ServiceCenter™

SCAuto Applications for Windows NT and UNIX

Version 4.0

October 2001

SCA-1.1-ENG-01001-00090

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

Peregrine
S Y S T E M S®
The Infrastructure Management Company™

This edition applies to version 4.0 of **SCAuto Applications for Windows NT and UNIX**.

Contents

Chapter 4 SCAuto Mail for UNIX

Chapter 5 Customizing SCAuto Mail

Chapter 1 Introduction

Overview

Welcome to Peregrine Systems' *SCAuto Applications for Windows NT and UNIX*. This product is part of the suite of SCAuto (SCAutomate) interface products that integrate ServiceCenter with premier network and systems management tools.

Prerequisite knowledge

This guide assumes you have:

- Working knowledge of ServiceCenter applications, ServiceCenter Client/ Server, and Windows NT and UNIX operating systems. While some procedures for these applications are explained, others are referenced. Refer to the appropriate ServiceCenter documentation for a more detailed explanation.
- Working knowledge of a GUI or text-based environment.
- (As an Administrator) a thorough knowledge of the operating system where the product is installed and implemented, as well as basic understanding of ServiceCenter applications and Event Services.

SCAutomate

ServiceCenter Automate (SCAuto), the successor to AXCES, provides event management services through a collection of automation products which enable external applications to be integrated with ServiceCenter. These generic services allow customers to create meaningful events from their environment and applications. Some common applications of this functionality include opening problem tickets, notification upon the closure of a ticket, automatic network inventory management and e-mail management between disparate external mail programs. The intent is to enable communication from any external application with any ServiceCenter RAD application, without predetermining what that communication requires.

Base server: *scautod*

The SCAuto Base server program is called **scautod** (or *scautod.exe* on Windows NT). This may be started from the command line, or from within ServiceCenter. When it has started and is running, it accepts connections from SCAuto clients (SCAuto/Notes, SCAuto/Spectrum, etc.).

Note: While running, it will appear with the user name **SCAuto Server** from within ServiceCenter, and should be stopped from there if needed. Existing clients will not be terminated when **scautod** is stopped.

The file *scautod.msg* should be present in the same directory as the **scautod** program. This file contains diagnostic messages that scautod uses, and may be customized for local language or usage.

Scautod is a modified version of the **scenter** program, and thus will use either the *sc.ini* file or the command line to find its parameters. The necessary parameters for scautod are:

system:	Service name for the ServiceCenter system.
path:	Directory containing ServiceCenter database.
log:	(optional) Output file for diagnostic log messages.

(These first three parameters are also used in the **scenter** program.)

scauto:	Service name for SCAutomate.

Important: The last parameter is important. Just as ServiceCenter requires a service name, so does SCAuto. This service name specifies a TCP/IP port, and must be different than the ServiceCenter service name. This service name needs to be specified by SCAuto clients in order to connect to the SCAuto server.

Note: Under Windows NT these service names are defined in *Winnt\system32\drivers\etc \services*. The ServiceCenter installation on Windows NT creates a shortcut to this file. This text file can be edited with any text editor. For testing purposes, this service name may be specified directly as a port number without changing the services file.

System structure

SCAuto products run on various supported platforms. Each product uses TCP/IP socket architecture to connect to an scauto base server (scautod) running on the ServiceCenter server platform. Once a connection is established with the base, a process on the server is created. The new process runs similar to an scenter process in ServiceCenter and reads and writes event records directly to the event files.

ServiceCenter events are internally written and read using a set of RAD applications known as Event Services. Some functions that are provided by the Event Services applications are event mapping and application scheduling within ServiceCenter. These services enable RAD applications to create events for external processing and external programs to asynchronously schedule a RAD application.

Any RAD application can be written or modified to take advantage of Event Services to interact with external SCAuto applications. Currently the Problem, Inventory, and E-mail applications utilize Event Services and interact with SCAuto applications.

External Event Services

Applications external to the ServiceCenter environment are provided with multiple options for interfacing to RAD applications.

- The simplest option is a File Monitor interface comprised of external event schedulers and event files. Using this option, ServiceCenter input events are written by the applications to an input event file. That file is read by an SCAuto external scheduler and forwarded to the ServiceCenter Event Manager. ServiceCenter output events are read from the ServiceCenter Event Manager by an SCAuto file monitor and placed in an application output events file. Each instance of an external scheduler is started for a different file, enabling you to batch events, create more parallelism or have separation of external applications. Each file monitor can be started to process a different event or event list. This interface mode permits external programs to use standard file read/write with a predefined record structure to communicate with ServiceCenter.
- A second option is an API and run-time library that provides connection services and event services. Any program that can call a C function and be called from a C environment can utilize this API. Under this option, applications may connect to ServiceCenter with an *scauto_connect* call and specify which events and ServiceCenter server your application requires.

After successful connection, you may create, retrieve, or delete ServiceCenter events by issuing *scauto_create*, *scauto_query*, and *scauto_delete* calls. Prior to termination, your program issues an *scauto_disconnect* call. Applications can have multiple sessions with one or more ServiceCenter servers.

- Another option provided through SCAuto is the e-mail interface. The e-mail option provides e-mail services to ServiceCenter. With some extensions, this option can be used to handle other than e-mail events. An example of this would be a dispatch application which runs from e-mail or has an e-mail bridge. An open problem can be sent to interested parties and responses can be sent to update or close problems. SCAuto provides the capability of opening, updating, and closing problems from e-mail.

Application Enhancement

ServiceCenter increases in usability and value when connected to many other applications and environments. With the SCAuto suite of standard solutions these tasks can be accomplished in a structured and routine fashion. Systems integrators should be especially interested in SCAuto and the ServiceCenter Event Services facilities.

This document covers several modules developed for SCAuto and Event Services which interface with external systems. Other SCAuto implementations are also available, such as SCAuto for HP OpenView NNM, SCAuto for SPECTRUM, SCAuto for Tivoli, etc. Please check with your Peregrine Systems Inc. Account Representative for a current product list. The products in this document include:

- The **Paging Server** allows ServiceCenter to automatically generate pages in response to RAD-controlled events (e.g. open problems reaching a high severity).
- The **Fax Server** enables ServiceCenter to generate faxes from within RAD applications as an adjunct to e-mail or printing. The internal command file allows for easy customization and use with many fax systems.
- The **SCAuto Mail** server provides SMTP and MAPI e-mail integration with Service Center.

Chapter 2 Installation

SCAutomate Applications CD-ROM

The SCAuto Applications CD contains four SCAuto products:

- Fax Server
- SCAutomate Mail
- Paging Server
- SCAuto SDK

This chapter provides the instructions for installing the SCAuto applications in the Windows NT and UNIX environments.

- To install SCAuto Applications on Windows NT, run **setup.exe** from the NT subdirectory.
- To install SCAuto Applications on UNIX, run **install.sh** from the UNIX subdirectory.

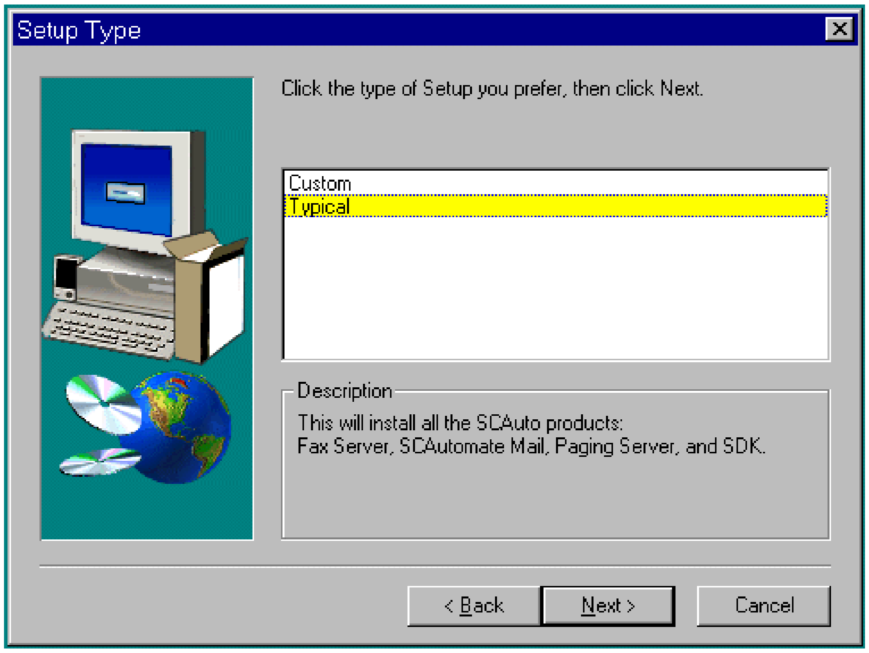
Windows NT Install

The bulk of the installation is done automatically. However, with each SCAuto application or utility you may need to perform specific setup configuration unique to that program. Please see the appropriate chapters in this manual for information regarding these customization procedures.

Note: Look for any ReadMe files which will provide the latest additional information.

To install SCAuto applications on Windows NT:

1. Insert the SCAuto Applications CD into your CD-ROM drive.
2. Access the CD.
3. Change to the \NT subdirectory.
4. Double click on **setup.exe**. The Welcome screen is displayed.
5. Click **Next**. The Setup Type screen is displayed.
6. Select the Setup Type:
   - **Custom** allows you to install specific product components.
   - **Typical** installs all four SCAuto products. The default is Typical.


Setup Type

Click the type of Setup you prefer, then click Next.

Custom
Typical

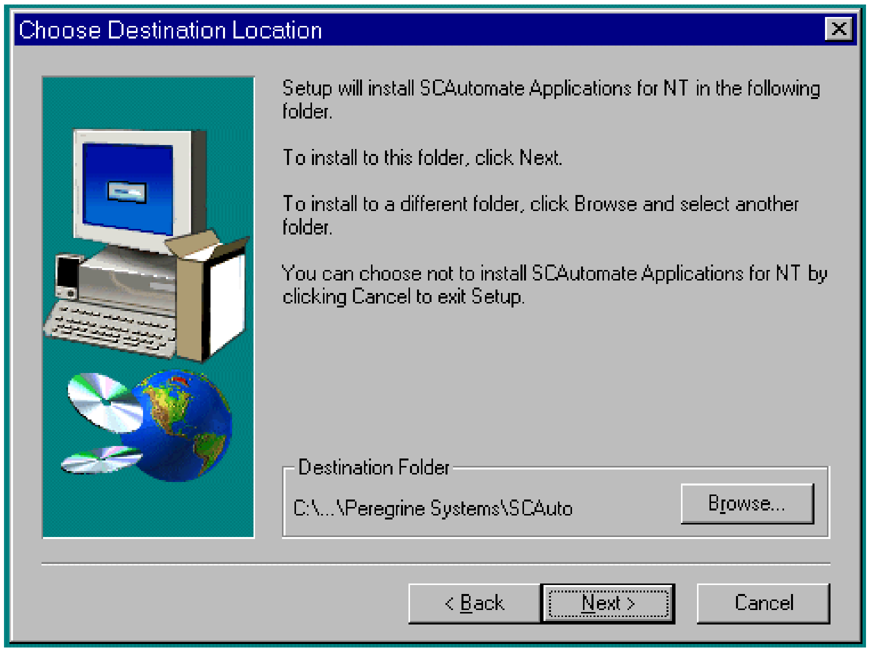
Description
This will install all the SCAuto products:
Fax Server, SCAutomate Mail, Paging Server, and SDK.

Note: If you select the Custom option, depending on which applications are selected, you may see different screens than those shown in this procedure.

7. Click **Next**. The Choose Destination Location screen is displayed.
8. Select a destination path to install the SCAuto applications.

   The destination location is the root directory for the SCAuto applications.

   - You can choose the default destination of *C:\Program Files\Peregrine Systems\SCAuto*
   - You can click the Browse button to install SCAuto in a different location. A new directory also can be created using Browse, or you can use an existing path.


Choose Destination Location

Setup will install SCAutomate Applications for NT in the following folder.

To install to this folder, click Next.

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

You can choose not to install SCAutomate Applications for NT by clicking Cancel to exit Setup.

Destination Folder

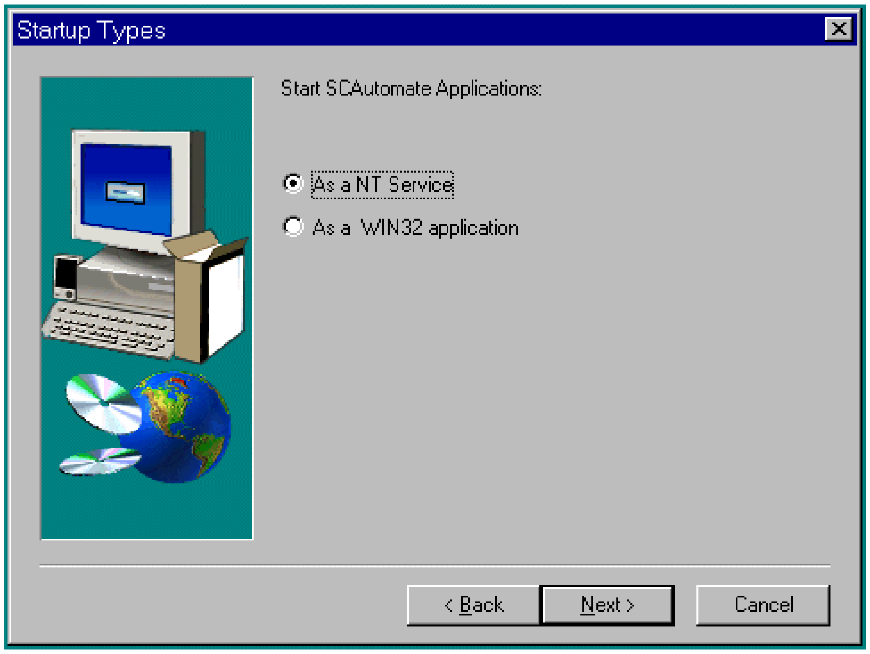
C:\...\Peregrine Systems\SCAuto

The installation process automatically adds a subdirectory for each SCAuto application installed:

- *Fax* for Fax Server
- *Mail* for SCAutomate Mail
- *Pager* for Paging Server
- *SDK* for SCAuto SDK

9. Click **Next**.

   The Startup Types screen is displayed.



Start SCAutomate Applications:

(●) As a NT Service

( ) As a WIN32 application

A *new installation* contains two options for starting SCAuto applications.

- **As a WIN32 application** that runs under a DOS window.

If you are *upgrading* SCAuto applications, the options are slightly different:

- **As a service from ServiceCenter** allows you to start SCAuto applications from ServiceCenter. This option appears only if that is the setup in your current system.
- **As an NT Service** runs SCAuto as a Windows NT service.

Note: If you can start the Fax and Paging services as Windows NT services:

All the command-line parameters for the SCAuto applications are moved from sc.cfg file to the new application configuration files:

Fax Server - scfax.ini

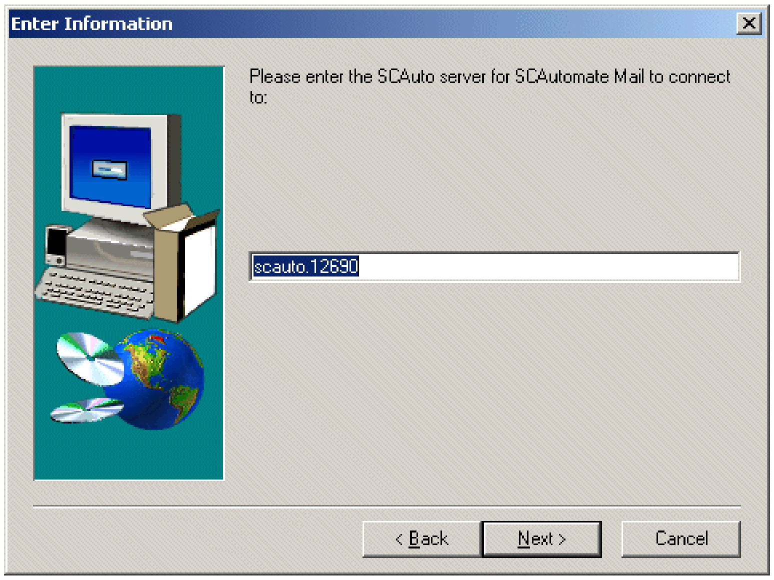
Paging Server - scpager.ini

Mail Server - scmapi.ini

The command line in sc.cfg is commented out.

10. Click **Next**.

The Enter Information screen is displayed.

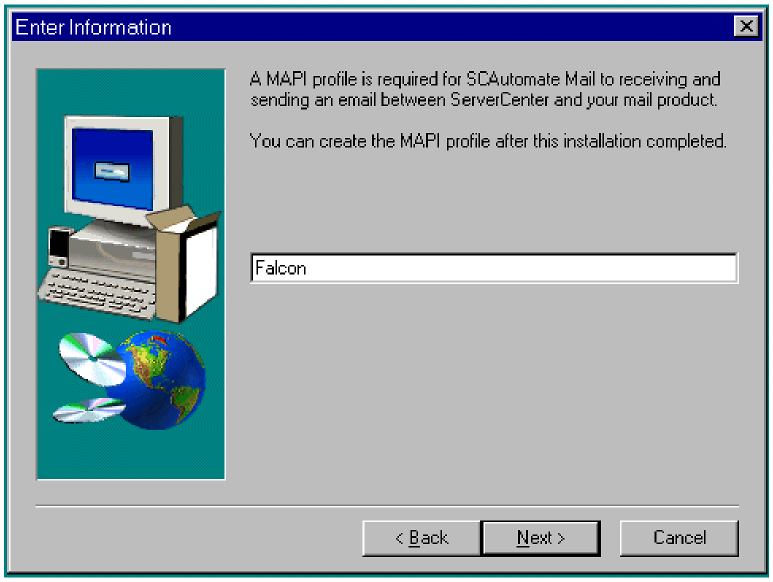


Please enter the SCAuto server for SCAutomate Mail to connect to:

scauto.12690

11. Enter the SCAuto server port as a hostname and port number separated by a period (default is *scauto.12690*).
12. Click **Next**.

    The Enter Information screen is displayed again. Now you are prompted for a MAPI profile. (This screen is for SCAutomate Mail only.)



A MAPI profile is required for SCAutomate Mail to receiving and sending an email between ServerCenter and your mail product.

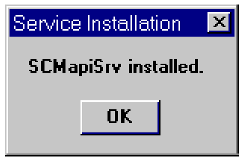
You can create the MAPI profile after this installation completed.

Falcon

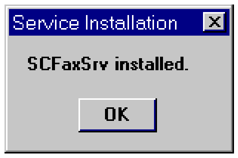
13. Enter a MAPI profile name (default is *Falcon*). The MAPI profile name is the one configured in the Windows NT Mail and Fax Control Panel. (See Chapter 3 for more information on MAPI.)

14. Click **Next**.

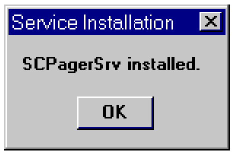
    A series of Service Information confirmation messages are displayed. The confirmation messages displayed are dependent on which SCAuto applications you installed.



