will put in a default value. This is particularly useful when a field is required in one system but not in the other.

```
if ( r.RFields( "Short Description" ).Value )
{
    r.RFields( "Short Description" ).Value = s.EvFields( "brief.description" ).Value;
}
else
{
    r.RFields( "Short Description" ).Value = "No brief description provided";
}
```

This example demonstrates the use of a switch statement to perform different data mappings depending on the event type. s.EvType is a special field that holds the ServiceCenter event type. Note that in the case of *pmu* and *pmc*, the data from multiple fields are being appended into one. Switch statements can also provide for a default values through the use of the default case.

```
switch( s.EvType )
{
case "pmo":
    r.RFields( "Details" ).Value = s.EvFields( "action" ).Value;
    break;
case "pmu":
    r.RFields( "Details" ).Value = s.EvFields( "action" ).Value;
    r.RFields( "Details" ).Value += "\n";
    r.RFields( "Details" ).Value += s.EvFields( "update.action" ).Value;
    break;
case "pmc":
    r.RFields( "Details" ).Value = s.EvFields( "action" ).Value;
    r.RFields( "Details" ).Value += "\n";
    r.RFields( "Details" ).Value += s.EvFields( "resolution" ).Value;
    break;
default:
    su.LogMsg( "Invalid EvType!" );
    break;
}
```

Notice the above example, that fields can be appended together through the use of the += operator.

The following example demonstrates how to switch from a ServiceCenter character field to a related Remedy ARS selection field. This is important when the data types in each help desk are different. Please note that we are again providing a final, default value through the use of the final else.

```
if ( s.EvFields( "severity.code" ).Value == "Low" )
{
    r.RFields( "Request Urgency" ).Value = 0;
}
else if ( s.EvFields( "severity.code" ).Value == "Medium" )
{
    r.RFields( "Request Urgency" ).Value = 1;
}
else if ( s.EvFields( "severity.code" ).Value == "High" )
{
    r.RFields( "Request Urgency" ).Value = 2;
}
else if ( s.EvFields( "severity.code" ).Value == "Urgent" )
{
    r.RFields( "Request Urgency" ).Value = 3;
}
else
{
    r.RFields( "Request Urgency" ).Value = 3;
}
```

In this next example, we check more than one condition through the use of the && and operator before deciding whether to fill in a field or not. This if statement first evaluates whether the field “SCTicket” is filled in, and if it is, checks to see if the EvType field is “pmo” for problem open. If both conditions are met, then the field is mapped.

```
if ( r.RFields( "SCTicket" ).Value && s.EvType != "pmo"  )
{
    s.EvFields( "number" ).Value = r.RFields( "SCTicket" ).Value;
}
```

Note: Not all the fields in both products are mapped in the default script. Further modifications are required to either or both systems to more closely match their respective fields so they can be properly mapped.

Chapter 7 Operating SCAuto for Remedy ARS

Overview

This chapter describes the procedures for operating SCAuto for Remedy ARS.

Starting SCAuto for Remedy ARS

There are three ways to start SCAuto for Remedy ARS:

1. Select **Start->Programs->SCAuto for Remedy ARS->Start SCAuto for Remedy ARS** from the menu.

   This executes the command **scarssrv -start** from the **\SCAuto for Remedy ARS\Bin** directory.

2. You can also issue the following command from a Command window (the current directory in the command window is unimportant):

   net start scautoars

3. You can also execute the Control panel services applet to start SCAuto for Remedy ARS.

All three of these methods cause the same code, and thus are completely identical in function.

Starting the **SCAUTOARS** service normally takes from two to five seconds. If the service is unable to start for any reason, a message will be logged in the Windows NT system log, and detailed error messages will appear in the *scars.log* file.

What happens when SCAuto for Remedy ARS is Started

When SCAuto for Remedy ARS is started, the main service executable ***scarssrv.exe*** is invoked. It then locates the file *scars.cfg* which is found in the **\SCAuto for Remedy ARS** directory. If this file cannot be found, the service will not start.

