HP SCAuto Applications

for supported Windows® and Unix® operating systems

Software Version: 4.0.2

User's Guide

Document Release Date: October 2007
Software Release Date: October 2007

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notices

Trademark Notices

Java™ and all Java based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

Oracle® is a registered US trademark of Oracle Corporation, Redwood City, California.

Unix® is a registered trademark of The Open Group.

Documentation Updates

This guide's title page contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates, or to verify that you are using the most recent edition of a document, go to:

http://ovweb.external.hp.com/lpe/doc_serv/

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

You can visit the HP software support web site at:

www.hp.com/go/hpsoftwaresupport

HP Software online support provides an efficient way to access interactive technical support tools. As a valued support customer, you can benefit by using the support site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To find more information about access levels, go to:

h20230.www2.hp.com/new_access_levels.jsp

To register for an HP Passport ID, go to:

h20229.www2.hp.com/passport-registration.html

Contents

1 SCAuto Applications

Welcome to HP SCAuto Applications for Windows and Unix. This product is part of the suite of SCAuto (ServiceCenter Automate) interface products that integrate Service Manager with premier network and systems management tools.

Topics in this chapter include:

What do I need to know?

This guide assumes you have:

- Working knowledge of Service Manager applications, Service Manager client/server, and Windows and Unix operating systems. While some procedures for these applications are explained, others are referenced. For more information, refer to the Service Manager online help.
- 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 Service Manager applications and Event Services.

Introduction to SCAuto

ServiceCenter Automate (SCAuto) provides event management services through a collection of automation products which enable external applications to be integrated with Service Manager. These generic services allow customers to create meaningful events from their environment and applications. Some common applications of this functionality include opening tickets, notification upon the closure of a ticket, automatic network inventory management and email management between disparate external mail programs.

The intent is to enable communication from any external application with any Service Manager RAD application, without predetermining what that communication requires.

SCAuto Listener

A dedicated Java-based listener (scautolistener) manages all SCAuto to and from Service Manager connections. You can start `scautolistener` from the command line or by using an entry in the `sm.ini` file. For more information, refer to the Service Manager online help.

For example, to start a listener on port 12690, you could use this command:

```
sm -scautolistener:12690
```

When the listener has started and is running, it accepts connections from SCAuto clients.

While running, it will appear with the user name SCAuto listener from within Service Manager, and should be stopped from there if needed. Existing clients will not be terminated when scautolistener is stopped.

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

Scautolistener uses either the `sm.ini` file or the command line to find its parameters.

Table 1 SCAuto listener Parameters

Parameter	Description
system:	Service name for the Service Manager system.
path:	Directory containing Service Manager database.
log:	(optional) Output file for diagnostic log messages. These first three parameters are also used in the `sm` program.
scauto:	Service name for SCAuto.

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

Under Windows these service names are defined in `Winnt\system32\drivers\etc\services`. The Service Manager installation on Windows 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 listener running on the Service Manager server platform. Once a connection is established with the listener, a process on the server is created. The new process runs similar to an `sm` process in Service Manager and reads and writes event records directly to the event files.

Service Manager 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 Service Manager. 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.

External Event Services

Applications external to the Service Manager 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, Service Manager 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 Service Manager Event Manager. Service Manager output events are read from the Service Manager 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 Service Manager.

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 Service Manager with an scauto_connect call and specify which events and Service Manager server your application requires.

After successful connection, you may create, retrieve, or delete Service Manager 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 Service Manager servers.

Another option provided through SCAuto is the email interface. The email option provides email services to Service Manager. With some extensions, this option can be used to handle other than email events. An example of this would be a dispatch application which runs from email or has an email 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 email.

Application Enhancement

Service Manager 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 Service Manager 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. Please check with your HP Account Representative for a current product list. The products in this document include:

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

2 Installation

You can install SCAuto applications on a Windows or Unix server. This section contains information about installation requirements and how to install the SCAuto applications, SCAuto Fax, SCAuto Mail, and SCAuto pager.

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

Topics in this chapter include:

Windows Installation

The bulk of the installation is done automatically. However, with each SCAuto application you may need to perform specific setup configuration unique to that program. For more information, see Customizing SCAuto Mail on page 55.

Installing SCAuto Applications on Windows systems

Follow these steps to install the SCAuto applications:

1. Log in to the Windows server as a user with local administrator privileges.
2. Insert the SCAuto installation media into the appropriate drive of the server.

   If you are installing on a system that has auto-run enabled, the installation media browser starts automatically. If auto-run is disabled, follow these steps to start the installation media browser manually.

   a. Navigate to the installation media folder.

   b. Double-click `setupwin32.exe`.

   The HP SCAuto Applications installation program opens.
3. Click **Next** to read and accept the licensing agreement.
4. Select the **I accept the terms in the License Agreement** option.

   The **Next** button becomes active.
5. Do one of the following:

   - Click **Next** to accept the default installation folder.

     The default installation folder is:
     `C:\Program Files\HP\HP_SCAuto_Applications_4.0.2`

     

     Do not install the applications over existing versions of SCAuto. You must uninstall first, or install into a new folder.

Click **Browse** to choose a different installation location.The installation

You must install the SCAuto applications in a folder containing only ASCII characters in the folder name. The applications cannot start if installed in a folder with non-ASCII characters in the folder name.

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

- `\Fax` for SCAuto Fax
- `\Mail` for SCAuto Mail
- `\Pager` for SCAuto Pager

6 Select the installation type:

- Typical installs all SCAuto products.
- Custom allows you to install specific product components.