SCMapiSrv installed.



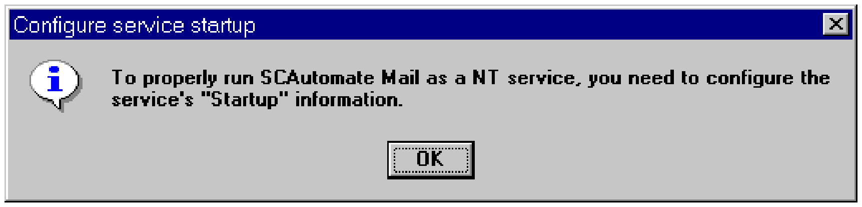
SCFaxSrv installed.



SCPagerSrv installed.

15. Click **OK** in each screen.

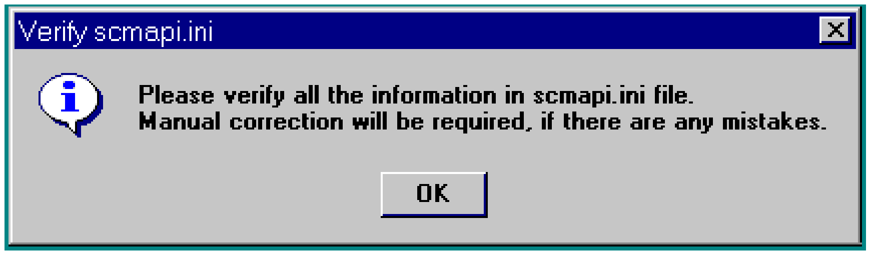
    If SCAuto Mail was installed, you are prompted to configure the SCAuto Mail service startup. These steps describe that configuration.



To properly run SCAutomate Mail as a NT service, you need to configure the service's "Startup" information.

a. This message box is only for the SCAuto Mail service. See *Running SCAuto Mail as a Windows NT service* on page 2-8 for instructions on how to configure the service for a selected user account to access the MAPI profile.

b. Next, you are prompted to verify the scmapi.ini.



Please verify all the information in scmapi.ini file.
Manual correction will be required, if there are any mistakes.

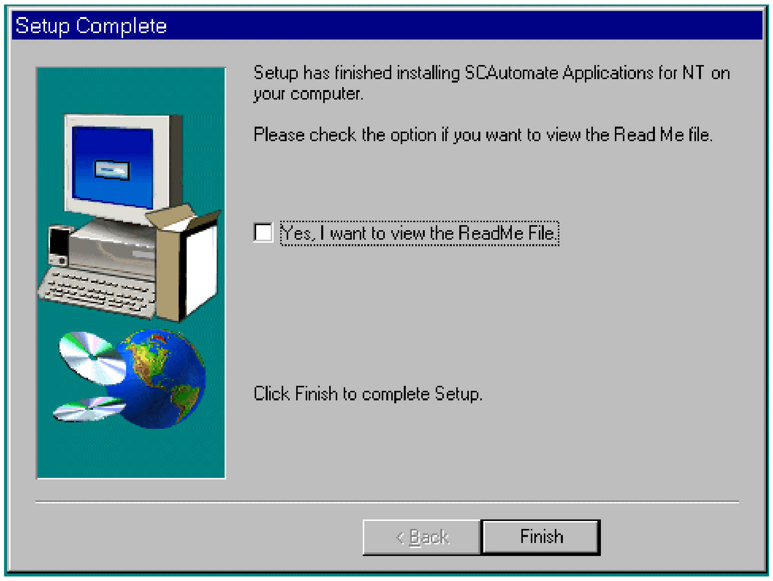
c. Click **OK**.

The scmapi.ini file is opened by Notepad:

```
events:1
usemaps:1
event_map_dir:EventMap
profile:Falcon
server:scauto.12690
log:scmapi.log
```

d. Verify that the settings match your system requirements. scmapi start-up parameters are discussed in the *SCAuto mail start-up parameters* section in Chapter 3. Click **OK**.

The Setup Complete screen is displayed.


Setup Complete

Setup has finished installing SCAutomate Applications for NT on your computer.

Please check the option if you want to view the Read Me file.

☐ Yes, I want to view the ReadMe File.

Click Finish to complete Setup.

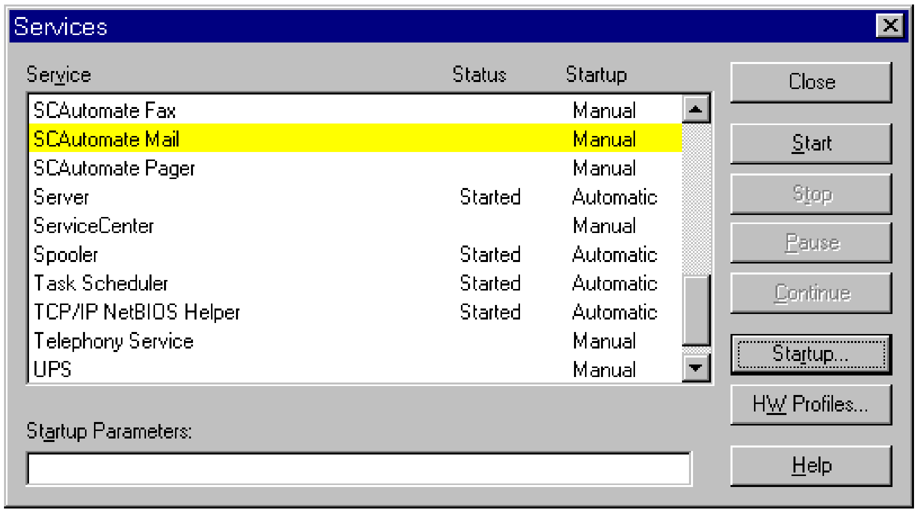
16. Select the Readme checkbox to display the readme file and click **OK**.

Running SCAuto Mail as a Windows NT service

To run SCAuto Mail as a Windows NT service, you need to configure the service *Startup* information. During the installation, the setup program installed the SCAuto Mail service to run under the default *LocalSystem* user ID. The selected user must have the rights to log on to the MAPI profile that you entered during the setup.

To set up the SCAuto Mail service to run under a specific user ID:

1. Open the Windows NT Control Panel.
2. Double-click on the **Services** icon. The **Services** window is displayed.
3. Highlight **SCAutomate Mail** from the Services list.

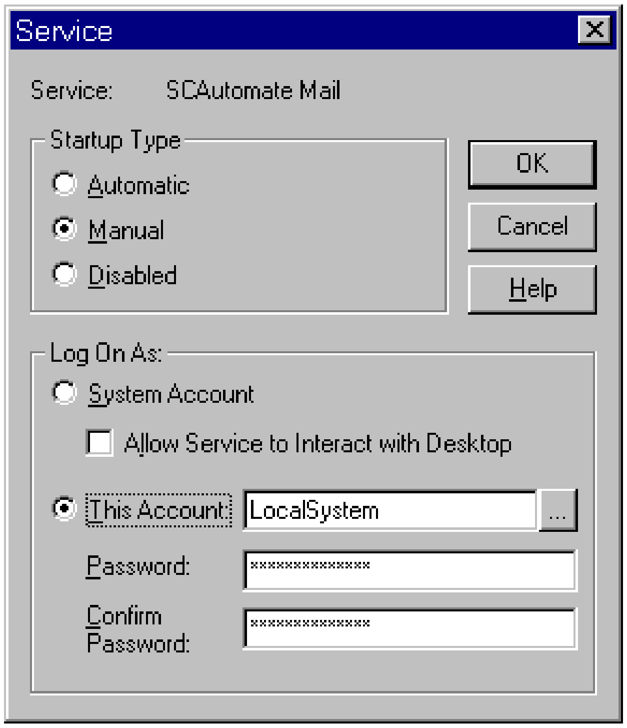


Service	Status	Startup
SCAutomate Fax		Manual
SCAutomate Mail		Manual
SCAutomate Pager		Manual
Server	Started	Automatic
ServiceCenter		Manual
Spooler	Started	Automatic
Task Scheduler	Started	Automatic
TCP/IP NetBIOS Helper	Started	Automatic
Telephony Service		Manual
UPS		Manual

4. Click on the **Startup** button.

The Service dialog box is displayed.

5. In the bottom of the box, click on the **This Account** button. The *LocalSystem* user ID is displayed in the *This Account* field.



Service: SCAutomate Mail

Startup Type: Automatic / [✓ Manual] / Disabled

Log On As: System Account

☐ Allow Service to Interact with Desktop

[✓ This Account]: LocalSystem

Password: ***************

Confirm Password: ***************

6. Click on the **Browse** button next to the *This Account* field. The Add User dialog box is displayed.
7. Double-click on the desired user ID from the Names list. The *Add Name* field displays the user ID.
8. Click **OK**. The Windows NT Service dialog box is displayed.

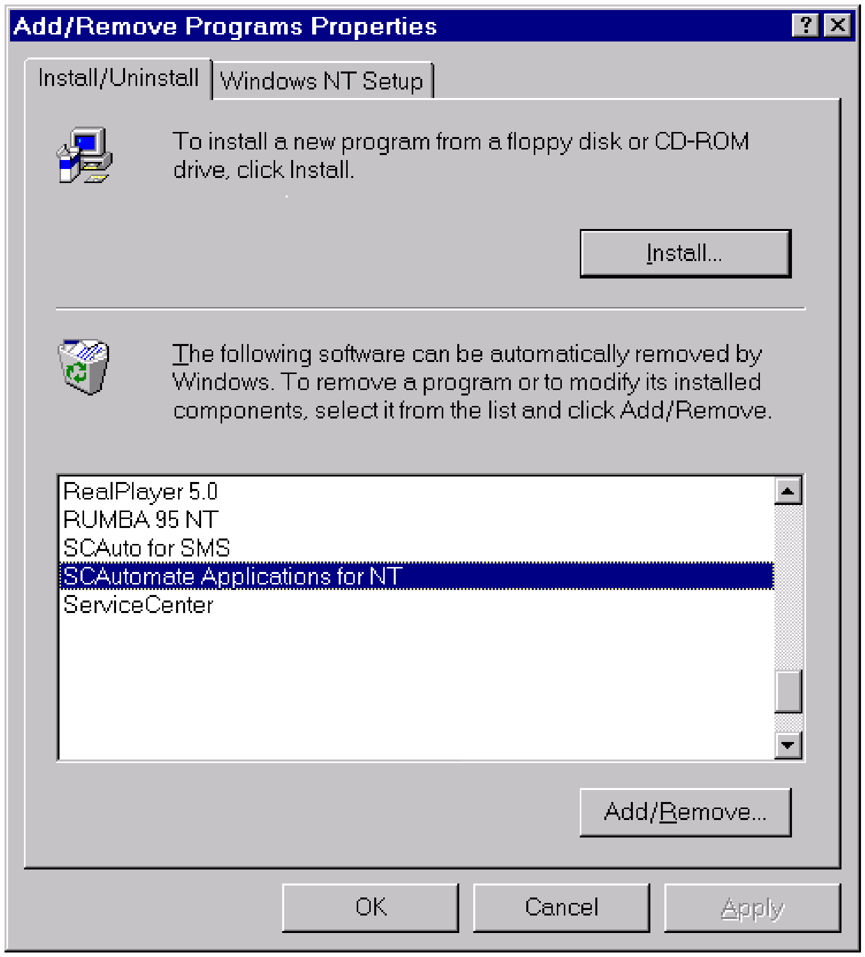
   The Account field now shows the user ID.
9. Enter the Password.
10. Re-enter the password in the *Confirm the Password* field.
11. Click **OK**. A message is displayed stating that the user has been granted the Log On As A Service right.
12. Click **OK**. You are returned to the Windows NT Services dialog box.

When you start the ServiceCenter service, the service now runs under the new user ID instead of the Local system user ID.

Uninstalling the SCAuto applications

To uninstall SCAuto Applications for Windows NT:

1. Access the Windows NT Control Panel.
2. Double-click the **Add/Remove Programs** icon. The Add/Remove Programs Properties window is displayed.
3. In the Install/Uninstall tab, highlight **SCAutomate Applications for NT**.
4. Click **Add/Remove**. A confirmation message is displayed.
5. Click **Yes**.
6. Click **OK** to exit the Control Panel.


Add/Remove Programs Properties

Install/Uninstall | Windows NT Setup

To install a new program from a floppy disk or CD-ROM drive, click Install.

The following software can be automatically removed by Windows. To remove a program or to modify its installed components, select it from the list and click Add/Remove.

RealPlayer 5.0
RUMBA 95 NT
SCAuto for SMS
SCAutomate Applications for NT
ServiceCenter

UNIX Install

The bulk of the installation is done automatically for you. However, with each SCAuto application or utility you may need to perform specific setup configurations unique to that program. Please see the appropriate chapters in this manual for information regarding these customization procedures.

Note: Case sensitivity may be dependent on the UNIX system that you are using.

Note: You can login as the root or any user to run the installation for SCAuto for UNIX. When you login as the root:

- An application specific initialization file (named *init*) is created under the /etc/scauto/APPLICATION/4.0 directory. For example: /etc/scauto/**scmail**/4.0/init for SCAuto Mail application.
- You are prompted for a SCAuto product user name.

To install SCAuto products for UNIX:

1. Mount the SCAutomate CD-ROM.

   This procedure assumes that the CD is mounted on */cdrom*. If it is mounted elsewhere, adjust the following steps accordingly. Note that depending on the options used in mounting the CD, the directories and filenames on the CD may be in lower case.
2. Change to the **/cdrom/UNIX** directory.
3. Start INSTALL.SH by issuing the command:

   ./INSTALL.SH

   A list of installation options is displayed.

```
Please select from the following SCAuto load options:

 1) Fax
 2) Mail
 3) Pager
 4) SDK
 5) All


 q) quit

 Enter option number:
```

4. Enter the number for the SCAuto applications you want to install.
5. Enter a temporary directory to unpack the software components. Enter a full path to an existing directory.
6. Enter a target directory for the product installation.

   For example, enter: **/scauto**

   You must enter a full path for the target directory. The target directory is the root directory for the SCAuto products.

   The installation process automatically adds a subdirectory for each SCAuto application that is installed:

   - *Fax* for Fax Server
   - *Mail* for SCAutomate Mail
   - *Pager* for Paging Server
   - *SDK* for SCAuto SDK
7. Enter an SCAuto product user (only if you logged in as the root).

   The user must have full access rights to the target directory. This is specified as *user:group*.

   For example, enter: **scuser:scgroup**
8. In you are installing SCAuto Mail, you must provide:

   - The mailbox name for SCAuto Mail. The name is the file name of the mailbox to monitor for incoming e-mail. The location of this file is dependent upon the UNIX version:
     - For AIX, use: */var/spool/mail/username*
     - For HP-UX or Solaris, use: */var/mail/username*
   - Enter the SCAuto Server hostname and port number. The format is *hostname.port*.

     The hostname is the SCAuto server (*scautod*) to which the user will connect. This is specified as a hostname plus service name or port number, separated by a period. For example, you could enter: **host.12690**

Chapter 3 SCAuto Mail for Windows NT

Overview

SCAutomate (SCAuto) Mail is a Windows NT program which allows sending and receiving of e-mail within ServiceCenter using different external mail applications. Under Windows NT, SCAuto Mail uses the *Messaging Application Program Interface* (MAPI). Microsoft Exchange, Lotus Notes, Lotus cc:Mail and other mail systems support this interface. SCAuto Mail for Windows NT is an SCAuto adapter product.

Note: SCAuto Mail sends outgoing e-mail the same way as the SCEMAIL program that comes with ServiceCenter. If you are already running SCEMAIL, you replace it with SCAuto Mail.

Mail profiles

MAPI uses the concept of a *profile*. A MAPI profile contains all of the information necessary to login to a group of mail services. A profile is not the same as a user login, and a single user may have many different entries within one MAPI profile.

For example, your SCAuto Mail/MAPI profile could be *Joe*; however, that profile contains the MS Exchange, cc:Mail, Lotus Notes etc. login and account information which allows you to interface with those systems. SCAuto Mail signs on using the SCAuto Mail profile, not the external mail account or login names. It is for this reason that a unique SCAuto Mail profile needs to be established in addition to a standard mail account.

Profiles were introduced with MAPI in Microsoft Windows 95 and Windows NT 4.0. The default Windows NT 3.51 system does not use profiles unless additional software has been installed which upgraded the MAPI system (i.e., such as Microsoft Exchange Client or Lotus cc:Mail). SCAuto Mail does not work under Windows NT 3.51 unless MAPI support is installed.

Note: It is highly recommended that SCAuto Mail be given its own MAPI profile and its own mailbox or mail account. This mail account acts as a gateway to and from ServiceCenter.

Configuration

SCAuto Mail files

Executable files

There are two executable files. *Scmapi.exe* sends and receives e-mail between external mail clients and ServiceCenter. *Scmapisrv.exe* allows you to start SCAuto Mail as a Windows NT service.

Configuration files

There are two configuration files. *Scmapi.ini* contains the SCAuto Mail starting parameters. You should add all the optional parameters that you need to this file. The second file is *scmapi.cfg*. When you start SCAuto Mail service, *scmapisrv.exe* reads scmapi.cfg and starts the SCAuto Mail application according to the command lines in the *scmapi.cfg* file. You can start more than one SCAuto Mail application from the SCAuto Mail service by adding more SCAuto Mail command lines in the *scmapi.cfg* file.

Event map files

The map files allow you to customize your e-mail events.

Outlook form

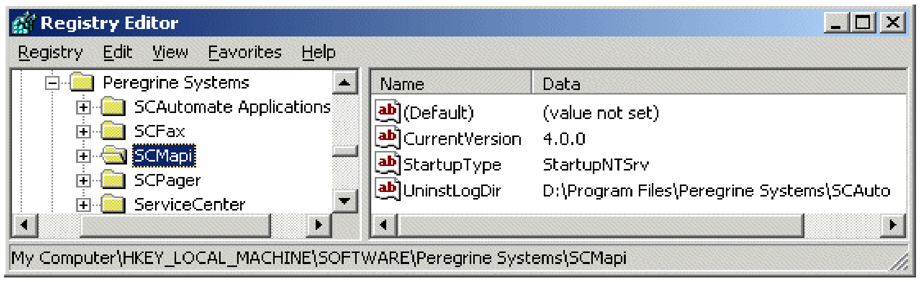
Pmo.oft is a Outlook message item. You can open a ServiceCenter problem ticket by sending e-mail using this form. The file pmo.oft works with Outlook 97 and pmo_98.oft works with Outlook 98 and 2000.

Registry entries

The SCAuto Application installation creates an SCAuto Mail application registry key and an SCAuto Mail service registry key in the Windows registry.

SCAuto Mail application registry key

The SCMapi key contains three registry values and one subkey.