Next, the *scars.cfg* file is processed. Each line in the file which is not blank or commented is interpreted as a process to run. Processes are typically either transient commands, i.e., invocations of *cmd.exe*, or one of the following background processes installed with SCAuto for Remedy ARS:

scevmon.exe

This process makes a TCP connection with the SCAUTOD component of ServiceCenter, using the hostname and port number information supplied to the configuration dialog program, *scarscfg.exe*. Whenever a connection is established, scevmon attempts to exchange events with ServiceCenter. **scevmon.exe** manages two local queues of events: an inbound events queue and an outbound events queue. These queues are contained in the SCAuto for Remedy ARS installation directory.

The name of the outbound queue is **scevents.to<host>.<port number>**. Outgoing events are logged to the *scevents.to.* file by processes such as the transient message action command **scevent.exe** or the long-running background process and consumed by scevmon.exe, which sends them to the ServiceCenter SCAUTOD process indicated in the file name of the **scevents.to** queue. When SCAUTOD accepts the events, they are written to the ServiceCenter EVENTIN file.

The name of the inbound events queue is **scevents.from<host>.<port number>**. This queue is created by **scevmon.exe** from events retrieved from the ServiceCenter EVENTOUT file via SCAUTOD. **scevmon.exe** determines its starting position in the EVENTOUT file of ServiceCenter by consulting a file called *syncfile.<host>.<service>* where *<host>* and *<service>* are the hostname and TCP port or service name of the SCAUTOD server. It then requests from SCAUTOD all events in EVENTOUT with sequence numbers greater than the value contained in the *syncfile*. The *syncfile* is updated as each event is received from SCAUTOD and successfully written to the *scevents.from* queue.

Stopping SCAuto for Remedy ARS

There are three ways to stop SCAuto for Remedy ARS:

1. Select ***Start/Programs/SCAuto for Remedy ARS/Stop SCAuto for Remedy ARS*** from the menu.

   This executes the command **scarssrv -stop** from the **\SCAuto for Remedy ARS\Bin** directory.

2. You can also issue the following command from a Command window. The current directory in the command window is unimportant:

   net stop scautoars

3. You can also execute the **Control Panel Services** applet to stop SCAuto for Remedy ARS.

All three of these methods execute the same code, and thus are completely identical in function.

Stopping the SCAUTOARS service normally takes five to ten seconds, except when CPU monitoring is in effect. If the service was instructed to monitor the CPU utilization via the optional **monitorcpu:1** parameter in the *scars.ini* file, stopping the SCAutoARS service may take up to twenty seconds.

What happens when SCAuto for Remedy ARS is Stopped

Stopping the SCAUTOARS service signals a Windows NT event called **SCAutoARS.StopEvent**. This causes the various processes that were started by the service to immediately go into termination logic, where they release whatever resources they acquired from Windows NT and then exit. Should any process fail to exit voluntarily for any reason, the main service executable **scarssrv.exe** will terminate it using the Windows NT *Terminate Process* command. For this reason, the service should not normally fail to stop. In this unlikely event, detailed error messages should appear in the *scars.log* file.

Determining if Remedy ARS is Running Normally

You can use the Windows NT Resource Kit command **SC** (Service Control) to query the SCAUTOARS service as follows:

sc query scautoars

This results in output similar to the following:

```
N:\tmp>sc query scautoars
SERVICE_NAME: scautoars
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 4  RUNNING
```

An alternative way to determine if SCAuto for Remedy ARS is running is to use the Task Manager. If the service is running, the **scarssrv.exe** process is visible in the task list displayed by Task Manager.

Day to Day Administration

SCAuto for Remedy ARS is designed for minimal administration overhead and maximum availability and operational flexibility.

SCAuto for Remedy ARS can tolerate indefinite outages of Remedy ARS and ServiceCenter. The operation of the latter products does NOT have to be synchronized with SCAuto for Remedy ARS in any way; i.e., SCAuto for Remedy ARS does NOT have to be stopped if ServiceCenter is being stopped or paused, nor is it necessary for those products to be running in order to start SCAuto for Remedy ARS.