The available options vary with your selections. The options you see may differ from those presented here.

7 Click **Next** to enter connection information.

8 Fill in the following fields

Table 2 Connection Information

Field	Default Value
Server Name for SCAuto to connect to	scautohp.com
Server Port for SCAuto to connect to	12690
MAPI profile for SCAuto Mail	falcon

9 Click **Next** to review the installation information.

The installation information page opens.

10 Click **Install** to begin copying the installation files.

You can stop the installation by clicking **Cancel**.

A summary page opens when the installation is complete.

11 Click **Finish** to exit the Setup wizard.

12 To set up SCAuto Mail, proceed to SCAuto Mail for Windows on page 23.

Running an SCAuto Application as a Windows Service

To run an SCAuto application as a Windows service, you need to configure the service startup information. During the installation, the setup program installed the SCAuto service to run under the default LocalSystem user ID. You can change it to run under another ID. The selected user must have the rights to log on to the MAPI profile that you specified during the setup.

Follow these steps to set up an SCAuto service to run under a specific user ID:

1 From the Windows main menu, click **Start > Settings > Control Panel > Services**.

The Services window opens.

2 Select `HP SCAuto Applications Fax`, `HP SCAuto Applications Mail`, or `HP SCAuto Applications Pager` from the Services list and click **Startup**.

The Service dialog box opens.

3 Click **This Account**.

The LocalSystem user ID is displayed in the This Account field.

4 Click the **Browse** button next to the This Account field.

The Add User dialog box opens.

5 Double-click on the desired <user ID> from the Names list.

The Add Name field displays the user ID.

6 Click **OK**.

The Windows Service dialog box opens and the Account field now displays the user ID.

7 Enter the Password for that user ID.

8 Re-enter the password in the Confirm the Password field.

9 Click **OK**.

A message is displayed stating that the user has been granted the Log On As A Service right.

10 Click **OK**.

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

Uninstalling SCAuto Applications from Windows systems

Follow these steps to uninstall SCAuto applications:

1 Stop the SCAuto application if it is running.

2 From the Windows main menu, click **Start** > **Settings** > **Control Panel** > **Add/ Remove Programs**.

The Add/Remove Programs dialog box opens.

3 Select SCAuto Applications and click **Change/Remove**.

The SCAuto Applications installation program opens.

4 Click **Next**.

5 Mark the checkboxes to indicate the features you would like to remove and

6 Click **Next**.

7 Review the uninstallation information.

8 Click **Uninstall** to remove the indicated features.

You can stop the installation by clicking **Cancel**.

A summary page opens when the installation is complete.

9 Click **Finish** to exit the Setup wizard.

10 Click **Close** to exit the Control Panel.

Unix Installation

The bulk of the installation is done automatically for you. However, with each SCAuto application you may need to perform specific setup configurations unique to that program. For more information, see Customizing SCAuto Mail on page 55.

Case sensitivity may be dependent on the Unix system that you are using.

Installing SCAuto Applications on Unix systems

Follow these steps to install the SCAuto applications:

1. Log on to the server.
2. Insert the SCAuto installation media into the appropriate drive of the server.
3. Mount the installation media.
4. CD to the mount location.
5. Run the appropriate setup script. For example, you could run:

   `<Path to Installation files>: setupHP11-parisc`

   Alternatively, you can run the script in console mode by using the following command:

   `setupHP11-parisc -console`

   The HP SCAuto Applications installation program starts.
6. Click **Next** to read the licensing agreement.

   Console mode users chose 1 for Next, and press Enter to read and accept the licensing agreement.
7. Select the **I accept the terms in the License Agreement** option.

   The **Next** button becomes active.

   Click **Next** to accept the licensing agreement.

   Console mode users press 1 to accept the terms of the license agreement. and press 1 to proceed.

The HP SCAuto Applications Install Location screen opens.

8 Do one of the following:

- Click **Next** to accept the default installation directory.

Do not install the applications over existing versions of SCAuto. You must uninstall first, or install into a new directory.

Click **Browse** to choose a different installation location.

You must install the SCAuto applications in a directory containing only ASCII characters in the directory name. The applications cannot start if installed in a directory with non-ASCII characters in the directory name.

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

- `/Fax` for SCAuto Fax
- `/Mail` for SCAuto Mail
- `/Pager` for SCAuto Pager

Console users, specify a directory or press Enter to accept the default directory, and then press 1 to proceed to the next screen.

The installation type screen opens.

9 Select the installation type:

- Typical installs all SCAuto products.
- Custom allows you to install specific product components.

Click **Next** to enter connection information.

Console users press 1 for Typical or 2 for Custom, and then press 1 to specify connection information.

The connection information screen opens.

10 Fill in the following fields.

Table 3 Connection Information

Field	Default Value
Server Name for SCAuto to connect to	scautohp.com
Server Port for SCAuto to connect to	12690
Mailbox for SCAuto Mail	falcon

11 Click **Next** to review the installation information.

Console mode users specify the connection information, and then press 1 to proceed to review the installation information.

The installation review screen opens.

12 Click **Install** to start installing.

You can stop the installation by clicking **Cancel**.

Console users, press 1 to start installing.

A summary page opens when the installation is complete.

13 Click **Finish** to exit the Setup wizard.

Console users, press 3 to exit the wizard.