Name	Data
(Default)	(value not set)
CurrentVersion	4.0.0
StartupType	StartupNTSrv
UninstLogDir	D:\Program Files\Peregrine Systems\SCAuto

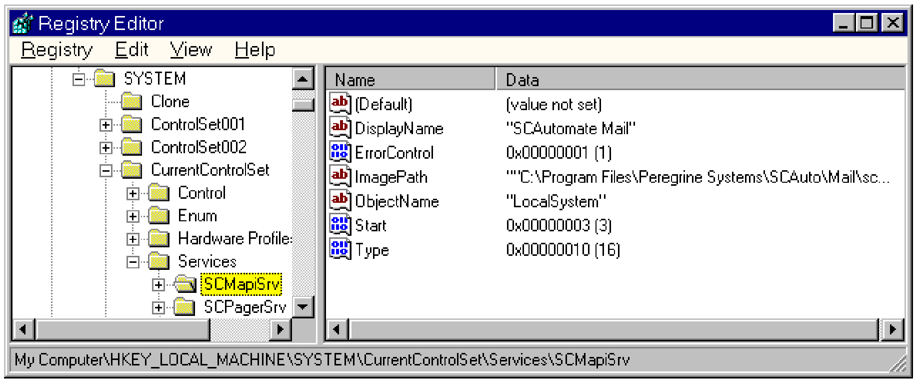
My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Peregrine Systems\SCMapi

The three values are:

- CurrentVersion: SCAuto Mail application version number.
- StartupType:
  - StartupNTSrv - start as a Windows NT service
  - StartupSC - start from ServiceCenter
  - StartupWin32App - start as a Win32 application
- UninstLogDir: points to the directory for uninstall file

The subkey is named according to the version of SCAuto Mail.

SCAuto Mail service registry key:



Name	Data
(Default)	(value not set)
DisplayName	"SCAutomate Mail"
ErrorControl	0x00000001 (1)
ImagePath	""C:\Program Files\Peregrine Systems\SCAuto\Mail\sc...
ObjectName	"LocalSystem"
Start	0x00000003 (3)
Type	0x00000010 (16)

My Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SCMapiSrv

Adding a MAPI profile

Windows NT does not automatically install the necessary MAPI support files. These files are installed whenever a MAPI-compliant e-mail client is installed. Therefore you should ensure that at least one e-mail client has been installed before you can create a MAPI profile. Note that Windows NT 4.0 comes with the *Windows Messaging* e-mail client, which may be used for this purpose if no other client is available.

After consulting documentation accompanying the external mail product you have chosen, use the following steps to add a new profile for SCAuto Mail.

1. Open the Windows NT Control Panel.
2. Double-click on the **Mail** or **Mail and Fax** icon. The MS Exchange Settings Properties window is displayed.

Note: If you do not have one of these icons, it is likely you have an incompatible older version of MAPI, or no e-mail clients have ever been installed on that machine.

3. Click **Show profiles...**
4. Click **Add...**
5. Select the service to use (make sure only one is selected).
6. Name and configure the profile as directed by the dialog box. This procedure allows you to assign the mailbox or mail user to be used by SCAuto Mail.

Note: This profile may be tested by logging into it normally with a MAPI-compliant mail client (Microsoft Exchange, cc:Mail, etc.).

SCAuto Mail Start-Up Parameters

The following optional parameters may be given when starting *scmapi*:

Note: For true or false parameters, put 1 or 0 after the colon. Include the hyphen before the parameter name if the parameter is on the scmapi command line. Omit the hyphen if the parameter is in the *scmapi.ini* file.

Parameter	Description
-events:	Allows creation of problem or change management events via e-mail. See *Events Via E-mail* on page 3-17.
-usemaps:	It indicates that the new, enhanced style of putting tags in e-mail bodies is to be supported. If this parameter is set to :0 or is absent, scmail only honors the old style of tags.
-event_map_dir: <EventMap>	It is the relative path from the home directory to the directory containing the event map files. These are text files which control the generation of various types of events from tags and values specified in e-mail bodies. Normally this is specified as *EventMap* and the EventMap directory is a subdirectory of the home directory.
-profile: <Mapi Profile Name>	It is the MAPI profile name.
-server: <host.service>	The SCAuto server to connect to. This is specified as hostname plus service name or port number, separated by a period. The default is *scauto*.
-log: <file>	File to log messages. The default is *scmapi.log*. You can specify **con** to log messages to the console.
-keepmail:	1 = Do not delete mail or events after they have been sent successfully.
-noincoming:	1 = Do not send incoming mail to ServiceCenter.
-nooutgoing:	1 = Do not retrieve events from the ServiceCenter eventout queue.
-scmapi_address_delimiters:<delimiter>	Delimiter character(s) used between addressee names.
-scmapi_convert_addr_type:<mail address type>	Convert incoming mail From field to entered mail address type.

Parameter	Description
-**clean:**	For ServiceCenter outbound email, removes the following items from the email message's body: • ServiceCenter Operator:operator name • SCenter_cc:cc name
-**savesent:**	0 = SCMAPI should delete mail from MAPI mailbox after forwarding to SC. 1 = save mail in MAPI mailbox. Default is 0.
-**sleep: <n>**	Number of seconds to sleep between checking for events and mail. Default is 10 seconds.
-**gui:**	1 = Allow a pop-up dialog if additional login information is required (no profile was passed on the command line, or a password is required).
-**admin: <id>**	ServiceCenter operator id to receive mail that is not addressed with *Co:*, or mail that cannot be delivered. Default is *falcon*.
-**debug:**	1 = Prints more diagnostics to scmapi.log. This turns on *-keepmail* as well.
-**debugscautoevents:**	This parameter enables SCAuto Mail to log more event generation information to the SCAuto Mail's log file. Default value is 0.
-mapi_mailserver_reconnect:	Retry to connect to SCAuto Mail to MAPI mail server 0: Do not retry to connect to the MAPI mail server. Default is 0.
-reconnect_interval:<n>	Number of seconds to sleep between retrying. Default is 120 (two minutes).
-numberof_retry_connect:<n>	Number of retries before exiting. Default is 10. 0 = retry forever.

Parameter	Description
-sc_reconnect:	Allows SCAuto Applications to retry to reconnect to ServiceCenter when ServiceCenter is restarted. 1: Retry connecting to ServiceCenter. 0: Do not try to reconnect to ServiceCenter server. Default is 0. This parameter works together with - reconnect_interval and - numberof_retry_connect. This parameter impacts the K (kill) command in ServiceCenter's system status list. The K command closes the connection between ServiceCenter and SCAuto Mail. When this parameter is not set to 1, SCAuto Mail checks the connection and if the connection is broken then SCAuto Mail exists. When this parameter is set to 1, SCAuto Mail sees a broken connection, but has no knowledge of it's caused by the K command or ServiceCenter shutdown. So, SCAuto Mail will try to reconnect even after a K command.
-terminate_mapi_string:<MAPI errors>	The MAPI error message(s) that cause a termination of the SCAuto Mail. This parameter overwrites the mapi_mailserver_reconnect. The MAPI error messages could be: E_OUTOFMEMORY, E_FAIL, MAPI_E_LOGON_FAILED, MAPI_E_NETWORK_ERROR, AND MAPI_E_DISK_ERROR. User can specify more than one MAPI error and separate them by a comma.
-usersepchar:<char>	Allows the user to specify a different evsepchar by entering the decimal representation of a character from 1 to 255. Default to the '^' if not defined.
-send_mail_as_attach:<file name>	Send the ServiceCenter's outbound mail message as an attachment <file name> in an email. When SCMAPI and ServiceCenter are installed in different NT language systems, user can save the attachment in the current system and then open it with a multi-language enabled application (e.g., Microsoft Word, Internet Explorer, etc.) to view it.

Setting-Up SCMAPI Configuration File - scmapi.ini

During the SCAuto Mail installation, a configuration file named *scmapi.ini* is created in the SCAuto Mail directory. The default *scmapi.ini* contains:

```
events:1
usemaps:1
event_map_dir:EventMap
profile:Your Profile Name
server:scauto.12690
```

You can set any of the optional parameters to control how to start the SCAuto Mail.

SCAuto Mail 4.0 takes parameters from both the *scmapi.ini* file and the command line. It is preferable to put the parameters in the *scmapi.ini* file.

Setting-Up Scmapi.cfg for SCAuto Mail Service

SCAuto Mail service (scmapisrv.exe) starts the SCAuto Mail processes that defined in scmapi.cfg file. You can specify more than one SCAuto Mail process in the scmapi.cfg file. Each SCAuto Mail process read the parameters from scmapi.ini file. You can add more optional parameters for each SCAuto Mail process by giving the command line parameters for the SCAuto Mail process.

The simplest scmapi.cfg file contains just one line:

```
scmapi
```

SCAutomate Mail service starts one SCAutomate Mail process.

The following scampi.cfg file starts two SCAuto Mail processes. The first line takes all the SCAuto Mail start-up parameters from scmapi.ini file.

The second line takes one more SCAuto Mail start-up parameter **noincoming** from the command line.

```
scmapi
scmapi -noincoming:1
```

Note: The command line parameter overrides the same parameter specified in the scmapi.ini file.

Note: Please test each command line separately. Make sure each service starts successfully and then put all the command lines in one scmapi.cfg file.

Setting-Up Event Map Files

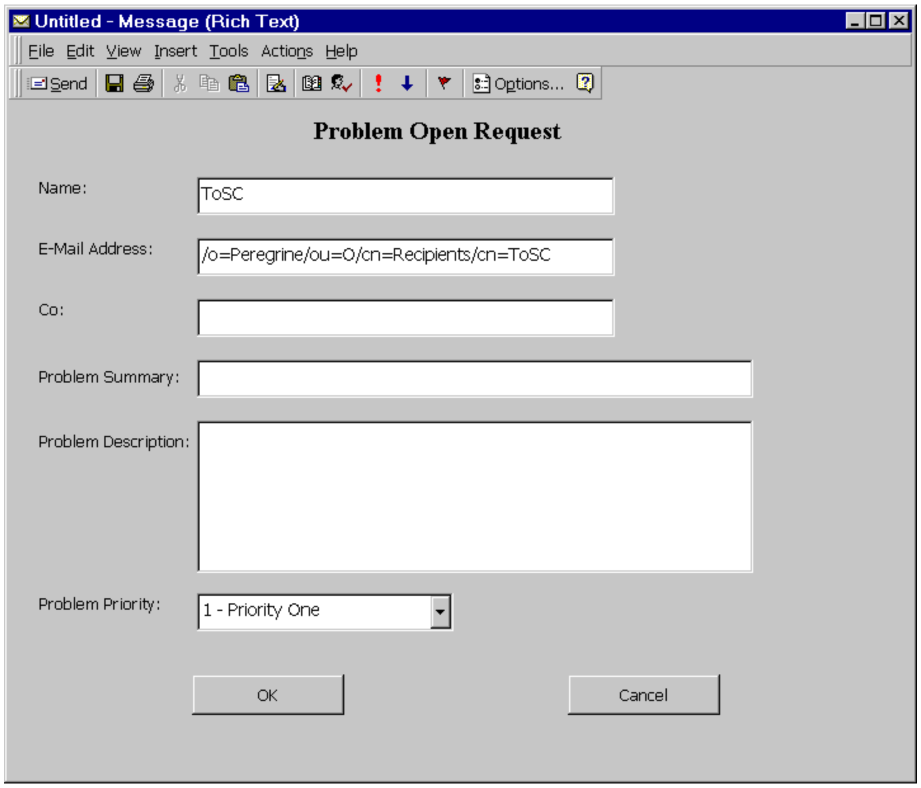
These .map files define how one or more ServiceCenter events going to ServiceCenter should be constructed from the e-mail messages.

This type of .map file has a special syntax which is oriented toward the creation of ServiceCenter event strings.

Please refer to Chapter 5 for details.

Microsoft Outlook Form

SCAuto Mail Version 3.1 installed a Microsoft Outlook form, named pmo.oft, in the Mail From subdirectory of your SCAuto Mail directory. You can use this form to open a ServiceCenter problem ticket.


Untitled - Message (Rich Text)
File Edit View Insert Tools Actions Help
Send
Options...

Problem Open Request

Name:	ToSC
E-Mail Address:	/o=Peregrine/ou=O/cn=Recipients/cn=ToSC
Co:	
Problem Summary:	
Problem Description:	
Problem Priority:	1 - Priority One

Form fields

Name	A mail user account that SCAuto Mail watches for incoming messages.
E-Mail Address	The e-mail address of the mail user account. This e-mail address will be automatically resolved for a user who has an e-mail address in an address book that is specified in the MAPI profile.
Co	A ServiceCenter recipient.
Problem Summary	A brief description for a problem.
Problem Description	A detail description for a problem. This field can not contains a “SC_” string, a “^” , or a “|” character. They are reserved for SCAuto Mail.
Problem Priority	The priority for a problem.
OK	Send this e-mail.
Cancel	Exit.

Starting SCAuto Mail

ServiceCenter release 1.4 or higher must already be installed and operational, and the SCAuto server (*scautod*) must be running, for SCAuto Mail to start and function properly.

SCAuto Mail version 4.0 takes the start-up parameters from the *scmapi.ini* file or the SCAuto Mail command line. *Scmapi.ini* is preferred. Before you start SCAuto Mail, please verify all the parameters in the *scmapi.ini* file.

It is recommended you test your SCAuto Mail configuration before running SCAuto Mail as a service or automatically starting SCAuto Mail from the ServiceCenter configuration file. The easiest way to do this is to start SCMAPI.EXE from Windows Explorer.

You can verify that the background processor has started successfully by checking the *scmapi.log* file. A successful start-up message reads: *Initializing*

From Windows Explorer

1. Change to the SCAuto Mail directory.

   By default this is:
   c:\Program Files\Peregrine Systems\Scauto\Mail
2. Double click on **scmapi.exe**.

In this case, scmapi.exe takes all the parameters from the *scmapi.ini* file.

From program group

1. This is only available when you selected to start SCAuto Mail as a WIN32 application.
2. Click Start\Programs\SCAutomate Applications\Mail

In this case, scmapi.exe takes all the parameters from the *scmapi.ini* file.

From a command line

Command line format:

For example, you could enter:
scmapi *-optional_parameter:parameter_value*

ex: scmapi -log:scmapi.log -profile:”My Profile Name”

1. Change to the SCAuto Mail directory.

   For example, you could enter:
   cd c:\Program Files\Peregrine Systems\SCAuto\Mail

2. Enter the command **scmapi** followed by any optional parameters. You need to use *double-quotes* if the profile name contains spaces.

   scmapi -profile:“*My Profile Name*”

3. If a password is needed to login to the profile, you must use the **-gui:1** parameter and scmapi will prompt you for the login information.

4. You can verify the background processor has started successfully by checking the *scmapi.log* file. A successful start-up message reads: *Initializing*.

Note: The command line parameter overrides the same parameter specified in the scmapi.ini file.

From control panel\services

1. This is only available when you select to start SCAuto Mail as a Windows NT service.

2. Make sure you set the service start-up logon account for the MAPI profile. To do this:

   a. Open Control Panel/Services.

   b. Select SCAuto Mail.

   c. Click **Startup**.

   d. Click **This Account**.

   e. Click **Browse**.

   f. Select a user and click **Add**.

   g. Type the password.

   The selected user has to have the rights to logon to the mailbox specified in the MAPI profile.

From the ServiceCenter configuration file

This allows SCAuto Mail to be started automatically whenever ServiceCenter is started.

Note: To automatically start SCAuto Mail when ServiceCenter is started, you need to set up a user ID through the Control Panel\Services for the ServiceCenter service, as in the previous section.

Note: It is likely that SCAuto Mail will not be able to run from the ServiceCenter *sc.cfg* configuration file under Windows NT. Currently, Microsoft Exchange Server is the only MAPI service provider that can be started from the configuration file. If Microsoft Mail, Lotus Notes, cc:Mail, or other MAPI service provider is used, you may only be able to start SCAuto mail interactively. This problem results from the implementation of MAPI on Windows NT, and may be fixed in future Windows releases.

1. Select the *Config File* entry from Windows' ServiceCenter menu. This starts an editor to customize the *sc.cfg* file which resides in the ServiceCenter RUN directory.
2. Find the sample line for *scmapi* and uncomment it by removing any # characters at the start of the line. If the line does not exist, you may add your own as shown:
3. Add any optional parameters you wish to use; otherwise *scmapi.exe* takes all the parameters specified in the *scmapi.ini* file.

From a Microsoft Windows menu or icon

1. From Microsoft Windows Explorer, open the directory into which SCAuto Mail was installed. The default is:

   C:\Program Files\Peregrine Systems\SCAuto\Mail
2. Click the right mouse button on the *scmapi.exe* file and drag it to the desktop to create a shortcut, or use the *Taskbar* settings to create a menu item (under Windows NT 4.0).

Note: If you create the item in the *Startup* menu, *scmapi.exe* will be started every time that user logs in.

3. Modify the scmapi.ini file and add any necessary parameters, or modify the shortcut properties to add command line parameters (the profile name, extra options, etc.).
4. Start SCAuto Mail by double-clicking on the desktop shortcut icon or selecting it from the menu.

Using E-mail with ServiceCenter

Sending ServiceCenter mail to e-mail

Sending ServiceCenter mail to e-mail users is a very simple process. Your System Administrator has to login and change the user's operator or contact record to point to the external e-mail address for that user. This is accomplished through the following steps:

1. Login to ServiceCenter with an account that has SysAdmin authority.
2. Go to the operator or contact record.
3. Enter the e-mail address for that person in the e-mail field.
4. Save that operator or contact record.

Note: There are different syntax variations when entering an e-mail address. Enter the name as it appears in the address book of an external mail client. You may also use SMTP-style addresses of the form *username@host.com*.

Once you have made the outlined changes to the operator or contact record, any user who has access to send mail can send ServiceCenter mail. If mail sent from ServiceCenter is undeliverable, it is returned to the user with an error message.

Sending e-mail back to ServiceCenter

Since ServiceCenter operators are not valid e-mail addresses, mail into ServiceCenter must be addressed specially. The mail must be sent to the mail account to which SCAuto Mail logged in (defined by its MAPI profile). The ServiceCenter recipient is specified by adding a line like the following in the body of the mail message:

Co: bob.helpdesk

Note: The *Co:* must not be indented. If you do not specify the Co:, mail routes to *falcon* by default. If the mail is successfully sent, it is removed from the MAPI profile's incoming mailbox (this can be disabled).

Note: The event background processor in ServiceCenter must also be started in order to process the e-mail events in the *eventin* file. This should automatically be started with ServiceCenter 2.0 and later, but may not be running automatically with older versions.

Events via e-mail

Problems can be opened, updated, and closed by sending e-mail via SCAuto Mail. The **events** parameter must be set to 1 in the *scmapi.ini* file to enable this facility. You can use a map file to customize your e-mail by setting the **usermap** parameter to *1* in the *scmapi.ini* file.

Mail without using map file

If you do not want to use the event map, please make sure the **usermap** parameter is set to *0* in the *scma.ini* file. Within the body of the e-mail message, include these fields:

SC_event: event_type

This specifies the type of event created by this e-mail message. Valid values are *pmo, pmu* and *pmc*. *Pmo* is a problem open event, *pmu* is problem update, and *pmc* is problem close.