SCAuto for Remedy ARS does not maintain a database. It produces a log file, but it automatically wraps this log when it reaches a defined maximum size. The default maximum log file size is 5 MB. This value is controlled by a setting in the *scars.ini* file.

You should never need, therefore, to stop SCAuto for Remedy ARS.

SCAuto for Remedy ARS can be started and stopped at any time. However, long periods of Remedy ARS operation without running SCAuto for Remedy ARS should be avoided, as this will result in out-of-date information in the ServiceCenter database.

Similarly, long periods of Remedy ARS and SCAuto for Remedy ARS operation without running ServiceCenter should be avoided, as large numbers of events may accumulate in the **scevents.to...** file awaiting transmission to ServiceCenter.

Appendix A SCARS.INI File Parameters

Overview

This appendix documents special parameters that can be coded either in *scars.ini* or on the command line of a given executable launched via the *scars.cfg* file. A setting on the command line of a given executable in the *scars.cfg* file overrides any global setting specified in the *scars.ini* file.

The appropriate values are automatically set when the product is installed. In normal operation, you should not have to change these parameters.

Important: In general, parameter settings should not be changed unless suggested by Peregrine support personnel. Experimentation with these values without consultation with customer support could lead to unexpected results, or impair the operation of SCAuto for Remedy ARS.

Logging Parameters

-log:

Controls where the log file is created. -log: is followed by a path specification, normally `scars.log`, which causes the log file to be called *scars.log* and written in the directory where the product is installed, which is normally the \Program Files\Peregrine Systems\SCAuto for Remedy ARS\ directory.

-logappend:1

A required parameter when multiple processes and threads share a common log, which is standard with SCAuto for Remedy ARS. This setting causes each process to append its messages to a common log file, rather than start a new log file.

-logmaxlen

Allows you to specify, in bytes, the maximum log file size. When the log reaches this size, SCAuto for Remedy ARS will begin writing log information at the beginning of the file again.

-logpreservelen

Allows you to specify, in bytes, the amount of log file text which is to be preserved when the log wraps. This feature permits the log to wrap without losing all immediately preceding messages. The last logpreservelen bytes in the log are saved when the log wraps, and are written to the beginning of the new log.

CPU Monitoring Parameters

SCAuto for Remedy ARS has the capability of monitoring the CPU utilization of its processes, and shutting itself down if specified values are exceeded. This feature is disabled by default. If enabled, care should be taken not to set the maximum CPU utilization values too low, or SCAuto for Remedy ARS will needlessly terminate one or more of its sub-processes. If this happens more than *retrymax* times, the service will shut itself down. Use of this feature is only advised if a problem is experienced with a runaway (looping) SCAuto for Remedy ARS process.

-monitorcpu:1

Enables CPU monitoring. Every 10 seconds, a snapshot of Windows NT performance counters is taken, and the percentage utilization of each SCAuto for Remedy ARS process is examined.

-maxprocesscpu

Allows you to specify a value from 0.00 to 1.00 which is the maximum allowable CPU utilization for a single SCAuto process within the monitoring interval 0.99 would be 99%.

-maxtotalcpu

Allows you to specify a value from 0.00 to 1.00 which is the maximum allowable CPU utilization for all SCAuto processes within the monitoring interval. 0.99 would be 99%.