14 To set up SCAuto Mail, proceed to SCAuto Mail for Unix on page 43.

Uninstalling SCAuto Applications from Unix systems

Follow these steps to uninstall the SCAuto applications:

1 Stop the SCAuto applications.

2 CD to `<install_directory>/_uninst`

3 Run the `uninstall` file.

3 SCAuto Mail for Windows

ServiceCenter Automate (SCAuto) Mail is a Windows program which allows sending and receiving of email within Service Manager using different external mail applications. Under Windows, SCAuto Mail uses the Messaging Application Program Interface (MAPI). SCAuto Mail for Windows is an SCAuto adapter product.

SCAuto Mail sends outgoing email the same way as the SCEMAIL program that comes with Service Manager. If you are already running SCEMAIL, you replace it with SCAuto Mail.

This chapter includes the following topics:

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 login and account information which allows you to interface with your email systems. SCAuto Mail signs on using the SCAuto Mail profile, not the external mail account or login names.

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 Service Manager.

Configuration

SCAuto Mail files

Executable files

There are two executable files. `Scmapi.exe` sends and receives email between external mail clients and Service Manager. `Scmapisrv.exe` allows you to start SCAuto Mail as a Windows 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 email events.

Outlook form

`Pmo.oft` is a Outlook message item. You can open a Service Manager ticket by sending email 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.

The three values are:

- CurrentVersion: SCAuto Mail application version number.
- StartupType:
  - StartupNTSrv - start as a Windows service
  - StartupSC - start from Service Manager
  - 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 Optional Start-Up Parameters

SCAuto Mail parameter usage:

- True or false parameters (Boolean) take 1 or 0 after the colon to turn them on or off, respectively.
- Include the hyphen before the parameter name when using it on the command line.
- Omit the hyphen before the parameter name when using it in the `scmapi.ini` file.
- Omit the angle brackets around the parameter value.

Table 4 Optional scmapi startup parameters

Parameter	Description
-admin:<id>	Specifies the Service Manager operator ID to receive mail that is not addressed with Co:, or mail that cannot be delivered. The default value is **`falcon`**.
-clean:<0/1>	Indicates whether to remove the following items from the Service Manager outbound email message's body. • `ServiceCenter Operator:<operator name>` • `SCenter_cc:<cc name>` **`1`** = Clean the email. **`0`** = Do not clean the email.
-create_email:<0/1>	Indicates whether to create an email event if an event mapping fails. **`1`** = When event mapping fails, create an email event. (Default) **`0`** = When event mapping fails, do not create an email event.
-debug:<0/1/2>	Enables you to add debug information to the `scmapi.log` file. **`0`** = Turns off debugging mode. **`1`** = Turns on debugging mode. **`2`** = Logs full debugging information. This parameter turns on the **`-keepmail`** parameter as well.

Table 4 Optional scmapi startup parameters

Parameter	Description
-debugscautoevents:<0/1>	Enables SCAuto Mail to log more event generation information to the SCAuto Mail's log file. The default value is **0**.
-event_map_dir: <EventMap>	Specifies the relative path from the home directory to the directory containing the event map files. Event map files are text files which control the generation of various types of events from tags and values specified in email bodies. Usually this is specified as EventMap and the EventMap directory is a subdirectory of the home directory.
-events:<0/1>	Allows the creation of Service Manager events via email. For more information, see Events via Email on page 39.
-force_useremailtype: <0/1>	Scauto mail creates an eventin record with an event type of email when there are no matching conditions in any of the event map files. If you use **-force_useremailtype:1**, scauto mail will use the value specified in the useremailtype parameter instead.
-gui:<0/1>	Indicates whether to enable a pop-up dialog if additional login information is required (no profile was passed on the command line, or a password is required). If scauto mail is started as a service then you still will not see the dialog box. **1** = allow pop-up.
-keep_cc:<0/1>	**0** = Do not retain the CC field of the email. (Default) **1** = Retain the CC field.
-keep_to:<0/1>	**0** = Do not retain the TO field of the email. (Default) **1** = Retain the TO field.
-keepmail:<0/1>	Indicates whether to keep mail and events after they have been sent successfully. **0** = Delete mail and event after they have been sent successfully. **1** = Do not delete mail or events after they have been sent successfully.

Table 4 Optional scmapi startup parameters

Parameter	Description
-log:<file name>	Specifies the file to log messages. The default value is **`scmapi.log`**.
-mapi_mailserver_reconnect: <0/1>	Indicates whether to retry to connect SCAuto Mail to the MAPI mail server. **`1`** = Retry to connect **`0`** = Do not retry to connect. (Default)
-newline_chars: <new line characters>	Changes the default line break character(s). The default new line characters for Windows are is **`\r\n`**. The default new line character for Unix is **`\n`**. Valid values are: • **`\r\n`** • **`\r`** • **`\r`** • **`\n\r`**
-noincoming:<0/1>	**`1`** = Do not send incoming mail to Service Manager.
-nooutgoing:<0/1>	**`1`** = Do not retrieve events from the Service Manager eventout queue.
-numberof_retry_connect: <n>	Specifies the number of retries before exiting. The default value is **`10`**. **`0`** = retry forever.
-profile: <MAPI profile name>	Specifies the MAPI profile name.
-reconnect_interval:<n>	Specifies the number of seconds to sleep between retrying. The default value is **`120`** (two minutes).
-savesent:<0/1>	**`0`** = SCMAPI should delete mail from MAPI mailbox after forwarding to Service Manager. (Default) **`1`** = save mail in MAPI mailbox.