SC_var: id

This identifies the problem in question. For problem open (*pmo*) events, this is the logical name field (and is optional). For problem update and close events, this must be the number of the problem being accessed.

SC_var: id, category

This is an alternate form for problem opens that allows specification of a problem category (otherwise the ServiceCenter default is used).

SC_user: userid

Allows specifying the operator to use for the event.

Note: These fields must not be preceded by white space.

For compatibility with PNMS, the fields may be called *PNMS_event* and *PNMS_var*. Alphabetic case is not significant for these prefixes.

The date fills in automatically. The header and body of the e-mail message become the problem description, or update/close description.

The following is a sample e-mail message for a problem update:

```
From: Field Service Rep
To: ServiceCenter
Subject: Maybe this solves it…

SC_event: pmu
SC_var: 57133
Plugged in machine. Try connecting again and see if it works.
Joe.
```

Mail with a map file

The following is a sample map file:

```
# This is a pmo map file

SC_priority != ""
SC_description != ""

# logical name
^

# network name
^

# reference no
^

# cause code
^

# description
SC_description^

# action,2 (unused)
^

# action,3 (unused)
^

# network address
^

# type
SC_type^

# category
^

# domain
^

# objid
^

# version
^

# model
^

# serial no
^

# vendor
^

# location
^

# contact name
^
```

```
# contact phone
^

# resolution
^

# assignee name
^

# priority code
SC_priority^

# failing component
^

# system
^
```

To edit the file:

1. Set the **usermap** parameter to *1* in the scmapi.ini file
2. Compose your e-mail by using the tags defined in the map file.

The following is a sample e-mail message for a problem open:

```
SC_type:pmo
SC_description:Can’t access internet
SC_priority:1
```

This e-mail generates a ServiceCenter problem open event. A problem is opened in ServiceCenter with priority 1.

Additional Compatibility and Setup Notes

Important: SCAuto Mail runs only as a Windows NT service if the mail service providers are *tightly coupled*. As of this writing, the only mail service provider that does this is Microsoft Exchange Server. For other mail service providers, SCAuto Mail will probably need to be running on an interactive desktop.

- If using Lotus Notes, the following restrictions apply.
  - Only Lotus Notes versions 4.11 or higher are supported. Lotus Notes version 4.5.2 is highly recommended, as MAPI support has been improved in that version.
  - Read the Lotus Notes installation guides for special instructions on creating a MAPI profile.
  - Be sure to install a MAPI-compliant mail client *before* installing Lotus Notes. Such clients include Windows Messaging (a part of Windows NT 4.0), Microsoft Exchange, or cc:Mail. This applies even if these mail clients will not be used. This is necessary because Lotus Notes installation will not add MAPI support if it fails to find MAPI installed. If Notes is already installed, refer to the Lotus Notes release notes for possible workarounds.
  - After setting up a profile for use with Lotus Notes, edit the properties of the profile and select the Delivery tab. Change the selection under *Deliver new mail to the following location* to read *Lotus Notes Message Store*.
  - When scmapi starts, it prompts for a password, regardless of the **-gui** parameter.

Important: If you are running a version of Lotus Notes earlier than 4.5.2, then do not install Microsoft Office 97, or use Microsoft Outlook as your mail client.

- If using Lotus cc:Mail, the following restrictions apply.
  - Lotus cc:Mail for Windows version 7 or higher is required to work with MAPI. (This means that the release 6, or DB8, post office is required.
  - If the profile has a password, cc:Mail always prompts for a password, even if one is given on the command line. In this case, you must pass the *-gui:1* flag when you start scmap; otherwise the session terminates with an error. This can be avoided by selecting the *Remember Password* checkbox when logging in with a normal cc:Mail client.
- Periodically check the *outbox* of the MAPI profile for deleted messages that can be purged.

Trouble Shooting

If the verification test fails during connection between the SCAuto Base (scautod) and the SCAuto Application (scmapi) please check for the following common failures:

- Is scautod running on ServiceCenter Server Platform? Use ServiceCenter's *status* command to check. If it's *not running* start it from the status option *start schedulers* selecting *scauto.startup*. If the start fails the scautod daemon logs to the *sc.log* file, so be sure to specify and check the log for any error messages.Check the TCP/IP specifications for your platform. A service name other than that specified for ServiceCenter server is required, and the *scauto:* keyword should be specified in the *sc.ini* file with this name. If it *is running* check the service name specified in *the sc.ini* file on the ServiceCenter platform for the *scauto:* keyword. If the keyword is not specified scautod defaults to the services name *scauto*. This will be useful in the next step.
- Are the TCP/IP specifications correct on the SCAuto Windows platform? Check the host and service specification, are they specified and do they *match* the *host* and *service* name of s*cautod?* Check with your network administrator. Also, check your specification of passed parameters and make sure they all match (ex. scmapi -server:*host.service*).
- Is there connectivity between SCAuto Base(server) and the SCAuto Application(client)? Ping the SCAuto Bas (scautod) platform from the SCAuto application(scmapi) system. If this fails and all TCP/IP specifications are correct contact your network administrator for assistance.

If you have an event in the* eventin *file but no RAD application is invoked (ex.email event and mail not delivered) check the following:

- Is the Event Scheduler running on the ServiceCenter server platform? Use ServiceCenter *status* command to check. If it's *not running* start it from the status option *start schedulers* selecting *event.startup*. If the start fails review error messages and retry. If errors persist call Peregrine Systems Inc. Customer Support. If it *is running* and not processing, stop the Event Scheduler and build a new *schedule record* and restart.
- Review the *Basic Trouble Shooting* section in the ***Event Services Guide*** in regard to schedule record specifications.

If you have an event in the eventout file but the external SCAuto Application is not invoked (e.g., email events and mail are not sent), check the following:

- Is there connectivity to the SCAuto Base server? Review *Connection between the SCAuto Base (scautod) and the SCAuto Application (client)* trouble shooting section.
- Manually test your mail specifications by sending and receiving mail using your mail account specifications and logon. If this fails contact your mail administrator for assistance.
- Is SCAuto application (scmapi) active? If it is *not running* check the SCAuto application *logfile (scmapi.log default)* for errors. Correct errors and, restart using appropriate command with the debug option if available. The debug option will allow you to determine if the operation was attempted and the mail service product failed, or if it's an SCAuto application failure. If restart fails review the SCAuto application *logfile* for additional data. If the mail operation was attempted and failed, check the mail service products log, if any, for any errors or messages. If it *is running*, check the SCAuto application *logfile* for errors. Correct errors and, restart using appropriate command with the debug option if available.

If you do not have an event in the eventin file but the external SCAuto Application event has occurred (e.g., mail was sent but not received in ServiceCenter) check the following:

- Is there connectivity to the SCAuto Base server? Review *Connection between the SCAuto Base (scautod) and the SCAuto Application (client)* trouble shooting section.
- Manually test your mail specifications by sending and receiving mail using your mail account specifications and logon. If this fails contact your mail administrator for assistance.
- Is SCAuto e-mail active? If it is *not running* check the SCAuto application logfile for errors. Correct errors and, restart using appropriate command with the debug option. The debug option will allow you to determine if it's an SCAuto application failure or a product installation or specification failure. If restart fails using debug option review the SCAuto application logfile for any new error message information. If it *is running*, check the SCAuto application logfile for errors. Correct errors and, restart using appropriate command with the debug option if available. Recheck the logfile and correct errors and retry.

If the event is not the eventout file but the RAD Application has been invoked (e.g., a problem was opened and an e-mail event was expected) check the following:

- Please review the *Basic Trouble Shooting* and *Using Format Control to Send E-mail Messages* sections in the ***Event Services Guide***.

If you need assistance in the resolution of your problem, please have the following available before calling for support:

- Error logs produced by mailers and SCAuto application with debug option(scmapi).
- ServiceCenter log and Scheduler log(s) if specified.
- ServiceCenter msglog for affected applications, or services.
- ServiceCenter print of agent status(Event Services Menu).
- Unloaded Event records relevant to errors.
- Unloaded filter specifications if relevant.

Chapter 4 SCAuto Mail for UNIX

SCAuto Mail can run in a UNIX environment, allowing you to send and receive e-mail within ServiceCenter using external UNIX mail utilities. SCAuto Mail for UNIX is an SCAuto adapter product and should be obtained through a Peregrine Systems Inc. sales representative.

SCAuto Mail for UNIX sends and receives mail using the standard UNIX *sendmail* program. SCAuto Mail can deliver mail to any address that UNIX can deliver to. For incoming mail destined for ServiceCenter, SCAuto Mail monitors a UNIX mailbox.

Installation

The SCAuto Mail files

- SCAuto Mail for UNIX has a single executable: *scmail*.
- SCAuto Mail for UNIX uses two configuration files:
  - *scmail.ini* - contains the SCAuto Mail start-up parameters. This file is created in the SCAuto Mail target directory during installation.
  - *init* - contains the root directory for SCAuto Mail and the name of the parameters file, which is *scmail.ini*.
- A user can set the environment variable **SCAPPSINSTDIR** to point to a SCAuto application install directory. The SCAuto Applications get the install path as follows:
  - The path specified by the **SCAPPSINSTDIR** environment variable.
  - The current directory.
  - The value of parameter **home** in the *init* file.
- The Event map files allow you to customize your e-mail events.

Optional Parameters

The following optional parameters may be used when starting *scmapi*.

Note: The leading hyphens are used only on the scmapi command line. Leave them out when you place parameters in the *scmail.ini* file. Boolean options take 1 or 0 after the colon to turn them on or off, respectively.

Parameter	Description
events:	Allow creation of problem or change management events via e-mail. See *Events via E-mail* on page 4-7.
-**usemaps:**	It indicates that the new, enhanced style of putting tags in e-mail bodies is to be supported. If this parameter is set to :0 or is absent, scmail honors only the old style of tags.
-**event_map_dir: <EventMap>**	It is the relative path from the home directory to the directory containing the event map files. These are text files which control the generation of various types of events from tags and values specified in e-mail bodies. Normally this is specified as "EventMap" and the EventMap directory is a subdirectory of the home directory.
-**mailbox: <mailbox>(or -mbox)**	The file name of the mailbox.
-**server: <host.service>**	The SCAuto server (scautod) to connect to. This is specified as an optional hostname plus service name or port number, separated by a period. The default is *scauto*.
-**log: <file>**	File to log messages. The default is *scmapi.log*. You can specify **con** to log messages to the console.
-**keepmail:**	Do not delete mail or events once they are sent successfully.
-**noincoming:**	Do not send incoming mail to ServiceCenter.
-**nooutgoing:**	Do not retrieve events from ServiceCenter eventout.
-**keep_cc:**	0 = Do not retain CC field of e-mail. 1 = Retain CC field. Default is 0.
-**keep_to:**	0 = Do not retain TO field of e-mail. 1 = Retain TO field. Default is 0.
-**sleep:<n>**	Number of seconds to sleep between checking for events and mail. Default is 10 seconds.
-**admin: <id>**	ServiceCenter operator id to receive mail that is not address with *Co:*, or mail that cannot be delivered. Default is *falcon*.

Parameter	Description
-**debug:**	Prints more diagnostics to scmapi.log. This turns on -*keepmail* as well.
-**debugscautoevents:**	This parameter enables SCAuto Mail to log more event generation information to the SCAuto Mail's log file. Default value is 0.
-sc_reconnect:	Allows SCAuto Applications to retry to reconnect to ServiceCenter when ServiceCenter is restarted. 1: Retry connecting to ServiceCenter. 0: Do not try to reconnect to ServiceCenter server. Default is 0. This parameter works together with -reconnect_interval and -numberof_retry_connect. This parameter impacts the K (kill) command in ServiceCenter's system status list. The K command closes the connection between ServiceCenter and SCAuto Mail. When this parameter is not set to 1, SCAuto Mail checks the connection and if the connection is broken then SCAuto Mail exists. When this parameter is set to 1, SCAuto Mail sees a broken connection, but has no knowledge of it's caused by the K command or ServiceCenter shutdown. So, SCAuto Mail will try to reconnect even after a K command.
-reconnect_interval:<n>	Number of seconds to sleep before retrying. Default is 120 (two minutes).
-**-numberof_retry_connect:<n>**	Number of retries before exiting. Default is 10. 0 = retry forever.
-from	1=Send mail with sendmail command option -f. Default is 0.
-usersepchar:<char>	Allows the user to specify a different evsepchar by entering the decimal representation of a character from 1 to 255. Default to the '^' if not defined.

Setting up SCMAIL configuration file - scmail.ini

During the SCAuto Mail installation, a default configuration file named scmail.ini is created in the Mail directory. There are several parameters you can set to control how you want to start SCAuto Mail.

The following is a sample *scmail.ini* file:

```
events:1
usemaps:1
event_map_dir:EventMap
log:scmail.log
mailbox:/var/mail/scuser
server:host.12690
```

Setting-up Event Map Files

The files in the */EventMap/ToSC* directory define how one or more ServiceCenter events going to ServiceCenter should be constructed from the e-mail messages.

This type of .map file has a special syntax which is oriented toward the creation of ServiceCenter event strings.

Please refer to Chapter 5 for details.

Starting SCAuto Mail

Note: ServiceCenter release 1.4 or higher must already be installed and operational, and the SCAuto server (scautod) must be running for SCAuto Mail to start and function properly.

SCAuto Mail version 4.0 takes the start-up parameters from the *scmail.ini* file or the SCAuto Mail command line, preferably the *scmail.ini* file.

Before you start SCAuto Mail, please verify all the optional parameters in the *scmail.ini* file.

To start scmail

1. Change to the SCAuto Mail directory.

   For example: **cd /scauto/mail**.

2. Type the command **scmail**. For example, type **scmail&**

   You also can enter the command **scmail** followed by optional parameters, such as **noincoming**, which tells the system not to send incoming mail to ServiceCenter. For example, **scmail –noincoming:1 &**

3. Verify the background processor has started successfully by examining the log output. A successful start-up message reads: *Initializing*.

Once SCAuto Mail is successfully started, it checks for ServiceCenter e-mail events and turns them into e-mail messages. When mail arrives for the SCAuto Mail mailbox, the mail is converted into ServiceCenter e-mail events.

Using E-mail with ServiceCenter

Sending ServiceCenter mail to e-mail

You can send ServiceCenter mail to e-mail users. To configure this feature, a user's operator or contact record must be updated to point to that user's external e-mail address.

Note: To modify an operator record, you must have SysAdmin authority.

To modify the operator record:

1. Login to ServiceCenter.
2. Access the user's operator record.
3. Enter the e-mail address for that respective user in the E-mail Address field. In GUI mode, this field is found under the Notification tab.
4. Save the updated operator record.

Once you have changed the operator record, any user who has access to send mail can send mail from within ServiceCenter to the user's external e-mail address. If mail sent from ServiceCenter is undeliverable, it is returned to the user with an error message.

Sending e-mail back to ServiceCenter

Important: The event background processor in ServiceCenter must be running in order to process the e-mail events in the *eventin* file.

ServiceCenter must be configured so that operators can receive external e-mail. The e-mail messages must be sent to the mailbox sub-directory that is monitored by SCAuto Mail. For example, if the mailbox directory is */var/mail/scenter*, then mail should be sent to *scenter* on the local machine.

The e-mail message cannot be addressed directly to a ServiceCenter operator. The ServiceCenter recipient is specified in the message by adding an identifying line at the top of the message body or header. The syntax for the identifier is:

Co:<*operator name*>

where <*operator name*> is the ServiceCenter user for whom the e-mail is intended.

For example: **Co: bob.helpdesk**

Note: The *Co:* must not be indented. If you do not specify the *Co:*, the mail is routed to the *falcon* operator by default.

ServiceCenter processes read the identifying syntax and route the message to the appropriate ServiceCenter operator.

If the mail is successfully sent, the message is removed from the mailbox. However, this function can be disabled by using the *-keepmail* option with scmapi.

Note: A valid email is composed of some header lines followed by a blank line followed by the message content. For example:

```
Sending an email from the Solaris box by mail command:

mail scuser
From:sctestuser
Subject: This is a test email, following is a blank line that
separates the email header and email body

This is the mail message.
```

Events via e-mail

Problem tickets can be opened, updated, and closed by sending e-mail via SCAuto Mail. The **-events** parameter must be passed when starting scmapi to enable this facility. Within the body of the e-mail message, include these fields:

SC_event: event_type	This specifies the type of event created by this e-mail message. Valid values are *pmo, pmu* and *pmc*. *pmo* is a problem open event. *pmu* is problem update. *pmc* is problem close.
SC_var: id	This identifies the problem in question. For problem open (*pmo*) events, this is the logical name field (and is optional). For problem update and close events, this must be the number of the problem being accessed.
SC_var: id, category	This is an alternate form for problem opens that allows specification of a problem category (otherwise the ServiceCenter default is used).
SC_user: userid	Allows specifying the operator to use for the event.

Note: These fields must not be preceded by white space.

For compatibility with PNMS, the fields may be called *PNMS_event* and *PNMS_var*. Alphabetic case is not significant for these prefixes.

The date fills in automatically. The header and body of the e-mail message become the problem description, or update/close description.

The following is a sample e-mail message for a problem update:

```
From: Field Service Rep
To: ServiceCenter
Subject: Maybe this solves it…

SC_event: pmu
SC_var: 57133
Plugged in machine. Try connecting again and see if it works.  Joe.
```

Trouble Shooting

If the verification test fails during connection between the SCAuto Base (scautod) and the SCAuto Application (scmail), check for the following common failures:

- Is scautod running on ServiceCenter Server Platform?

  Use the ServiceCenter *status* command to check. If it is *not running*, start it from the status option *start schedulers* selecting *scauto.startup*.

  - If the start fails, the scautod daemon logs to the *sc.log* file. Be sure to specify and check the log for any error messages. Check the TCP/IP specifications for your platform, a service name other than that specified for ServiceCenter server is required, and the *scauto:* keyword specified in the *sc.ini* file must match this name.
  - If it *is running*, check the service name specified in *the sc.ini* file on the ServiceCenter platform for the *scauto:* keyword. If the keyword is not specified, scautod defaults to the services name *scauto*. This will be useful in the next step.

- Are the TCP/IP specifications correct on the SCAuto UNIX platform?

  Check the host and service specification. Are they specified and do they match the *host* and *service* name of s*cautod?*

  - Specifications may be made in local files (*/etc/hosts*, */etc/services*) or a name server may be used.

  Check with your network administrator for specifications.

  Check your specifications in the passed parameters and make sure they all match. For example, *scmail -server host.service*.

- Is there connectivity between SCAuto Base (server) and the SCAuto Application (client)?

  Ping the SCAuto Base (scautod) platform from the SCAuto application (client) platform.

  - If this fails and all TCP/IP specifications are correct, contact your network administrator for assistance.

If you have an event in the eventin file but no RAD application is invoked (e.g., pmo event and problem not opened) check the following:

- Is the Event Scheduler running on the ServiceCenter server platform?

  Use the ServiceCenter *status* command to check.