Event Logging Parameters

-sceventserver

Set by the configuration utility, and consists of the hostname or IP address of the ServiceCenter server and the SCAUTO port number or service name. These two values are separated by a period, for example - *sceventserver:helpdesk.12673*, and are used by the SCAUTO.DLL to connect to the SCAUTOD process running on the ServiceCenter server.

scevmon_sleep_interval

Specifies the length of time, in seconds, that the scevmon.exe process should sleep between polls of the ServiceCenter EVENTOUT queue to look for outbound ServiceCenter events. The default value if not specified is 5 seconds. The delivered value in the scars.ini file is 1 second.

scevents

Specifies an optional list of event types which are to be retrieved from ServiceCenter's EVENTOUT queue. The default is to retrieve all types, however, this can be inefficient, because it can result in bringing over certain types of events, such as outbound page messages or email messages, which have no meaning to Unicenter. To restrict the types of events that are retrieved, code them in a list separated by commas and enclosed in parentheses. For example,

```
scevents:(pmo,pmu,pmc)
```

To totally disable retrieval of outbound events from ServiceCenter, use the "noeventsfromsc:" option.

noeventsfromsc

Specifies whether or not scevmon.exe should retrieve events from ServiceCenter's EVENTOUT queue. The default is 0 (which means events will be retrieved). To disable retrieval of all outbound events from ServiceCenter, code "noeventsfromsc:1"

-eventlogmaxlen

Specifies the maximum length of the scevents.to.... or scevents.from... event logs, before **scauto.dll** will attempt to purge them. The purge process is automatic, and involves rebuilding the affected file, while removing processed events. For example, purging the scevents.to... file involves removing events which have been successfully transmitted to ServiceCenter. Purging the scevents.from... file involves removing events from ServiceCenter which have been successfully processed by scevmon. It is possible for a purge attempt to cause little or no reduction in the size of the event log; for example if few or none of the events qualifies for purging because ServiceCenter has been unavailable and the “to” log is full of unsent events. If a purge attempt fails, the new event that was to have been appended to the log is discarded, and a message to that effect is written to the scars.log file. If this happens, investigate why events have not been being processed successfully. You can shut down the service and delete the scevents.from and scevents.to log files, along with the syncfile file without harming SCAuto for Remedy ARS, if necessary.

Remedy ARS Parameters

ars_server:

The *ars_server* parameter defines the TCP/IP hostname of the Remedy ARS server.

ars_username:

The *ars_username* parameter defines the user id for SCAuto for Remedy ARS to use when logging in to Remedy ARS.

ars_password:

The *ars_password* parameter defines the password for SCAuto for Remedy ARS to use when logging in to Remedy ARS.

ars_schema:

The *ars_schema* parameter defines the ARS schema to be used for updating the Remedy ARS database. This is a required parameter. All interaction with the Remedy ARS database is performed in terms of a particular schema. The default schema delivered with the SCAuto for Remedy ARS gateway is for the standard Remedy Help Desk application without customization applied.

Miscellaneous Parameters

-killableprocesses:

-killableprocesses:**1** causes the processes which are started by SCAuto for Remedy ARS to be killable by anyone using Task Manager or a similar utility. By default, the processes started by SCAuto for Remedy ARS are created with NULL Windows NT security parameters, such that they are not killable via the Task Manager or *pview.exe,* or similar utilities. They can only be killed by shutting down the service, or by a program which acquires the special Windows NT debug privilege.

Debugging Parameters

Debugging parameters all begin with -*debug* and are given a value of **1** or **0** to turn them on or off. All debugging parameters are **0** by default. Debugging parameters are not intended for everyday use, but only for investigating why something is or is not happening as expected. For example, if events are not being generated as expected, **debugscautoevents:1** might be coded.

These settings should generally only be used if requested by customer support. The output of these commands is not designed to be intelligible to the customer, but rather to the developers.

-debug:1

Causes the process to intercept **CTRL-C** and interpret this as a shutdown event. Used in conjunction with -event:0. Not for production use.

-debugall:1

The equivalent of coding **:1** for all other available **-**debug options. When **-debugall:1** is present in *scars.ini*, enormous numbers of messages are written to scars.log by all processes. This setting should not be used in normal operation, as it will slow down the product and cause the log to wrap frequently.

-debugcpu:1

Causes logging of CPU utilization monitoring.

-debugscautoevents:1

Causes information regarding construction and transmittal of events to be logged.

Index

Peregrine
S Y S T E M S®

The Infrastructure Management Company™