Table 4 Optional scmapi startup parameters

Parameter	Description
-sc_reconnect:<0/1>	Allows SCAuto Applications to retry to reconnect to Service Manager when Service Manager is restarted. **1** = Retry connecting to Service Manager. **0** = Do not try to reconnect to Service Manager. (Default) This parameter works together with the parameters **-reconnect_interval** and **-numberof_retry_connect**. This parameter impacts the K (kill) command in Service Manager's system status list. The K command closes the connection between Service Manager 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 Service Manager shutdown. So, SCAuto Mail will try to reconnect even after a K command.
-scmapi_address_delimiters: <delimiter>	Specifies the delimiter character(s) used between addressee names.
-scmapi_convert_addr_type: <mail address type>	Converts the incoming mail From field to entered mail address type.
-send_mail_as_attach: <file name>	Send the Service Manager's outbound mail message as an attachment <file name> in an email. When SCMAPI and Service Manager are installed in different 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.
-server:<host.service>	Specifies the SCAuto server to connect to. This is specified as hostname plus service name or port number, separated by a period. The default value is **scauto**.
-sleep:<n>	Specifies the number of seconds to sleep between checking for events and mail. The default value is **10** seconds.

Table 4 Optional scmapi startup parameters

Parameter	Description
-terminate_mapi_string: <MAPI errors>	Specifies the MAPI error message(s) that cause a termination of the SCAuto Mail. This parameter overwrites the **mapi_mailserver_reconnect** parameter. The MAPI error messages could be: • **E_OUTOFMEMORY** • **E_FAIL** • **MAPI_E_LOGON_FAILED** • **MAPI_E_NETWORK_ERROR** • **MAPI_E_DISK_ERROR** • **All** You can specify more than one MAPI error and separate errors by using a comma.
-usemaps:<0/1>	Indicates that the new, enhanced style of putting tags in email bodies is to be supported. If this parameter is set to **0** or is absent, scmail only honors the old style of tags.
-useremailtype: <email type>	By default Scauto mail queries the eventout table for records that have an event type of email. You can change this by using the **useremailtype** parameter. For example, if you specify the following: **useremailtype:testemail** Scauto mail will only process eventout records with a event type of testmail. The default value is **email**.
-usersepchar: <char>	Specifies a different event separator character. Enter the decimal representation of a character from 1 to 255. The default value is **^**, if not defined.

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:scautolistener.12690
```

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

ServiceCenter Automate Mail service starts one ServiceCenter Automate Mail process.

A parameter specified on the command line overrides the same parameter specified in the `scmapi.ini` file.

The following scmapi.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
```

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 Service Manager events going to Service Manager should be constructed from the email messages. This type of `.map` file has a special syntax which is oriented toward the creation of Service Manager event strings. For more information, see Customizing SCAuto Mail on page 55.

Microsoft Outlook Form

SCAuto Mail installs Microsoft Outlook forms, named `pmo.oft` and `pmo_98.oft`, in the Mail From subdirectory of your SCAuto Mail directory. You can use these forms to open a Service Manager ticket.

Table 5 Fields in the MS Outlook form

Name	A mail user account that SCAuto Mail watches for incoming messages.
E-Mail Address	The email address of the mail user account. This email address will be automatically resolved for a user who has an email address in an address book that is specified in the MAPI profile.
Co	A Service Manager 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 email.
Cancel	Exit.

Starting SCAuto Mail

Service Manager must already be installed and operational, and the SCAuto server (scautolistener) must be running, for SCAuto Mail to start and function properly.

SCAuto Mail 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 Service Manager 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\HP\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 > ServiceCenter Automate 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

scmapi -log:scmapi.log -profile:"My Profile Name"

1. Change to the SCAuto Mail directory.

   For example, you could enter:
   cd c:\Program Files\HP\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.

A parameter specified on the command line 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 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 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\HP\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 Task bar settings to create a menu item.

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 Email with Service Manager

Sending Service Manager Mail to Email

Sending Service Manager mail to email 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 email address for that user. This is accomplished through the following steps:

1. Login to Service Manager with an account that has SysAdmin authority.
2. Go to the operator or contact record.
3. Enter the email address for that person in the email field.
4. Save that operator or contact record.

There are different syntax variations when entering an email 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 Service Manager mail. If mail sent from Service Manager is undeliverable, it is returned to the user with an error message.

Sending Email back to Service Manager

Since Service Manager operators are not valid email addresses, mail into Service Manager 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 Service Manager recipient is specified by adding a line like the following in the body of the mail message:

```
Co: bob.helpdesk
```

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).

The event background processor in Service Manager must also be started in order to process the email events in the eventin file.

Events via Email

Problems can be opened, updated, and closed by sending email 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 email by setting the usemaps 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 usemaps parameter is set to 0 in the `scmapi.ini` file. Within the body of the email message, include these fields:

SC_event:event_type

This specifies the type of event created by this email 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 Service Manager default is used).

SC_user:userid

Allows specifying the operator to use for the event.

These fields must not be preceded by white space.