  If it is *not running*, start it from the status option *start schedulers* selecting *event.startup*.

  - If the start fails, review error messages and retry.
  - If errors persist, call Peregrine Systems Inc. Customer Support.

  If it *is running* and not processing:

  - Stop the Event Scheduler and build a new schedule record and restart.
- Review *Basic Trouble Shooting* section in the ***Event Services Guide*** in regard to schedule record specifications

If you have an event in the event.out file but the external SCAuto Application is not invoked (ex.e-mail event and mail not sent) check the following:

- Is there connectivity to the SCAuto Base server?

  Review the first trouble shooting item in this section.
- Manually send and receive mail using the mailbox assigned in the execution (e.g.. *scmail -mailbox user1*). If manual functions fail, check with your systems administrator.
- Is SCAuto application (e-mail) active?

  If it is *not running*, check the application logfile (standard out is default).

  - Correct errors and restart using the appropriate command with the debug option. The debug option allows you to determine if the mail operation was attempted and the mail service failed, or if it's an SCAuto e-mail failure.
  - If the restart fails, review the logfile for any additional information. If the mail operation was attempted and failed, check the syslog for any errors or messages.

  If it *is running*, check the application logfile for errors.

  - Correct any errors and restart using the appropriate command with the debug option.

If you do not have an event in the eventout file, but the RAD Application was invoked (e.g., a problem was opened and an e-mail event was not created), check the following:

- Please review the *Basic Trouble Shooting* and *Using Format Control to Send e-mail Messages* sections of the ***Event Services Guide***

If you do not have an event in the eventin file, but the external SCAuto Application event has occurred (e.g., mail was sent but not in ServiceCenter), check the following:

- Is there connectivity to the SCAuto Base server?
  - Review the first item in this trouble shooting section.
- Manually send and receive mail using the mailbox assigned in the **mailbox** parameter (e.g., *scmail -mailbox:user1*).
  - If manual functions fail, check with your systems administrator.
- Is the scmail application active?

  If it is *not running*, check the logfile (standard out by default) for errors.
  - Correct errors and restart using the appropriate command with the debug option. The debug option allows you to determine if the problem is an scmail application failure or a product installation or specification failure.
  - If restart fails using debug option, review the application *logfile* for any new error message information.

  If it *is running*, check the SCAuto application logfile for errors.
  - Correct errors and, restart using appropriate command with the debug option if available.
  - Recheck the *logfile* and correct errors and retry.

If you need assistance in the resolution of your problem, please have the following available before calling for support:

- Core files.
- Error logs produced by mail service and SCAuto application with the debug option(scmail).
- Copy of mail that is failing or ability to recreate.
- ServiceCenter log and Scheduler log(s), if specified.
- ServiceCenter msglog for affected applications or services.
- ServiceCenter print of agent status (Event Services Menu).
- Unloaded Event records relevant to errors.
- Unloaded filter specifications, if relevant.

Chapter 5 Customizing SCAuto Mail

This chapter explains how to customize SCAuto Mail using the event mapping files contained in the directory:

\Program Files\Peregrine Systems\SCAuto\Mail\EventMap\ToSC\

Concepts

SCAuto Mail uses external event mapping control files to define how a particular ServiceCenter event should be constructed from your e-mail. This chapter explains how these files work and how to customize them to meet your site's requirements.

For each type of ServiceCenter event there must be at least one *.map* file pointed to by the *event.ini* file in the ***\EventMap\ToSC*** subdirectory for SCAuto Mail.

The *.map* files in the *\EventMap\ToSC* directory tree define how one or more ServiceCenter events going TO ServiceCenter should be constructed from your e-mail.

The purpose of a *.map* file is to tell SCAuto Mail how to construct a particular type of ServiceCenter event transaction and, optionally, specify one or more conditions which control event generation.

Note: SCAuto Mail's *event mapping* function discussed here is not the same thing as the ServiceCenter *Event Services event mapping* function.

Both types of event mapping definitions are used together. The event mapping function within SCAuto Mail is used to generate and emit the event transaction which is sent to ServiceCenter. The event mapping within ServiceCenter Event Services, which is documented in the ***Event Services Guide***, is used to interpret the event transaction once it arrives in ServiceCenter's event input queue.

How Event Mapping Works

Event map directory structure

All of the files that pertain to event mapping are kept in the SCAuto Mail directory *\EventMap\ToSC*. The directory *EventMap* is named by the **event_map_dir** parameter in the scmapi.ini file.

Under the SCAuto Mail\EventMap\ToSC subdirectory, a Windows profile named *event.ini* contains all the ServiceCenter events that you want to constructed from your e-mail and the corresponding map files. Following is a sample of the *event.ini*:

```
   [EMAIL ]

pmo = Problem\pmo.map
cm3rin = Change\cm3rin.map
```

event.ini contains only one section that is [EMAIL]. The left-hand side of the statement `pmo = Problem\pmo.map` defines the ServiceCenter transaction (event) type, i.e. pmo. The right-hand side is interpreted as a relative path to the event mapping file which contains the instructions for generating a *pmo* event from the body of an e-mail message. You can put all the map files in one directory or, like the sample, in different directories. All the directories need to exist under the ToSC directory.

Attributes

An *attribute* is a tagged value in the body of an e-mail message. A tag is a line starting either with *SC_* or *EVENT_* followed by a colon(:) and a value in the body of an e-mail message.

```
SC_category:software
SC_Severity:1
```

In order to generate a ServiceCenter event, the body of an e-mail message has to contain an named “attribute” which appears in the *.map* file. The names of attributes are totally defined by the user as long as they start with either *SC_* or *EVENT_*.

There are a few tags with special meaning which have been kept for backward compatibility with e-mail bodies formatted for use with earlier versions of SCAuto Mail. These tags are *SC_type*, *SC_event*, *SC_var* and *SC_user*.

Note: For compatibility with PNMS, these tags may be called *PNMS_*.

Important: Pay close attention to spelling and case when coding an attribute specification in a *.map* file. You must code the attribute name exactly as it would appear in a body of an e-mail.

Event generation

When an e-mail arrives in SCAuto Mail's mail box, SCAuto Mail does the following:

1. Reads through the body of an e-mail message, parses, and saves all the *tags* and *values* if there are any.

   ```
   SC_category:software
   ```

2. The event.ini file is consulted. A sample *event.ini* file would be:

   ```
       [EMAIL ]

   pmo = Problem\pmo.map
   cm3rin = Change\cm3rin.map
   ```

   The [EMAIL] section of the *event.ini* file is examined. If no [EMAIL] section exists, then no event is generated for this e-mail. Each line in the section [EMAIL] defines a ServiceCenter event type to be generated, if possible, from the e-mail, together with the event mapping file which describes how the event is to be constructed.

   For example, with the *event.ini* file shown above, an event mapping file called *pmo.map* is consulted in an effort to generate a ServiceCenter transaction of type *pmo*.

3. The *.map* file is consulted. *.map* files contain rules (expressions) governing the generation of the event transaction. SCAuto Mail replaces all the matched attributes in the *.map* file by the saved tagged values, format and generate a ServiceCenter event according to the rules that defined in the *.map* file.

Note: An e-mail together with a given *.map* file may or may not produce a ServiceCenter event transaction. At least one rule must evaluate to TRUE or the event will not be generated.

Note: You can generate multiple events from a single e-mail. The event type to be generated from a particular e-mail has traditionally been specified by coding one of the historical tags SC_type:, SC_event:, or PNMS_event:. If one of those tags is present in the e-mail body, then SCAuto Mail will only look at event.ini entries for the specified event type. But if none of these historical event type tags appears in the body of the e-mail, then SCAuto Mail will visit each and every *.map* file whose name is specified in the [EMAIL] section of the *event.ini* file, and evaluate each of these, trying to generate that type of event. Whether this succeeds or not for a particular event type depends on whether the expression(s) in the *.map* file evaluate to TRUE or not. What appears in the particular events depends on what tags and values appear in the e-mail body, and what tags appear in the various *.map* files.

A sample .map file

Here is a sample *.map* file, named *pmo.map*. This file is used to generate *pmo* events from an e-mail.

```
# # PMO.MAP
#
# This is for PMO events going TO ServiceCenter Problem Management
# being created from email

SC_priority != ""
SC_description != ""

# logical name
SC_logicalname^

# network name
SC_networkname^

# reference no
SC_reference^

# cause code
SC_causecode^

# description
SC_description + $BODY^

# action,2 (unused)
^
```

```
# action,3 (unused)
^

# network address
SC_networkaddr^

# type
SC_type^

# category
SC_category^

# domain
SC_domain^

# objid
SC_objid^

# version
SC_version^

# model
SC_model^

# serial no
SC_serialno^

# vendor
SC_vendor^

# location
SC_location^

# contact name
SC_contactnm^

# contact phone
SC_contactphone^

# resolution
SC_resolution^

# assignee name
SC_assignee^

# priority: C – Critical, H – High , N – Normal, L - Low
TRANSLATE(SC_ priority, “CHNL”, “1234”) ^

# failing component
SC_component^
```

```
# system
SC_system^

# End of file
```

How .map files relate to ServiceCenter events

ServiceCenter event transactions consist of a header portion followed by a series of data fields separated by a separator character, which by default is the caret symbol **^**. Currently, SCAuto Mail requires the use of this symbol as the separator character.

SCAuto Mail *.map* files are concerned solely with defining the data fields portion of event transactions. This portion is called *EVFIELDS*. These data fields correspond positionally to fields defined within the ServiceCenter event map for the event type in question, for example: *cm3rin*, *pmo*, etc.

The header portion of the event transaction is generated without reference to anything in the *.map* file. The event type section of the header is taken from the *event.ini* line which points to *.map* file.

Most of the lines in an SCAuto Mail *.map* file cause the attribute data to be emitted to EVFIELDS which is constructed as the file is sequentially processed. The exceptions are white space lines, comment lines, and event generation rules, which take the form of relational expressions.

How .map Files are Processed

This section provides a basic overview of the structure of *.map* files.

Types of lines found in .map files

There are several types of lines in *.map* files, including blank lines, comments, expressions, and event field definitions.

Blanks and # comments

In processing a *.map* file, SCAuto Mail ignores any blank lines or lines starting with the **#** character, which indicates a comment.

Event generation expressions

Apart from blank lines and comments which are ignored, lines in a *.map* file must either be event generation expressions or definitions of event fields. Event generation expressions are evaluated but do not generate event field data, regardless of where they appear in the file.

If a line contains a relational operator such as =, !=, <, >, <=, >=, then it is an event generation expression which evaluates to TRUE or FALSE. At least one such expression must evaluate to TRUE for the event to be generated, but a given rule evaluating to FALSE does NOT prevent the event from being generated. In other words, the results of each expression are logically ORed (connected through an *or* statement) together.

Event generation expressions can be present anywhere in the file but it makes sense to place them at the top, as is shown in the example. Note that the lines in the sample *.map* files containing # *Start of Event Generation Section* and # *End of Event Generation Section* are comments, they do not have any special significance.

Event field definitions

Any line in a *.map* file which is not an expression and is not a comment or white space is interpreted as an event field specification. These cause the attribute data to be emitted to the next field of the buffer which is used for the construction of the event.

Event fields are defined in terms of attributes and/or literal data, and are terminated with the separator character **^**. Event fields can also be empty, i.e. consist only of the separator character.

Positional nature of event field definitions

Each line of the *.map* file which is not a rule or a comment or blank corresponds positionally to the next field of the event specification, as defined in the ServiceCenter database dictionary. A *pmo* or *cm3rin* or other event has certain fields, in a particular order, each having a particular meaning. SCAuto Mail has no intrinsic knowledge of these formats or their meaning. It simply constructs a buffer in accordance with the specifications found in a particular *.map* file and sends it. Therefore you must refer carefully to the ServiceCenter specification for a particular event while customizing a *.map* file.

Starting at the top of the *.map* file, each line in the *.map* file that defines event data to be emitted must correspond to the next sequential field in the event to be generated.

Empty fields

All event field specifications in a *.map* file must be terminated with the ^ separator character. Event field specifications which contain only ^ cause empty event fields. This is used when there is no attribute which corresponds to a particular ServiceCenter event field.

Operators

There are two operators used in event field definitions; the choice operator and the concatenation operator. These are distinct from the relational operators found in event generation expressions.

Choice (,) This operator allows you to list several attributes separated by commas for a single event field specification. This means that the first one of these attributes which is found in the e-mail and is not null is chosen and used in the event.

Concatenation (+)

The + operator functions as a concatenation operator, allowing you to paste multiple attributes together. For example:

SC_dategen+SC_Timegen+SC_text

This explicit form of concatenation is much more limited than the implicit form of concatenation discussed earlier, because it is limited to joining the attributes. Unlike implicit concatenation, it cannot be used to concatenate literals with the attribute values. It also inserts a blank between the concatenated values, which implicit concatenation does not do.

Built-in functions

Data may be enclosed in a function specification, such as *ANY( )*, *ALL( )*, *NTH( )*, *SUBSTR( )*, *TOUPPER( )*, *TOLOWER( )*, or *TRANSLATE( )*. These functions are discussed next.

Note: Only very limited nesting of functions is permitted. In particular, nesting of two functions which both contain commas and sub-parameters, e.g. SUBSTR( ), NTH( ), and TRANSLATE( ), is *not* supported, but nesting of the simpler functions which do not have parameters is permitted. Nesting of functions wherein only one function with parameters occurs is also permissible, for example, ANY(SUBSTR(....

AFTER( ) Example: AFTER(SC_logicalname, “\\”) would evaluate to “Jsmith” if SC_logicalname contained “SERVER_SD\Jsmith”. Two backslashes are required because \ is the escape character.

ALL( ) ALL( ) allows attributes which may occur multiple times to be concatenated into a ServiceCenter *array*. This function concatenates the data with **|** symbols which denote *array* to ServiceCenter event services.

Example:

In the pmo.map file replace the description line by:

```
# description
#SC_description^
ALL(SC_description)^
```

In an email, we have:

```
SC_type:pmo
SC_priority:1
SC_description:Line1-this is a test, we will have 4 lines,
SC_description:Line2-all this lines go to the
SC_description:Line3-description.
SC_description:Line4-End of message.
```

In this case, a pmo event will be created in the ServiceCenter’s eventin queue and all these 4 description lines will appear in the ServiceCenter’s Problem Details window.

ANY( )	The ANY( ) operator can be used in rule expressions to test if any instance of a multiply valued attribute which occurs repeatedly in the body of an e-mail equals some value.
BEFORE( )	This is like AFTER(), but obtaining the substring that precedes the specified substring.
NTH( )	NTH( ) causes selection of a specific instance of an attribute which may occur multiple times. If the instance doesn't exist, no data is emitted by the specification.
SUBSTR( )	The SUBSTR operator allows you to take a substring of an attribute value. Results of application of SUBSTR to other than string attributes are not well-defined. The first argument following the attribute name is the starting offset and the second argument is the desired length. SUBSTR( ) offsets are zero-relative. The length specification may be the special value * which means *the rest of the field*.
TOUPPER( )	The TOUPPER operator forces the data for the enclosed an attribute to upper case. For example, you could code: *TOUPPER(SC_logicalname)* if you wanted to get values like SERVER_SD\JSMITH in upper case
TOLOWER( )	The TOLOWER operator forces the data for the enclosed an attribute to lower case. For example, you could code: *TOLOWER(SC_logicalname)* This is coded to get values such as server_sd\jsmith all in lowercase
TRANSLATE( )	The TRANSLATE( ) function is used to convert all occurrences of a specific character within an attribute to a different value. For example, TRANSLATE(SC_priority, "CHNL", "1234") would convert SC_priority of "C" to "1, "H" to 2", and so forth.

Special variables

There are five special variables, $TO, $FROM, $SUBJECT, $BODY, and $DATE. You can add these variables to the *.map* file in the desired event field definitions.

Example:

In the pmo.map file replace the description line by:
SC_description + $DATE + $BODY^

In an email, we have:

SC_type:pmo
SC_priority:C
Line1-this is a test, we will have 4 lines,
Line2-all this lines go to the
Line3-description.
Line4-End of message.

In this case, a pmo event will be created in the ServiceCenter's eventin queue and the Problem Details is:

Thu, 30 Aug 2001 10:12:01 -0700
Line1-this is a test, we will have 4 lines,
Line2-all this lines go to the
Line3-description.
Line4-End of message.

The email date is saved in $DATE and added to the Problem Details. The email message – Line1…Line4 have no tag, so they all go to the $BODY and added to the Problem Details too. Because, we do not have a SC_description line in the email, so the email date is the first line of the Problem Details.

Restrictions

- Expressions cannot be arbitrarily complex, they must be simple relational tests against a particular an attribute which is expected to be found in the boy of an e-mail. An example would be SC_SEVERITY `!= ""` which requires a non-null value for the serverity. Expressions can contain only relational operators used to test a given an attribute value.
- Built-in functions such as SUBSTR, TOLOWER, TOUPPER etc. currently cannot be used together with special variables.
- Only very limited nesting of built-in functions is permitted. In particular, nesting of two functions which both contain commas and sub-parameters, e.g. SUBSTR( ), NTH( ), and TRANSLATE( ), is not supported.
- The separator character must be **^**.

- Expressions cannot be split between multiple lines. Continue the line to the right as far as is necessary to code the expression. There is no continuation character defined.
- If a literal string is coded as part of an event field definition, it must be enclosed in double quotes, and may not contain embedded double quote characters.

Format of .map Files

This section contains a more detailed discussion of the format of *.map* files.

The format of a .map file is as follows:

<item> <separator> newline

Separator character

The separator character is always **^**. The current event sub-field is terminated when the separator character appears.

Implicit concatenation

Multiple *.map* file lines can contribute to a single EVFIELDS sub-field if desired, by not coding the ^ character until the end of the last *.map* file line for that EVFIELDS sub-field. This allows multiple items to be concatenated into a single EVFIELDS sub-field. This form of concatenation is implicit. There is an explicit concatenation operator, +, also provided (discussed later).

Null items

An item may be a NULL item, in which case the line is either completely blank, or only the separator **^** character appears. Completely blank lines are ignored. Lines with only the separator character cause an empty EVFIELDS sub-field to be generated.

Comments

An item may be a comment starting with **#**, in which case the line is ignored, just as though it were completely white space. Any separator character at the end of a comment is ignored, since comments do not cause EVFIELDS data to be emitted.

Event generation expressions

An item can be a relational expression, in which case it functions as a rule controlling whether or not the event should be generated. If the expression evaluates to TRUE, the event will be generated. Multiple such expressions may be coded. If any single one evaluates to TRUE, the event will be generated.

Expressions exist only to control event generation. They do not cause any EVFIELDS data to be emitted, regardless of where they may occur in the *.map* file. As with comments, any separator character at the end of an expression is ignored.

For example:

```
SC_priority != ""
```

As long as a tag called SC_priority appears with a value in the e-mail body, the event will be generated.

Expressions cannot be split between multiple lines. Continue the line to the right as far as is necessary to code the expression. There is no continuation character defined.

Literal strings

An item can be a literal string enclosed within double-quotes. No double-quote characters may appear within the string, as there is no escape character currently defined. The characters between the double quotes are emitted into the current EVFIELDS sub-field.

Chapter 6 SCAuto FAX Server

Overview

SCAuto provides a FAX Server, allowing ServiceCenter to interface with an external fax system. Faxes can be generated from within ServiceCenter RAD applications as an adjunct to email or printing. The central component to this functionality is the standard ServiceCenter Event Services **fax** event.

The FAX Server uses an external command file to generate the necessary fax delivery commands. This file may be modified by the user in order to utilize different fax systems. The default command file, however, assumes the LanFax version 5 system from Alcom is used on Windows NT systems and the Replix fax management system from SoftLinx Inc. on UNIX systems. The figure below illustrates the SCAuto FAX Server.

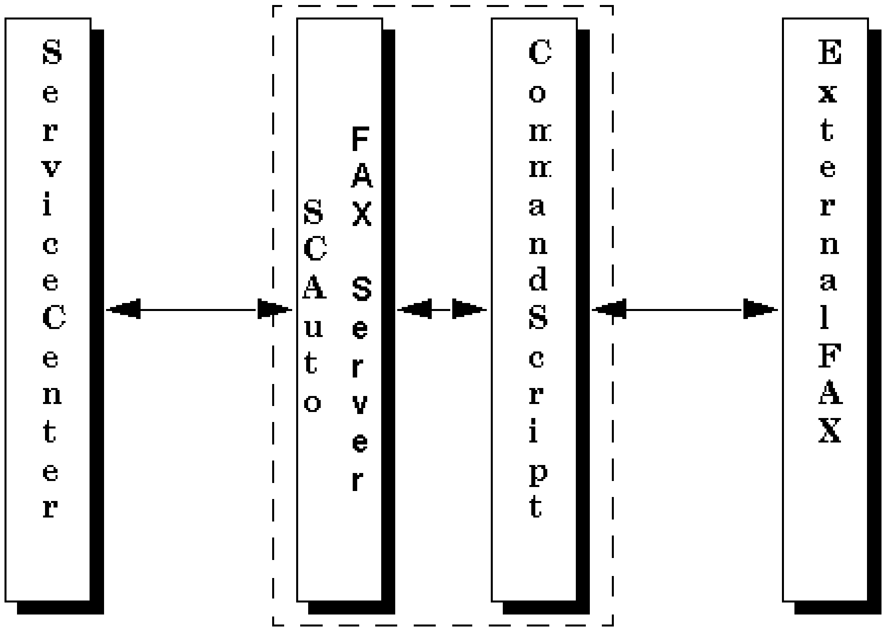


Figure 6-1. SCAuto Fax Server

Components

SCAuto FAX Server is equipped with the following components:

- **scfax** (*scfax.exe* for Windows NT)
  - The FAX Server executable.
- **scauto.msg**
  - Contains messages printed by the Fax Server and other SCAuto interface tools. This is optional, and may be customized for use with languages other than English.
- **scauto.dll** (Windows NT only)
  - The SCAuto run-time library. This can be shared by all SCAuto products.
- **fax.sh**(UNIX) or **fax.cmd** (Windows NT)
  - Sample command script. This may be modified for use with other FAX systems.

The following components are also included for integration with Alcom's LanFax system (Windows NT only):

- **lanfax.exe**
- **lfs32_50.dll**

Operation

To start the FAX Server, give a command in the following format:

scfax -server:<host.service>

The *scfax* command string options are as follows:

-server:<host.service>

Host name and service name of the SCAuto Base Server. This should be the same as the **scauto***:* parameter used by the *scautod* server, in the *host.service* format. The default is *scauto* or *12690* if that service does not exist.

-command:<name>

Name of the command to execute for each fax event. This command defaults to *fax.sh* (or UNIX) or *fax.cmd* (Windows NT) in the same directory as the **scfax** command.

-log:<file> Name of file to use for logging and error messages. This defaults to *scfax.log* on Windows NT, and standard error output for UNIX.

-debug: 1: Turns on debugging mode (command is printed before executing, and events are not deleted after processing).

-sc_reconnect: Allows SCAuto Applications to retry to connect to ServiceCenter when ServiceCenter is restarted.

1: Retry connecting to ServiceCenter.
0: Do not retry to connect to ServiceCenter.
Default is 0.

This This parameter works together with - reconnect_interval and - numberof_retry_connect.

This parameter impacts the K (kill) command in ServiceCenter's system status list. The K command closes the connection between ServiceCenter and SCAuto Fax. When this parameter is not set to 1, SCAuto Fax checks the connection and if the connection is broken, then SCAuto Fax exists. When this parameter is set to 1, SCAuto Fax sees a broken connection but has no knowledge if it is caused by the K command or ServiceCenter shutdown. So, SCAuto Fax tries to reconnect even after a K command.

-reconnect_interval:<n>

Number of seconds to sleep between retrying. Default is 120 (two minutes).

-numberof _retry_connect:<n>

Number of retries before exiting. Default is 10.
0 = retry forever.

When running, *scfax* retrieves and processes *fax* events from ServiceCenter. After processing, the events are deleted. Each event is parsed, and the command specified with **-command** is called for each event, passing the *fax* event fields as arguments.

Under UNIX, the SCAuto fax server uses the Replix fax management system from SoftLinx Inc. by default. Under Windows NT, the LanFax system from Alcom is used by default. Other fax systems will be supported in the future. Please refer to the documentation that came with the fax management system for information about installation and usage.

Note: For the Replix fax system, the default fax program option (after -*command*) will be the **rpxsend** command. The full location must be given if it is not in the current path when **scfaxd** is run. There is no default, because there is no standard location for this file.

Note: For the LanFax system, the fax program option will use the *lanfax.exe* program, which should be in the current directory, along with the *lfs32_40.dll* (both part of the SCAuto fax distribution). *Lanfax.exe* is for LanFax version 5.

When a fax is sent from Service Center, the fax number is looked up in the contacts database, and the name, company name, and voice phone number in the contacts database will be used on the cover page.

Customization

If the LanFax or Replix paging systems are not being used, or it is desired to process events differently than the default, the FAX Server may be customized. Most importantly, the parameter passed with the **-command** option can specify any command to be called, not just the supplied command files. The *fax.sh* or *fax.cmd* may be changed to call any external command, and is commented so as to make this process easier. If customizing the fax server, it is very helpful to use the **-debug** option while testing.

The most important item to note is that there are several arguments passed to the external command file that correspond to the ServiceCenter *fax* event fields. Each argument is quoted, as it may have embedded white space (with single quotes for UNIX and double quotes for Windows NT). Many of these arguments may be blank. The arguments in order are:

1. **File name** containing FAX body. This is a text file. This file is temporary, and deleted when the external command returns. If you need to keep it, make a copy.
2. **FAX phone number** to use.
3. **Name of operator** sending the FAX.
4. **Subject** for the FAX.
5. **Name** of FAX recipient.
6. **Company** recipient belongs to.
7. **Voice phone number** of recipient.

The last five fields are intended for use on the FAX cover page, and may be blank.

FAX event

The FAX event is composed of seven fields. These are:

1. **Name** of FAX sender.
2. **Company** of FAX recipient.
3. **Subject** of FAX.
4. **Name** of FAX recipient.
5. **FAX phone number**.
6. **Voice phone number** of FAX recipient.
7. **Body** of the FAX.

Note: These may not be in the same order as the external command expects them.

Trouble Shooting

If connection fails between the SCAutomate Base (scautod) and the SCAutomate Application (scfax) please check the following common failures:

- Is scautod running on ServiceCenter Server Platform? Use ServiceCenter *status* command to check. If it's ***not running*** start it from the status option *start schedulers* selecting *scauto.startup*. If the start fails the scautod daemon logs to the *sc.log* file, so be sure to specify and check the log for any error messages.Check the TCP/IP specifications for your platform, a service name other than that specified for ServiceCenter server is required, and the *scauto:* keyword should be specified in the *sc.ini* file must match this name. If it ***is running*** check the service name specified in *the sc.ini* file on the ServiceCenter platform for the *scauto:* keyword. If the keyword is not specified scautod defaults to the services name *scauto*. This will be useful in the next step.
- Are the TCP/IP specifications correct on the SCAuto UNIX or Microsoft Windows platform? Check the host and service specification, are they specified and do they ***match*** the *host* and *service* name of s*cautod?* Specifications may be made in local files (UNIX /etc/hosts, /etc/services) or a name server may be used. Check with your network administrator for specification. Also, check your specification in the program, configuration file, or passed parameters and make sure they all match (ex.scfaxd -server **host.service**).
- Is there connectivity between SCAutomate Base (server) and the SCAutomate Application (axfaxd)? *Ping* the SCAutomate Base (scautod) platform from the SCAutomate application (client) platform. If this fails and all TCP/IP specifications are correct contact your network administrator for assistance.

If you have an event in the **event.out** ***file (Output Events section of the Event Services Guide) but the external SCAutomate Application is not invoked (ex.fax event and fax not sent) check the following***:

- Is there connectivity to the SCAutomate Base server? Review ***connection between the SCAutomate Base (scautod) and the SCAutomate Application (client)*** trouble shooting section.
- Is SCAutomate application (scfaxd) active? If ***it's not running*** check the SCAutomate scfaxd *logfile* for errors (*scfax.log* is the default). Correct errors and, restart using appropriate command with the debug option. The debug option will allow you to determine if the fax operation was attempted and the service product (Replix, LanFax) failed, or if it's an SCAutomate application failure. If restart fails review the SCAutomate application *logfile.* If the fax operation was attempted and failed, check the log file for

the fax product, if any, for errors or messages. SCAutomate applications in debug when appropriate log the command issued to the service product. If the command is present, issue the command manually. If it fails correct the problems with the service product and retry. If ***it's running***, check the SCAutomate application *logfile* for errors. Correct errors and, restart using appropriate command with the debug option if available. Recheck the *logfile* and try commands manually as shown with the -debug option.

If you do not have an event in the* event.out *file (Output Events section of the Event Services Guide) but the external SCAutomate Application is not invoked (ex.fax event and fax not sent) check the following:

Review the specific RAD application and how fax generation is implemented (ex. PM msgclass with a special device definition). Review the **Basic Trouble Shooting** and **Using Format Control to Send FAX Messages** sections of the ***Event Services Guide***.

If you need assistance in the resolution of your problem, please have the following available before calling for support:

- Core files
- Error logs produced by server application(ex. REPLIX) and the SCAutomate application with debug option(scfax).
- ServiceCenter log and Scheduler log(s) if specified.
- ServiceCenter msglog for affected applications, or services.
- ServiceCenter print of agent status(Event Services Menu).
- Unloaded Event records relevant to errors.

Chapter 7 SCAuto Paging Server

Overview

The SCAuto Paging Server interfaces ServiceCenter with an external paging system. Through this connection ServiceCenter automatically generates pages in response to RAD controlled internal events (e.g. open problems).

ServiceCenter Event Services includes a standard *page* event and an *scauto.page* application which generates the *page* events. Please see the ***Event Services Guide*** for syntax and usage of this event and application.

The standard *page* event information includes an alphanumeric message, as well as a numeric message, used for numeric-only pagers. Paging information retrieved by the paging server from the ServiceCenter *contacts* database is converted into an external pager command. The paging server uses an external command file to generate the paging command. A user can modify this file to utilize many different paging systems. (The default command file assumes the TelAlert system by Telamon will be used.)

The figure below illustrates the SCAuto Paging Server.

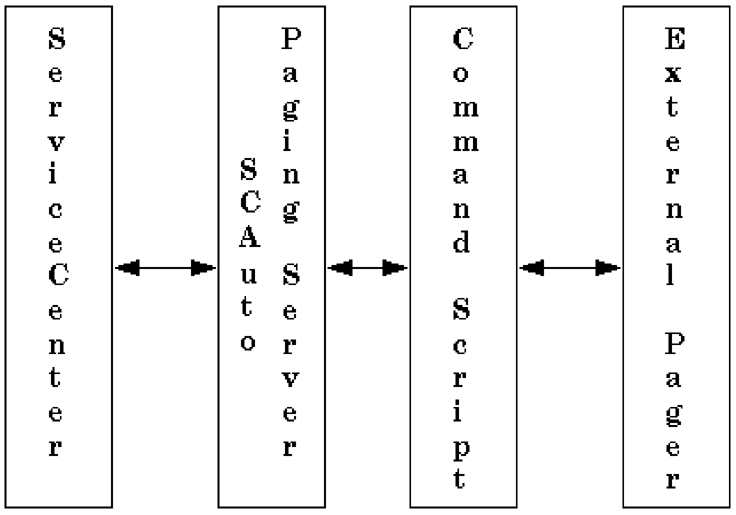


Figure 7-1. SCAuto Paging Server

The SCAuto Paging Server also provides support for two way paging, if the paging system used provides this capability. For example, with SCAuto two way paging, a problem ticket can be updated through a return reply to a page.

To support this option the paging system must be able to run a program or command script when a page reply is received. The TelAlert system provides this capability. A sample script and the settings required for this capability are provided with the SCAuto Paging Server.

Components

SCAuto Paging Server comes with the following standard components:

- **scpager** (*scpager.exe* for Windows NT)
  - The Paging Server executable.
- **scauto.msg**
  - Contains messages printed by the Paging Server and other SCAuto interface tools. This is optional, and may be customized for use with languages other than English.
- **scauto.dll** (Windows NT only)
  - The SCAuto run-time library. This can be shared by all SCAuto products.

The following components are used for integration with Telamon's TelAlert system. These components may also be modified for use with other paging systems.

- **pager.sh** and **notify.sh** (UNIX only)
  - Sample shell scripts.
- **pager.cmd** and **notify.cmd** (Windows NT only)
  - Sample command scripts.
- **SCenter.ini**
  - Sample initialization sections to support two way paging.
- **spawn.exe** (Windows NT only)
  - Helper program to simplify integration with paging servers.

Operation

To start the Paging Server, use the following command string (inserting your specific host.service, ID, and command):

scpager -server:<host.service>

The *scpager* options are:

-server:<host.service>

Host name and service name of the SCAuto Base Server. This should be the same as the **scauto:** parameter used by the *scautod* server, in the *host.service* format. The default is *scauto* or *12690* if that service does not exist.

-command:<name>

Name of the command to execute for each page event. If the *TELALERTCFG* environment variable is set, then this command defaults to *ServiceCenter/pager.sh* (for UNIX) or *ServiceCenter\pager.cmd* (Windows NT) in the TelAlert directory.

-log:<file>

Name of file to use for logging and error messages. If not specified, messages will be printed to the screen (on UNIX) or *scpager.log* (Windows NT).

-debug:

1: Turns on debugging mode (command is printed before executing, and events are not deleted after being processed).

-noerror:

1: Prevents creation of error events.

The following options are used in conjunction with two way paging.

-input:<file>

Name of a file to be monitored for incoming events. (See the section on *two way paging* for details).

-checkpoint:<file>

Name of a file to use for checkpointing the input event file. This defaults to *pager.chk*.

-sc_reconnect:

Allows SCAuto Applications to retry to connect to ServiceCenter when ServiceCenter is restarted.

1: Retry connecting to ServiceCenter.
0: Do not retry connecting to ServiceCenter.
Default is 0.

This parameter works together with -reconnect_interval and - numberof_retry_connect.

This parameter impacts the K (kill) command in ServiceCenter's system status list. The K command closes the connection between ServiceCenter and SCAuto Pager. When this parameter is not set to 1, SCAuto Pager checks the connection and if the connection is broken, then SCAuto Pager exists. When this parameter is set to 1, SCAuto Pager sees a broken connection, but has no knowledge if it is caused by the K command or ServiceCenter shutdown. So, SCAuto Pager tries to reconnect even after a K command.

-reconnect_interval:<n>

Number of seconds to sleep between retrying. Default is 120 (two minutes).

-numberof _retry_connect:<n>

Number of retries before exiting. Default is 10.
0 = retry forever.

When running, *scpager* will retrieve *page* events from ServiceCenter and process them. After processing, the events are deleted. Each event is parsed, and the command specified with **-command** is called for each one, with the *page* event fields passed as arguments.

If the TelAlert system is being used, the supplied *pager.sh* or *pager.cmd* command file will be used to process each event appropriately for TelAlert.

If the *-input* option is being used, *scpager* will also monitor that file for incoming events to send to ServiceCenter. This function is utilized for two way paging.

Sending a Page

There are many different types of pagers, and different protocols used to send pages. Thus it is usually necessary to know more than just a phone number to send a page. In particular, a pager type usually needs to be known. It is strongly recommended that the paging system being used be installed and tested before trying to integrate it with ServiceCenter. Knowing how to send a page outside of ServiceCenter simplifies knowing what values are needed within ServiceCenter.

In the *contacts* records in ServiceCenter, there are fields for *pager number*, *name*, *group*, *type*, *PIN*, and *mailbox*. These fields can all passed to the ServiceCenter *page* event.

With the TelAlert system, all of these fields have their uses.

name	Name of person to page. If this is specified, then none of the other fields are necessary.
group	Name of group of people to page. If this is specified, then none of the other fields are necessary.
type	Type of pager. If this is specified, then the following three fields may be used.
number	Phone number of pager.
PIN	PIN number to use if necessary.
mailbox	Voice mail box number to use if necessary.

The *name*, *group*, and *type* values must be previously defined within TelAlert's configuration file (*telalert.ini*). The *name* field is intended for cases where TelAlert is configured with enough information to send the page with no additional information (in the *[Destinations]* section of *telalert.ini*). The *group* field is intended for sending multiple pages, with possible escalation (in the *[Groups]* section of *telalert.ini*).

Note: The value of *Speaker* may be put in the *name* field, and this will cause TelAlert to speak aloud the alphanumeric message using its speaker.

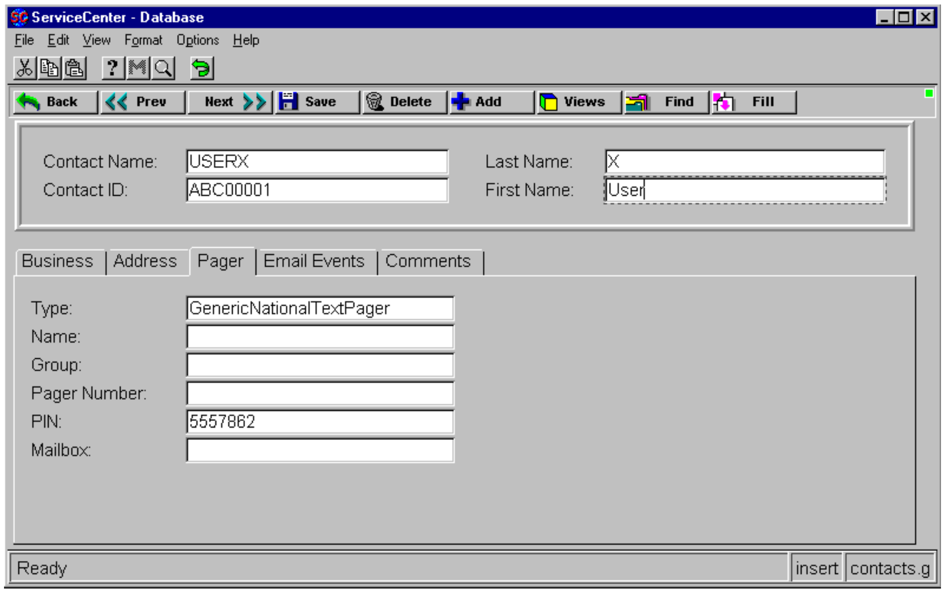
The *type* field is used when you have not pre-configured TelAlert to know about individual users and groups (types are defined in the *[Configurations]* section of *telalert.ini*). Many different types of pagers are already configured in TelAlert by default, such as *Pager*, *PagNetNationalTextPager*, *SkyTelTwoWayTextPager*, etc. These are defined in the *Pagers* sub-directory of TelAlert.

If you know the correct command to send a page from a command line using TelAlert, then that same information can be used in the *contacts* records.

Different paging programs may use a different subset of this information. See the *Customization* section that follow for details on how to modify the behavior of the Paging Server.

Note: Simple numeric pagers very often have poor reliability when paged from modems. This is because the number dialed is usually intended for use by humans, and the modem has to rely upon blind dialing without receiving confirmation of success. Better reliability is achieved by using the pager service provider's text pager service instead, which can usually page numeric pagers as well. For instance, instead of using *GenericPager* with phone number 800-555-7862, you can use *GenericNationalTextPager* with PIN number of 5557862 (*Generic* is just an example).

For example, here's a sample contact record:



Contact Name: USERX
Contact ID: ABC00001
Last Name: X
First Name: User

Type: GenericNationalTextPager
Name:
Group:
Pager Number:
PIN: 5557862
Mailbox:

Figure 7-2. Pager tab on Contacts Record

Customization

If the TelAlert paging system is not being used, or it is desired to process events differently than the default, customizing the Paging Server is simple to accomplish. Most importantly, the parameter passed with the *-command* option can specify any command to be called, not just the supplied command files. The *pager.sh* or *pager.cmd* scripts may be customized as well, and contains comments throughout, which make this process easier. If customizing the Paging Server, it is very helpful to use the *-debug* option while testing.

The most important item to note is that there are several arguments passed to the external command file which correspond to the ServiceCenter *page* event fields. Each argument is quoted, as it may have embedded white space (with single quotes for UNIX and double quotes for Windows NT). Many of these arguments may be blank. The arguments, in order, are:

1. **Vendor name**- Can be ignored, but should be set to **telalert** or the name of the system being used. Thus, scripts can check this argument to determine which paging system to use.
2. **Name of person** to page.
3. **Name of group** to page.
4. **Type of pager** (If this is specified, at least one of the next 3 arguments must be specified.)
5. **Phone number** of pager.
6. **PIN number** of pager.
7. **Voice mail box** number.
8. **Numeric message** (Intended for numeric-only pagers.)
9. **Text message** (Intended for alphanumeric pagers, voice mail, or any other device handling alphanumeric text.)
10. **Reply event**- The event type to use for errors or replies involving a two way page. See the section on *two way paging* for details.
11. **Reply ticket number -** Extra information to use with the reply event type.

The last two fields are optional, and are only used when a reply to a page is expected (see the following section on *two way paging* for details).

Note: If TelAlert is being used, only one of *name*, *group*, or *type of pager* should be specified. These arguments correspond to TelAlert's *-i*, *-g*, and *-c* arguments.

Note: The external command must specify how the pages are to be processed. For instance, the TelAlert system can interface with voice mail, alarms, and many other messaging formats.

Two Way Paging

The SCAuto Paging Server can monitor a file for incoming events, a capability which allows a two way paging system to communicate with ServiceCenter once a reply to a page has been received. Any event could be put into this file, but it is intended for page replies only.

Several files are included for supporting two way paging with the TelAlert system. The rest of this section discusses integration of two way paging into TelAlert. (If you are not using TelAlert, or do not use two way paging, this section may be skipped.)

The *SCenter.ini* file contains two subsections to add to the *telalert.ini* initialization file. These should be added to the *Notifications* and *Responses* sections as follows:

```
[Notifications]
    .
    .
    .
{ServiceCenter}
Active=True
Shell=""
Command=${TelAlertCfg} ...
Reply=$(TimeStamp) ...

    ...

[Responses]
    .
    .
    .
{ServiceCenter}
NumReplies=5
Acked=1
AckedHold=3,5
NotAcked=2
Released=4
Reply1=Yes
Reply2=No
Reply3=Hold
Reply4=Release
Reply5=Info
    .
    .
    .
```

Note: For Windows NT, an extra *spawn.exe* file is used to call *notify.cmd*. This is done only to simplify the added notifications section.

You must reinitialize TelAlert after changing the *telalert.ini* file.

Of course, these sections do not have to be used as is, and you will want to modify the responses section to suit your specific needs. The supplied *pager.sh* and *pager.cmd* files however assume these section names by default.

At the top of the *pager.sh* and *pager.cmd* files are variables which control how TelAlert is called. You may wish to review these variables and give appropriate values for your site, if you decide to make changes to the default file formats.

When a reply to a page is received, TelAlert will execute the *notify.sh* or *notify.cmd* files. By default, these will create events in a file called *event.in* in TelAlert's *ServiceCenter* directory.

Customizing two way paging

The standard ServiceCenter *page* event has two optional fields used for page replies. If these two fields are present in the page event, it is assumed a reply to the page is wanted. The external command should remember these two fields so they may be passed back to ServiceCenter. These optional fields are:

- *Reply event type* - this is the type of event created in ServiceCenter.
- *Reply ticket number* - this is extra information identifying this page once it has been received by ServiceCenter.

When a reply to a page is received, an event with a specific format is created and placed in the input event file which the Paging Server is monitoring. This is handled by a shell script or command files (such as the *notify.sh* included for use with TelAlert).

(The ***Event Services Guide*** provides information about the format for events in the *input* file.)

The format of the reply events is fixed, and consists of only two parameters. The first parameter is the *reply ticket number*, and the second parameter is the *reply message*. No other fields should be present.

For example, if a page was sent from ServiceCenter's problem management, and a reply to the page is desired, the reply event type would be *pageresp*, and the reply ticket number might be *pm3271*. If the page recipient replies with *Yes*, the following line is added to the end of the input event file:

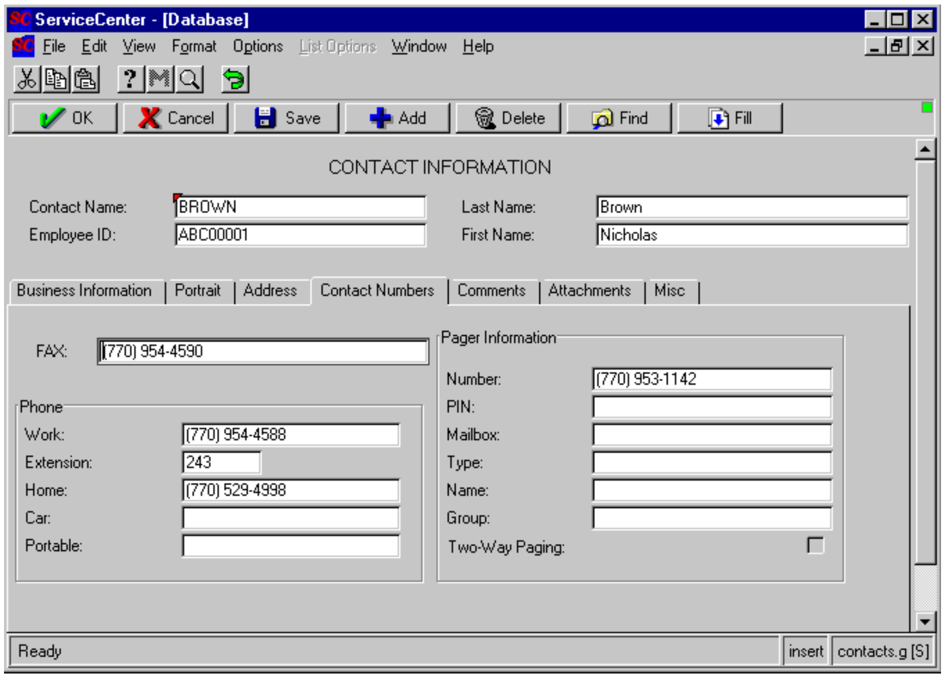
```
pageresp,,,,,,,^:pm3271^Page Response: Yes
```

The SCAuto Paging Server would then create this event within ServiceCenter, and the problem ticket would be updated with the reply.

Using two way paging

The key to two way paging is the *page.response* field, which resides in the contacts database.

The *Send Page Response* (*page.response*) field, if selected in the Pager Information box of the *User Contact Form* (Figure 7-3), will trigger a response for each pager message sent by the individual named in the contact record. A new event type, *pageresp*, is used for all page responses.



CONTACT INFORMATION

Contact Name:	BROWN	Last Name:	Brown
Employee ID:	ABC00001	First Name:	Nicholas

FAX: (770) 954-4590

Phone

Work:	(770) 954-4588
Extension:	243
Home:	(770) 529-4998
Car:	
Portable:	

Pager Information

Number:	(770) 953-1142
PIN:	
Mailbox:	
Type:	
Name:	
Group:	
Two-Way Paging:	☐

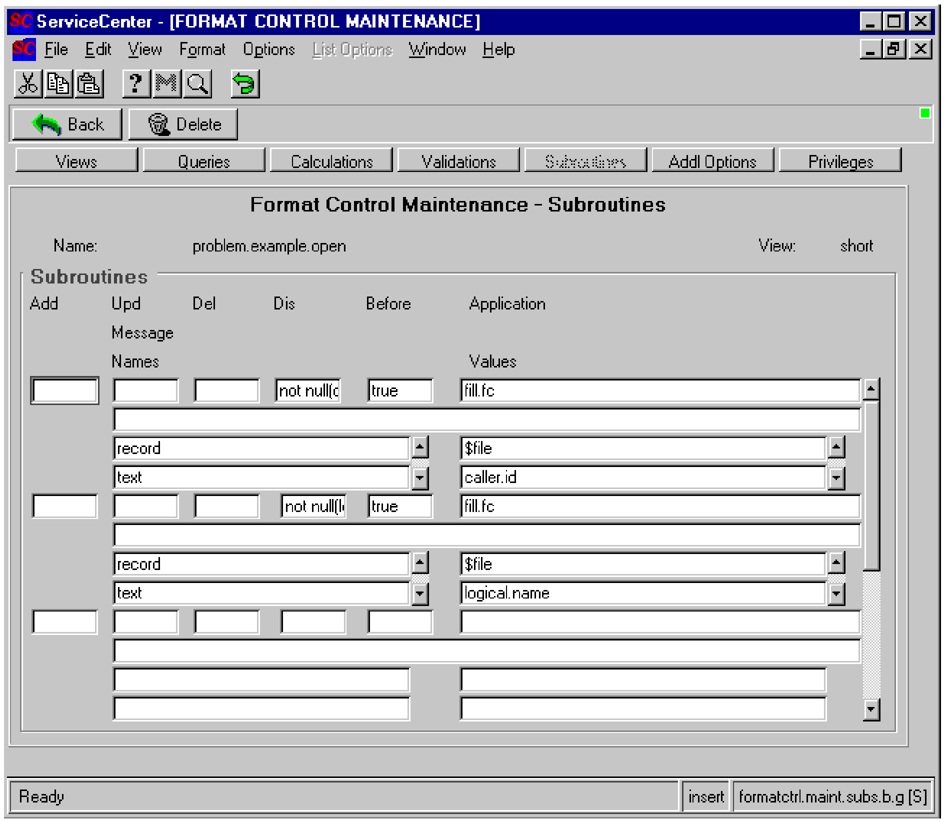
Figure 7-3. User Contact Form

The scauto.page application checks the value in the *page.response* field. If it is true, a value passed to *User Sequence* (query), when calling, the application is appended to the page. The SCAuto paging software returns that value as the first element of the response to *eventin*.

Using two way paging in Problem Management

Although paging can be invoked within any application using Format Control, the prototype for paging in ServiceCenter is attached to the Problem Management application.

In Figure 7-4, both a numeric (prompt) and an alphanumeric (text) message is sent to the Contact Name (name), using @ (*string1*) as the separator character. When the page is sent, a reply will be written to the eventin table and identified by the string "pm" followed by the problem number (query). The page will only be sent if the problem's Priority Code is *EMERGENCY* and a Contact Name exists in the problem record. *Chapter 7* of the ***Base Utilities Manual*** describes how to access and use ServiceCenter's Format Control capabilities.



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.example.open View: short

Subroutines

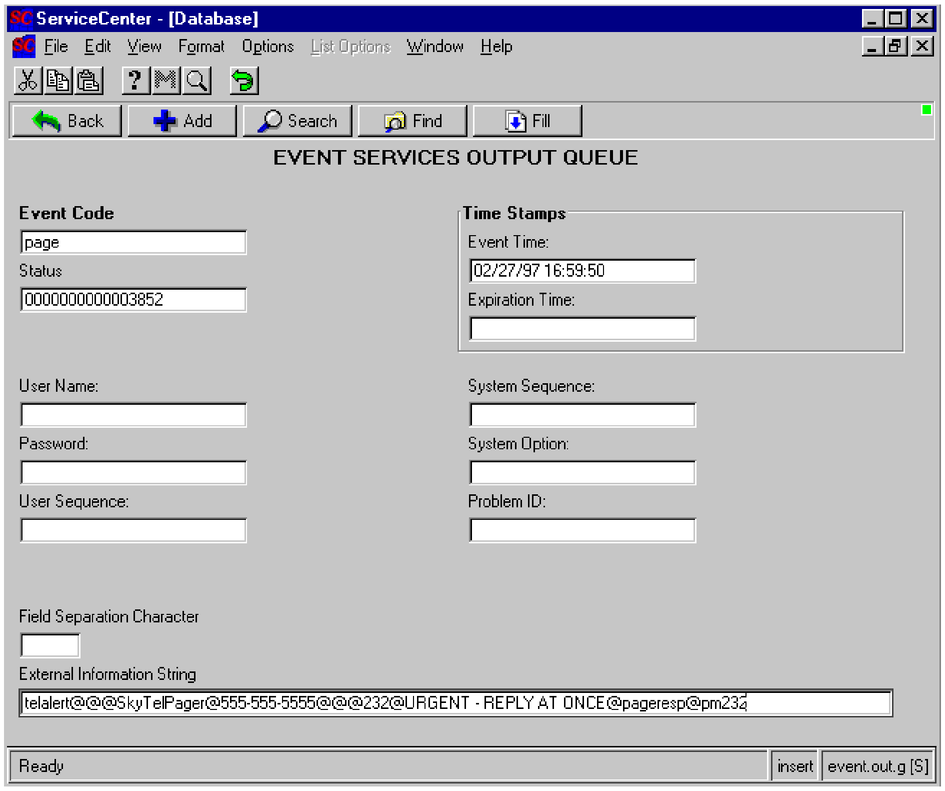
Add	Upd	Del	Dis	Before	Application
			not null(c	true	fill.fc
			not null(l	true	fill.fc

Message Names	Values
record	$file
text	caller.id
record	$file
text	logical.name

Ready | insert | formatctrl.maint.subs.b.g [S]

Figure 7-4. Format Control Paging Profile

When the format control is executed, it writes a record in the eventout queue, as shown below.


ServiceCenter - [Database]
File Edit View Format Options List Options Window Help
Back Add Search Find Fill

EVENT SERVICES OUTPUT QUEUE

Event Code: page

Status: 0000000000003852

Time Stamps

Event Time: 02/27/97 16:59:50

Expiration Time:

User Name:

Password:

User Sequence:

System Sequence:

System Option:

Problem ID:

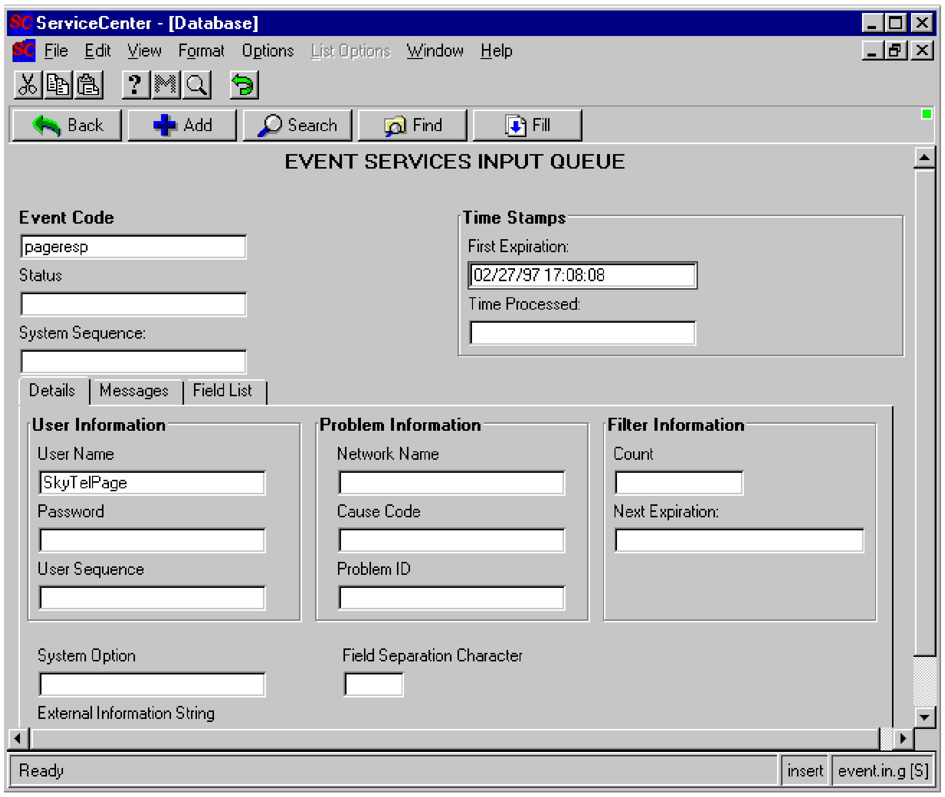
Field Separation Character

External Information String: telalert@@@SkyTelPager@555-555-5555@@@232@URGENT - REPLY AT ONCE@pageresp@pm232

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

Figure 7-5. Page Output Event

When SCAuto's paging software sends the page, the page response code - *pm390* - is used to identify the received response. In SCAuto, the response updates the problem ticket by adding a new event called *pageresp* to the eventin table. This event, shown below, calls the scauto.problem application to update problem PM390.



EVENT SERVICES INPUT QUEUE

Event Code: pageresp
Status:
System Sequence:

Time Stamps
First Expiration: 02/27/97 17:08:08
Time Processed:

User Information
User Name: SkyTelPage
Password:
User Sequence:

Problem Information
Network Name:
Cause Code:
Problem ID:

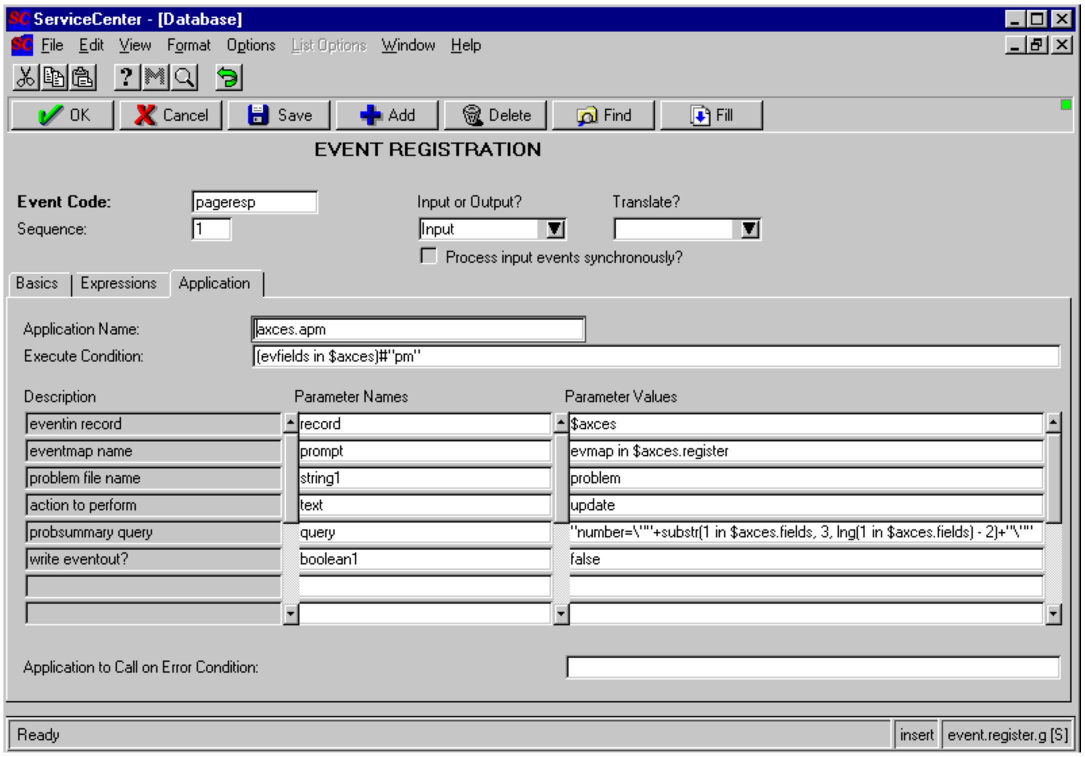
Filter Information
Count:
Next Expiration:

System Option:
Field Separation Character:
External Information String

Figure 7-6. Page Input Event

Registering pageresp Events

Pager Responses are generic. In other words, a page can be dispatched if a problem is critical, if a change requires approval, or if a new node is activated. You will want to update the problem, approve the change or acknowledge the node activation respectively, and each operation executes a different application within ServiceCenter. The Event Registration table, Figure 7-7, is used to register all events. When a response is received, the problem is updated with that response.


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

EVENT REGISTRATION

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

Application Name: axces.apm
Execute Condition: (evfields in $axces)#"pm"

Description	Parameter Names	Parameter Values
eventin record	record	$axces
eventmap name	prompt	evmap in $axces.register
problem file name	string1	problem
action to perform	text	update
probsummary query	query	"number=\""+substr(1 in $axces.fields, 3, lng(1 in $axces.fields) - 2)+"\""
write eventout?	boolean1	false

Application to Call on Error Condition:

Ready insert event.register.g [S]

Figure 7-7. Event Register

The *Execute* Condition is used to determine whether to use this registration record; other *pageresp* registrations may call *scauto.cmr* or *scauto.inventory* rather than *scauto.problem* as the *Application Name*. The pmpage response event map simply writes the response to the *update.action* field in the problem above. More detailed information on events, event registration and event mapping is included in the ***Event Services Guide***.

Errors

When there is an error sending a two way page (that is any *page* event with the extra reply parameters), the SCAuto Paging Server will create a reply event that states an error has occurred. This allows users from within ServiceCenter to know that an expected response is not forthcoming.

The SCAuto Paging Server determines if there was an error by checking the completion code upon the completion of the external command. Following usual conventions in UNIX, a completion code of **0** indicates success. In case the paging system does not indicate errors through this convention, you may turn this feature off using the *-noerror* flag. Whether or not this flag is set, the error will still be logged in the SCAuto Paging Server's log file.

Note: Errors which occur after the page is submitted to the paging system will not be detected this way. If such errors need to be caught and passed on to ServiceCenter, the paging system will need to be configured to create a reply event as if a reply occurred.

Sample Shell Files

pager.sh

The following shell script is included with the UNIX version of SCAuto Paging Server:

```
#!/bin/sh
#
# Script to parse pager event and call "telalert" command.
# Each event field is a separate shell parameter.
# These parameters are:
#   1 - vendor name, should be blank or ’telalert’
#   2 - name to page
#   3 - group to page
#   4 - type of page
#   5 - phone number
#   6 - pin number
#   7 - voice mail box number
#   8 - numeric message
#   9 - alphanumeric message
#  10 - event type (for two way pages)
#  11 - ticket number (for two way pages)
# Some of these parameters exclude the use of others.  For instance, only
# one name, group, or type of page may be specified in a single page event;
# and phone, pin, and voice mail number is useful only when a type of pager
# is specified.
# The name of the telalert command to  use.  The most useful values here
# are "telalert" and "telalertc".  COMMAND=${TELALERTBIN:-/ps00/telalert/bin}/telalertc

# Where to log commands.  This should not be the same as a file used # by scpager or
any other application. LOGFILE=${TELALERTDIR:-/ps02/ps01/chase20}/pager.log

# How long to wait for an acknowledgement ACKWAIT=2h

# Name for Reponses section entry to use (for two-way paging) RESPONSE=ServiceCenter

# Notifications section entry to use.  This can be blank, in which case
#   the   destination   pager   definition   needs   a   NotificationsReply   keyword.
REPLY=ServiceCenter

#  Default  destination  if  none  is  given  (this  shouldn’t  happen)  DEFAULT_DEST="-i
ServiceCenter"
#
# You should no
t need to mess with anything below here.
#
PARM_NAME=$2
PARM_GROUP=$3
PARM_TYPE=$4
PARM_PHONE=$5
PARM_PIN=$6
PARM_VMBOX=$7
PARM_NUMMSG=$8
PARM_MSG=$9
shift
PARM_EVENT=$9
shift
```

```
PARM_INFO=$9

# Check parameters and build command line if [ -n "$PARM_NAME" ]; then
DEST="-i \"$PARM_NAME\""
elif [ -n "$PARM_GROUP" ]; then
DEST="-g \"$PARM_GROUP\""
elif [ -n "$PARM_TYPE" ]; then
DEST="-c \"$PARM_TYPE\""
if [ -n "$PARM_PHONE" ]; then
DEST="$DEST -n \"$PARM_PHONE\""
fi
if [ -n "$PARM_PIN" ]; then
DEST="$DEST -n \"$PARM_PIN\""
fi
if [ -n "$PARM_VMBOX" ]; then
DEST="$DEST -n \"$PARM_VMBOX\""
fi
else
DEST="$DEFAULT_DEST"
fi
# Get messages
if [ -n "$PARM_NUMMSG" ]; then
MESSAGE="-mp \"$PARM_NUMMSG\""
else
MESSAGE=""
fi
if [ -n "$PARM_MSG" ]; then
MESSAGE="-m \"$PARM_MSG\""
else
MESSAGE=""
fi

# If we have the PARM_EVENT variable set, we have a bunch of extra stuff to add
# to support two-way paging.  We use -remark and -ticket both to specify
# information we want back on a page response.  (this makes things simpler to
# parse when Telalert is running on Windows NT) TWOWAY=""
if [ -n "$PARM_EVENT" ]; then
if [ -n "$ACKWAIT" ]; then
TWOWAY="-ackwait $ACKWAIT"
fi
if [ -n "$RESPONSE" ]; then
TWOWAY="$TWOWAY -response $RESPONSE"
fi
if [ -n "$REPLY" ]; then
TWOWAY="$TWOWAY -notificationreply $REPLY"
fi
TWOWAY="$TWOWAY -remark $PARM_EVENT"
if [ -n "$PARM_INFO" ]; then
TWOWAY="$TWOWAY -ticket $PARM_INFO"
fi
fi
if [ -n "$LOGFILE" ]; then
echo "=================" >>$LOGFILE
echo "Telalert options: $DEST $MESSAGE $TWOWAY" >> $LOGFILE
fi
# Now give the command eval "exec $COMMAND $DEST $MESSAGE $TWOWAY" >> $LOGFILE
```

pager.cmd

The following command file is included with the Windows NT version of SCAuto Paging Server:

```
@echo off
rem
rem Process ServiceCenter page events destined for Telalert
rem
rem Each event field is a separate parameter.These parameters are:
rem   1 - vendor name, should be blank or ’telalert’
rem   2 - name to page
rem   3 - group to page
rem   4 - type of page
rem   5 - phone number
rem   6 - pin number
rem   7 - voice mail box number
rem   8 - numeric message
rem   9 - alphanumeric message
rem  10 - event type (for two way pages)
rem  11 - ticket number (for two way pages)
rem Some of these parameters exclude the use of others.

if “&TELALERTBIN%” == ““ set TELALERTBIN=c:\usr\telalert

rem
rem Set this line to the command to use to run telalert
rem COMMAND=%TELALERTBIN%\telalert

rem
rem Set these parameters to appropriate choices for your site
rem

rem Where to log commands (should not be the same as a file used by scpager
rem or any other application.)
set LOGFILE=%TELALERTCFG%\ServiceCenter\pager.log

rem How long to wait for acknowledgement
set ACKWAIT=2h

rem Name for Reponses section entry to use (for two-way paging)
set RESPONSE=ServiceCenter

rem Notifications section entry to use.
rem This can be blank, in which case the destination pager definition
rem needs a NotificationsReply keyword.
set REPLY=ServiceCenter

rem Default destination if none is given (this shouldn't happen)
set DEFAULT_PAGE=-i ServiceCenter

rem ===============================================================
rem
rem Give names to our parameteres
rem
set PARM_NAME=%2
set PARM_GROUP=%3
set PARM_TYPE=%4
set PARM_PHONE=%5
set PARM_PIN=%6
set PARM_VMBOX=%7
set PARM_NUMMSG=%8
set PARM_MSG=%9
shift
set PARM_EVENT=%9
shift
set PARM_INFO=%9

rem
rem Now check parameters and build command line
rem
if %PARM_NAME%==”” goto check_grooup
set COMMAND=%COMMAND% -g %PARM_NAME%
goto check_message

:check_group
if %PARM_GROUP% == ““ goto check_type
set COMMAND=%COMMAND% -g %PARM_GROUP%
goto check_message

:check_type
if %PARM_TYPE% == ““ goto use_default
set COMMAND=%COMMAND% -c %PARM_TYPE%
if not %PARM_PHONE% == ”” set COMMAND=%COMMAND% -n %PARM_PHONE%
if not %PARM_PIN% == ““ set COMMAND=%COMMAND% -pin %PARM_PIN%
if not %PARM_VMBOX% == ““ set COMMAND=%COMMAND% -mailbox %PARM_VMBOX%

:use_default
set COMMAND=%COMMAND% -g %DEFAULT_PAGE%

:check_message
if not %PARM_MSG% == ““ set COMMAND=%COMMAND% -m %PARM_MSG%
if not %PARM_NUMMSG == ““ set COMMAND=%COMMAND% -mp %PARM_NUMMSG%

rem
rem Check for two-way page infor and modify command
rem

rem We use -remark and -ticket both to specify
rem information we want back on a page response.
rem(this makes things simpler to  parse when Telalert is running on Windows NT)

if %PARM_EVENT% == ““ goto submit
if not %ACKWAIT% == ““ set COMMAND=%COMMAND% -ackwait %ACKWAIT%
if not %RESPONSE% == ““ set COMMAND=%COMMAND% -reponse %RESPONSE%
if not %REPLY% == ““ set COMMAND=%COMMAND% -notificationreply %REPLY%
set COMMAND=%COMMAND% -remark %PARM_EVENT%
if not %PARM_INFO% == ““set COMMAND=%COMMAND% -ticket %PARM_INFO%

:submit

rem Now give the command
if not %LOGFILE% == ““ echo %COMMAND% >> %LOGFILE%
echo %COMMAND%
%COMMAND
```

Trouble Shooting

If SCPager is to be run as a service, use the following procedure:

This process requires tailoring of the *sc.cfg* file.

The ServiceCenter install process adds these lines to *sc.cfg*:

```
# D:\PROGRA~1\SERVIC~1\SCAUTO\FAX\scfax.exe -server jsmithg2k.12690
# D:\PROGRA~1\SERVIC~1\SCAUTO\PAGER\scpager.exe -server jsmithg2k.12690
```

1. Un-comment the appropriate line, i.e. remove the pound (**#**) symbol before text in line.

If the verification test fails during connection between the SCAuto Base (scautod) and the SCAuto Application (scpager) please check the following common failures:

- Is scautod running on ServiceCenter Server Platform?

  Use ServiceCenter *status* command to check.

  - If it is *not running* start it from the status option *start schedulers* selecting *scauto.startup*.
  - If the start fails the scautod daemon logs to the *sc.log* file, so be sure to specify and check the log for any error messages. Check the TCP/IP specifications for your platform, a service name other than that specified for ServiceCenter server is required, and the *scauto:* keyword specified in the *sc.ini* file must match this name.

  If it *is running* check the service name specified in the *sc.ini* file on the ServiceCenter platform for the *scauto:* keyword.

  - If the keyword is not specified scautod defaults to the services name *scauto*. This will be useful in the next step.

- Are the TCP/IP specifications correct on the SCAuto UNIX or Microsoft Windows platform?

  Check the host and service specification, are they specified and do they *match* the *host* and *service* name of s*cautod?*

  - Specifications may be made in local files (*UNIX /etc/hosts,/etc/services*) or a name server may be used. Check with your network administrator for specification. Also, check your specification in the program, configuration file, or passed parameters and make sure they all match (ex. scpager -server **host.service**).

- Is there connectivity between SCAuto Base (server) and the SCAuto Pager?

  Ping the SCAuto Base (scautod) platform from the SCAuto pager platform.

  - If this fails and all TCP/IP specifications are correct, contact your network administrator for assistance.

If you have an event in the eventin file but no RAD application is invoked (e.g., two way paging only) check the following:

- Is the Event Scheduler running on the ServiceCenter server platform?

  Use ServiceCenter **status** command to check.

  If it is *not running* start it from the status option *start schedulers* selecting *event.startup*.

  - If the start fails, review error messages and retry. If errors persist, call Peregrine Systems' Customer Support.

  If it *is running* and not processing, stop the Event Scheduler and build a new *schedule record* and restart.

- Review *Basic Trouble Shooting* section in the ***Event Services Guide*** in regard to schedule record specifications

No event created in eventout file but RAD application has been invoked (e.g., a problem was opened to generate a page and no pager event created) check the following:

- Use manual paging function in Event Services to create a pager event. Refer to *Sending a Page* section in the ***Event Services Guide***.

  - If this succeeds the Event Services RAD pager function is operable, and the problem is associated with the application generating the page (ex. Problem Management), or the specification(s) in the **contacts** table.

- Review the *Basic Trouble Shooting*, and *Using Format Control to Write Page Messages* sections in the ***Event Services Guide***.

If you have an event in the eventout file but the external SCAuto Application is not invoked (e.g., page event and page was not issued) check the following:

- Is there connectivity to the SCAuto Base server? Review *connection between the SCAuto Base (scautod) and the SCAuto Application (client)* trouble shooting section.

- Is SCAuto application (scpager) active?

  If it is *not running* check the SCAuto pager logfile *(default scpager.log)* for errors.

  - Correct errors and restart using appropriate command with the *-debug* option. The option will allow you to determine if the page operation was attempted and the service product (Telalert, etc.) failed, or if it's an SCAuto pager failure.
  - If restart fails review the SCAuto pager logfile.
  - If the page operation was attempted and failed, check the *service products log* for any errors or messages. SCAuto pager application in debug will log the command issued to the service product.

- If the command is present, issue the command manually. If it fails, correct the problems with the service product and retry.

  If it *is running*, check the SCAuto application logfile for errors. Correct errors and restart using appropriate command with the debug option if available. Recheck the logfile and try commands manually as shown with the -debug option.

If you do not have an event in the event.in file but the external SCAuto Application event has occurred(e.g., two way paging only) check the following:

- Is there connectivity to the SCAuto Base server? Review *connection between the SCAuto Base (scautod) and the SCAuto Application (client)* trouble shooting section.
- Is SCAuto pager active?

  If it is *not running* check the SCAuto pager logfile *(scpager.log default)* for errors.

  - Correct errors and restart using appropriate command with the *-debug* option. The debug option allows you to determine if it is an SCAuto application failure or a product installation or specification failure.
  - If restart fails using the debug option review the SCAuto application logfile for any new error message information.

  If it *is running*, check the SCAuto application logfile for errors.

  - Correct errors and restart using the appropriate command with the debug option if available.
  - Recheck the logfile and correct errors and retry.

If you need assistance in the resolution of your problem, please have the following available before calling for support:

- Core files.
- Error logs produced by service program (e.g. Telalert) and SCAuto application with debug option (scpager.log).
- ServiceCenter log and Scheduler log (s) if specified.
- ServiceCenter msglog for affected applications, or services.
- ServiceCenter print of agent status (Event Services Menu).
- Unloaded Event records relevant to errors.

Unloaded filter specifications if relevant.

Appendix A Contacting Peregrine Systems

For further information and assistance with SCAuto Applications for Windows NT and UNIX, contact Peregrine Systems' Customer Support. Current details of local support offices are available through these main contacts.

North America, South America, Asia/Pacific

Telephone:	(1) (800) 900-8152 (within US only, toll free)
	+(1) (858) 794-7402
Fax:	+(1) (858) 480-3986
Email:	`support@peregrine.com`
Headquarters:	Peregrine Systems, Inc.
	Attn: Customer Support 3611 Valley Centre Drive San Diego, CA 92130

Europe, Africa

Telephone:	(0) (800) 834 770 (within United Kingdom only, toll free)
	+(44) (0) (02) 8334-5844
Fax	+(44) (0) (02) 8334-5890
Email:	`uksupport@peregrine.com`

Documentation Web Site

For a complete listing of the current SCAutomate documentation, see the Documentation pages on the Peregrine Systems, Inc. Customer Support Web site at:

```
http://support.peregrine.com
```

You will need the current login and password to access this Web page.

For copies of the manuals, you can download `.pdf` files of the documentation using the Adobe Acrobat Reader (also available on the Web site). Additionally, you can order printed copies of the documentation through your Peregrine Systems sales representative.

Index

Peregrine
S Y S T E M S®

The Infrastructure Management Company™