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

The following is a sample email message for a problem update:

```
From: Field Service Rep
To: Service Manager
Subject: Maybe this solves it…
SC_event: pmu
SC_var: 57133
I plugged the machine in. 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 usemaps parameter to 1 in the `scmapi.ini` file
2. Compose your email by using the tags defined in the map file.

The following is a sample email message for a problem open:

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

This email generates a Service Manager problem open event. A problem is opened in Service Manager with priority 1.

4 SCAuto Mail for Unix

SCAuto Mail can run in a Unix environment, allowing you to send and receive email within Service Manager using external Unix mail utilities. SCAuto Mail for Unix is an SCAuto adapter product and should be obtained through a HP 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 Service Manager, SCAuto Mail monitors a Unix mailbox.

This chapter includes the following topics:

The SCAuto Mail files

- SCAuto Mail for Unix has a single executable: `scmail`.
- SCAuto Mail for Unix uses the configuration file `scmail.ini`, which stores SCAuto Mail start-up parameters. This file is created in the SCAuto Mail target directory during installation.
- The Event map files allow you to customize your email events.

SCAuto Mail Optional Start-Up Parameters

SCAuto Mail parameter usage:

- True or false parameters (Boolean) take 1 or 0 after the colon to turn them on or off, respectively.
- Include the hyphen before the parameter name when using it on the command line.
- Omit the hyphen before the parameter name when using it in the `scmail.ini` file.
- Omit the angle brackets around the parameter value.

Table 6 Optional scmail startup parameters

Parameter	Description
-admin:<id>	Specifies the Service Manager operator ID to receive mail that is not addressed with Co:, or mail that cannot be delivered. The default value is **`falcon`**.
-clean:<0/1>	Indicates whether to remove the following items from the Service Manager outbound email message's body. • `ServiceCenter Operator:<operator name>` • `SCenter_cc:<cc name>` **`1`** = Clean the email. **`0`** = Do not clean the email.
-create_email:<0/1>	Indicates whether to create an email event if an event mapping fails. **`1`** = When event mapping fails, create an email event. (Default) **`0`** = When event mapping fails, do not create an email event.
-debug:<0/1/2>	Enables you to add debug information to the scmail.log file. **`0`** = Turns off debugging mode. **`1`** = Turns on debugging mode. **`2`** = Logs full debugging information. This parameter turns on the **`-keepmail`** parameter as well.

Table 6 Optional scmail startup parameters

Parameter	Description
-debugscautoevents:<0/1>	Enables SCAuto Mail to log more event generation information to the SCAuto Mail's log file. The default value is **0**.
-event_map_dir: <EventMap>	Specifies the relative path from the home directory to the directory containing the event map files. Event map files are text files which control the generation of various types of events from tags and values specified in email bodies. Usually this is specified as EventMap and the EventMap directory is a subdirectory of the home directory.
events:<0/1>	Allow creation of problem or change management events via email. For more information, see Events via Email on page 54.
-force_useremailtype: <0/1>	Scauto mail creates an eventin record with an event type of email when there are no matching conditions in any of the event map files. If you use **-force_useremailtype:1**, scauto mail will use the value specified in the useremailtype parameter instead.
-from:<0/1>	Send mail with **sendmail** command option **-f**. The default value is **0**.
-keep_cc:<0/1>	**0** = Do not retain the CC field of the email. (Default) **1** = Retain the CC field.
-keep_to:<0/1>	**0** = Do not retain the TO field of the email. (Default) **1** = Retain the TO field.
-keepmail:<0/1>	Indicates whether to keep mail and events after they have been sent successfully. **0** = Delete mail and event after they have been sent successfully. **1** = Do not delete mail or events after they have been sent successfully.
-log:<file name>	Specifies the file to log messages. The default value is **scmail.log**.
-mailbox:<mailbox> (or -mbox)	The file name of the mailbox.

Table 6 Optional scmail startup parameters

Parameter	Description
-newline_chars: <new line characters>	Changes the default line break character(s). The default new line characters for Windows are is **\r\n**. The default new line character for Unix is **\n**. Valid values are: • **\r\n** • **\r** • **\r** • **\n\r**
-noincoming:<0/1>	**1** = Do not send incoming mail to Service Manager.
-nooutgoing:<0/1>	**1** = Do not retrieve events from the Service Manager eventout queue.
-numberof_retry_connect: <n>	Specifies the number of retries before exiting. The default value is **10**. **0** = Retry forever.
-reconnect_interval: <n>	Specifies the number of seconds to sleep between retrying. The default value is **120** (two minutes).

Table 6 Optional scmail startup parameters

Parameter	Description
-sc_reconnect:<0/1>	Allows SCAuto Applications to retry to reconnect to Service Manager when Service Manager is restarted. **1** = Retry connecting to Service Manager. **0** = Do not try to reconnect to Service Manager server. (Default) This parameter works together with the parameters **-reconnect_interval** and **-numberof_retry_connect**. This parameter impacts the **K** (kill) command in Service Manager's system status list. The **K** command closes the connection between Service Manager 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 Service Manager shutdown. So, SCAuto Mail will try to reconnect even after a **K** command.
-server:<host.service>	Specifies the SCAuto server to connect to. This is specified as hostname plus service name or port number, separated by a period. The default value is **scauto**.
-sleep:<n>	Specifies the number of seconds to sleep between checking for events and mail. The default value is **10** seconds.

Table 6 Optional scmail startup parameters

Parameter	Description
-usemaps:<0/1>	Indicates that the new, enhanced style of putting tags in email bodies is to be supported. If this parameter is set to **0** or is absent, scmail only honors the old style of tags.
-useremailtype: <email type>	By default Scauto mail queries the eventout table for records that have an event type of email. You can change this by using the **useremailtype** parameter. For example, if you specify the following: **useremailtype:testemail** Scauto mail will only process eventout records with a event type of testmail. The default value is **email**.
-usersepchar:<char>	Specifies a different event separator character. Enter the decimal representation of a character from 1 to 255. The default value is **^**, 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.

Sample scmail.ini file:

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

Setting Up Event Map Files

The files in the `/EventMap/ToSC/` directory define how one or more Service Manager events going to Service Manager should be constructed from the email messages. This type of `.map` file has a special syntax which is oriented toward the creation of Service Manager event strings. For more information, see Customizing SCAuto Mail on page 55.

Starting SCAuto Mail

Service Manager must already be installed and operational, and the SCAuto server (scautolistener) must be running for SCAuto Mail to start and function properly.

SCAuto Mail 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.

Starting scmail

Use the following steps to 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 Service Manager. 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 Service Manager email events and turns them into email messages. When mail arrives for the SCAuto Mail mailbox, the mail is converted into Service Manager email events.

Using Email with Service Manager

Sending Service Manager mail to Email

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

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

To modify the operator record:

1. Login to Service Manager.
2. Access the user's operator record.
3. Enter the email 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 Service Manager to the user's external email address. If mail sent from Service Manager is undeliverable, it is returned to the user with an error message.

Sending Email back to Service Manager

The event background processor in Service Manager must be running in order to process the email events in the eventin file.

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

The email message cannot be addressed directly to a Service Manager operator. The Service Manager 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 Service Manager user for whom the email is intended.

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

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

Service Manager processes read the identifying syntax and route the message to the appropriate Service Manager 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 scmail.

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 smuser
From: smtestuser
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 Email

Tickets can be opened, updated, and closed by sending email via SCAuto Mail. The events parameter must be set to **1** in the `scmail.ini` file to enable this facility. Within the body of the email message, include these fields:

Table 7 Email message fields

Field	Description
SC_event: event_type	This specifies the type of event created by this email 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 Service Manager default is used).
SC_user:userid	Allows specifying the operator to use for the event.

These fields must not be preceded by white space.

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

The following is a sample email message for a problem update:

```
From: Field Service Rep
To: Service Manager
Subject: Maybe this solves it…
SC_event: pmu
SC_var: 57133
I plugged the machine in. Try connecting again and see if it
works. -- Joe.
```

5 Customizing SCAuto Mail

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

This chapter includes the following topics:

Overview

The event mapping files are located in the following directories:

Windows

```
C:\Program Files\HP\SCAuto\Mail\EventMap\ToSC\
```

Unix

```
<install_directory>/EventMap/ToSC/
```

For each type of Service Manager event there must be at least one `.map` file pointed to by the `event.ini` file in the `\EventMap\ToSC\ (/EventMap/ToSC/ for Unix)` subdirectory for SCAuto Mail.

The `.map` files in the `ToSC` directory tree define how one or more Service Manager events going TO Service Manager should be constructed from your email.

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

SCAuto Mail's event mapping function discussed here is not the same thing as the Service Manager 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 Service Manager. The event mapping within Service Manager Event Services, which is documented in the Event Services online help, is used to interpret the event transaction once it arrives in Service Manager'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\ (/EventMap/ToSC/ for Unix)`. The directory `EventMap` is named by the **`event_map_dir`** parameter in the `scmapi.ini` (Windows) or `scmail.ini` (Unix) file.

Under the SCAuto Mail `ToSC` subdirectory, a file named `event.ini` contains all the Service Manager events that you want to constructed from your email 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 Service Manager 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 email 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 email message. A tag is a line starting either with SC_ or EVENT_ followed by a colon(:) and a value in the body of an email message.

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

In order to generate a Service Manager event, the body of an email 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 email bodies formatted for use with earlier versions of SCAuto Mail. These tags are SC_type, SC_event, SC_var and SC_user.

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 email.

Event generation

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

1. Reads through the body of an email 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 email. Each line in the section [EMAIL] defines a Service Manager event type to be generated, if possible, from the email, 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 Service Manager transaction of type pmo.

3. The `.map` file is consulted. The `.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 Service Manager event according to the rules that defined in the `.map` file.

An email together with a given `.map` file may or may not produce a Service Manager event transaction. At least one rule must evaluate to TRUE or the event will not be generated.

You can generate multiple events from a single email. The event type to be generated from a particular email has traditionally been specified by coding one of the tags SC_type: or SC_event:. If one of those tags is present in the email 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 email, 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 email 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 email.
# # PMO.MAP
#
# This is for PMO events going TO Service Manager 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 Service Manager events

Service Manager 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 Service Manager 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 Service Manager 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 Service Manager 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 Service Manager 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.

Table 8 Field Definition Operators

Operator	Description
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 email 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.

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(...)

Table 9 Operators and Functions

Operator	Description
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 Service Manager array. This function concatenates the data with | symbols which denote array to Service Manager 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 Service Manager's eventin queue and all these four description lines will appear in the Service Manager'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 email equals some value.

Table 9 Operators and Functions

Operator	Description
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 Service Manager'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 email. 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 email 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.

6 SCAuto Pager

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

This chapter explains how to set up and use SCAuto Pager.

Topics in this chapter include:

Overview

Service Manager Event Services includes a standard page event and an `scauto.page` application which generates the page events. For syntax and usage of this event and application, refer to the Event Services online help.

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 pager from the Service Manager contacts database is converted into an external pager command. The pager 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 SCAuto Pager also provides support for two way paging, if the paging system used provides this capability. For example, with SCAuto two way paging, a 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 Pager.

Components

SCAuto Pager comes with the following standard components:

- `scpager` (`scpager.exe` for Windows)
  - The Pager executable.
- `scauto.msg`
  - Contains messages printed by the Pager and other SCAuto interface tools. This is optional, and may be customized for use with languages other than English.
- `scauto.dll` (Windows only)
  - The SCAuto run-time library.

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 only)
  - Sample command scripts.
- `SCenter.ini`
  - Sample initialization sections to support two way paging.
- `spawn.exe` (Windows only)
  - Helper program to simplify integration with pagers.

Operation

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

`scpager -server:<host.service>`

SCAuto pager parameter usage:

- True or false parameters (Boolean) take 1 or 0 after the colon to turn them on or off, respectively.
- Include the hyphen before the parameter name when using it on the command line.

Omit the hyphen before the parameter name when using it in the `.ini` file.

Omit the angle brackets around the parameter value.

Table 10 scpager options

Option	Description
-server:<host.service>	Host name and service name of the SCAuto listener server. This should be the same as the scauto: parameter used by the scautolistener server, in the host.service format. The default value 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 the default value is **`ServiceCenter/pager.sh`** (for Unix) or **`ServiceCenter\pager.cmd`** (Windows) 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).
-debug:<0/1>	**`1`** = Turns on debugging mode (command is printed before executing, and events are not deleted after being processed).
-trace:<0/1/2>	**`0`** = Off (Default) **`1`** = Print evsysseq for each page event. **`2`** = Also print several messages for each page event.

Table 10 scpager options

Option	Description
-sc_reconnect:<0/1>	Allows SCAuto Applications to retry to connect to Service Manager when Service Manager is restarted. **1** = Retry connecting to Service Manager. **0** = Do not retry connecting to Service Manager. (Default) This parameter works together with -**reconnect_interval** and - **numberof_retry_connect**. This parameter impacts the K (kill) command in Service Manager's system status list. The K command closes the connection between Service Manager 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 Service Manager shutdown. So, SCAuto Pager tries to reconnect even after a K command.
-reconnect_interval:<n>	Number of seconds to sleep between retrying. The default value is **120** (two minutes).
-numberof _retry_connect: <n>	Number of retries before exiting. The default value is **10**. **0** = Retry forever.
-noerror:<0/1>	**1** = Prevents creation of error events.

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

Table 11 Two-way Paging Options

Option	Description
-input:<file>	Name of a file to be monitored for incoming events. For more information, see Two Way Paging on page 81.
-checkpoint:<file>	Name of a file to use for checkpointing the input event file. The default value is `pager.chk`.

When running, `scpager` will retrieve page events from Service Manager 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 Service Manager. 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 Service Manager. Knowing how to send a page outside of Service Manager simplifies knowing what values are needed within Service Manager.

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

Table 12 TelAlert Fields

Field	Definition
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`).

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. For details on how to modify the behavior of the Pager, see Customization on page 79.

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).

Customization

If the TelAlert paging system is not being used, or it is desired to process events differently than the default, customizing the Pager 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 Pager, 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 Service Manager page event fields. Each argument is quoted, as it may have embedded white space (with single quotes for Unix and double quotes for Windows). 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. For more information, see Two Way Paging on page 81.
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. For more information, see Two Way Paging on page 81.

- 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.
- 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 Pager can monitor a file for incoming events, a capability which allows a two way paging system to communicate with Service Manager 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]
    .
    .
    .
{Service Manager}
Active=True
Shell=""
Command=${TelAlertCfg} ...
Reply=$(TimeStamp) ...
    ...
[Responses]
    .
    .
    .
{Service Manager}
NumReplies=5
Acked=1
AckedHold=3,5
NotAcked=2
Released=4
Reply1=Yes
Reply2=No
Reply3=Hold
Reply4=Release
Reply5=Info
    .
```

.
.

For Windows, 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 Service Manager directory.

Customizing two way paging

The standard Service Manager 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 Service Manager. These optional fields are:

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

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 Pager 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 online help 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 Service Manager'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 Pager would then create this event within Service Manager, and the 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, 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.

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 Service Manager is attached to the Problem Management application.

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. For more information, see the Format Control section of the Service Manager online help.

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

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 ticket by adding a new event called pageresp to the eventin table. This event, calls the `scauto.problem` application to update problem PM390.

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 Service Manager. The Event Registration table is used to register all events. When a response is received, the problem is updated with that response.

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 online help.

Errors

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

The SCAuto Pager 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 Pager's log file.

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 Service Manager, 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 Pager:

```
#!/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=${TELALERTBIN:-/usr/telalert}/telalertc
# Where to log commands.  This should not be the same as a
file used
# by scpager or any other application.
LOGFILE=${TELALERTDIR:-/usr/telalert}/ServiceCenter/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 not need to change anything below this point.
#
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)
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 version of SCAuto Pager:

```
@echo off
rem
rem Process Service Manager 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 Responses 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 parameters
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 info 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)
if %PARM_EVENT% == "" goto submit
if not %ACKWAIT% == "" set COMMAND=%COMMAND% -ackwait
%ACKWAIT%
if not %RESPONSE% == "" set COMMAND=%COMMAND% -response
%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
```

7 SCAuto Fax

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

This chapter explains how to set up and use SCAuto Fax.

Topics in this chapter include:

Overview

The FAX 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 systems and the Replix fax management system from SoftLinx Inc. on Unix systems.

Components

SCAuto FAX is equipped with the following components:

- `scfax` (`scfax.exe` for Windows)
  - The FAX executable.
- `scauto.msg`
  - Contains messages printed by the Fax and other SCAuto interface tools. This is optional, and may be customized for use with languages other than English.
- `scauto.dll` (Windows only)
  - The SCAuto run-time library.
- `fax.sh` (Unix) or `fax.cmd` (Windows)
  - 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 only):

- `lanfax.exe`
- `lfs32_50.dll`

Operation

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

`scfax -server:<host.service>`

SCAuto Fax parameter usage:

- True or false parameters (Boolean) take 1 or 0 after the colon to turn them on or off, respectively.
- Include the hyphen before the parameter name when using it on the command line.
- Omit the hyphen before the parameter name when using it in the `.ini` file.
- Omit the angle brackets around the parameter value.

Table 13 scfax command options

Option	Description
-server:<host.service>	Host name and service name of the SCAuto listener. This should be the same as the scauto: parameter used by the scautolistener in the host.service format. The default value is **`scauto`** or **`12690`** if that service does not exist.
-command:<name>	Name of the command to execute for each fax event. The default value is `fax.sh` (or Unix) or `fax.cmd` (Windows) in the same directory as the **`scfax`** file.
-log:<file>	Name of file to use for logging and error messages. The default value is `scfax.log` on Windows, and standard error output for Unix.
-debug:<0/1>	**`1`**: Turns on debugging mode (command is printed before executing, and events are not deleted after processing).

Table 13 scfax command options

Option	Description
-sc_reconnect:<0/1>	Allows SCAuto Applications to retry to connect to Service Manager when Service Manager is restarted. **1** = Retry connecting to Service Manager. **0** = Do not retry to connect to Service Manager. (Default) This parameter works together with the **-reconnect_interval** parameter and the **- numberof_retry_connect** parameter. This parameter impacts the K (kill) command in Service Manager's system status list. The K command closes the connection between Service Manager 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 Service Manager shutdown. So, SCAuto Fax tries to reconnect even after a K command.
-reconnect_interval:<n>	Number of seconds to sleep between retrying. The default value is **120** (two minutes).
-numberof _retry_connect: <n>	Number of retries before exiting. The default value is **10**. **0** = retry forever.

When running, scfax retrieves and processes fax events from Service Manager. 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 uses the Replix fax management system from SoftLinx Inc. by default. Under Windows, the LanFax system from Alcom is used by default. Other fax systems will be supported in the future. Refer to the documentation that came with the fax management system for information about installation and usage.

- 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 scfax is run. There is no default, because there is no standard location for this file.
- 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 Manager, 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 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, 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 Service Manager fax event fields. Each argument is quoted, as it may have embedded white space (with single quotes for Unix and double quotes for Windows). 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.

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

A Trouble Shooting

This appendix explains what to do if you have having specific problems with your SCAuto Applications.

If the connection fails between the SCAuto server (scautolistener) and the SCAuto Application, check the following:

- Is scautolistener running on Service Manager? Use the Service Manager 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 scautolistener daemon logs to the `sm.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 Service Manager server is required, and the **`scauto:keyword`** specified in the `sm.ini` file must match this name.
  - If it is running check the service name specified in the `sm.ini` file on the Service Manager platform for the **`scauto:keyword`**. If the keyword is not specified scautolistener, the default value is the services name scauto. This will be useful in the next step.
- Are the TCP/IP specifications correct on the SCAuto server?
  - Check the host and service specification. Are they specified and do they match the host and service name of scautolistener? Check with your network administrator.
  - Check the specifications in the passed parameters and make sure they all match. (For example, **`scmapi -server:host.service`**).
- Is there connectivity between SCAuto listener server and the SCAuto Application?
  - Ping the SCAuto listener server from the SCAuto application server. If this fails and all TCP/IP specifications are correct, contact your network administrator for assistance.

If there is an event in the eventin file but no RAD application is invoked (for example,there is email event but the mail was not delivered), check the following:

- Is the Event Scheduler running on Service Manager? Use the Service Manager 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 HP 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 online help in regard to schedule record specifications.

If there is no event in eventout file but the RAD application was invoked (for example, 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 an event. Refer to the Event Services online help.

  If this succeeds the Event Services RAD function is operable, and the problem is associated with the Service Manager application (such as Problem Management), or the specification(s) in the contacts table.

If there is an event in the eventout file but the external SCAuto Application was not invoked (for example, email event exists but mail was not sent), check the following:

- Is there connectivity to the SCAuto listener server?
  - Ping the SCAuto listener (scautolistener) server from the SCAuto application's server.

    If this fails and all TCP/IP specifications are correct, contact your network administrator for assistance.
  - Manually test your mail specifications by sending and receiving mail using your mail account specifications and logon. (e.g. **`scmail -mailbox user1`**)

    If this fails contact your mail administrator for assistance.

- Is SCAuto application active?
  - If it is not running check the SCAuto application logfile 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 there is no event in the eventin file but the external SCAuto Application event occurred (for example, mail was sent but not received in Service Manager), check the following:

- Is there connectivity to the SCAuto listener server?

  Review Connection between the SCAuto listener (scautolistener) and the SCAuto Application (client) trouble shooting section.

- Is the SCAuto application 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 Service Manager logfile for errors.

    Correct errors and, restart using appropriate command with the debug option if available.

    Recheck the logfile and correct errors and retry.

There is no event in the eventout file but the RAD Application was invoked (for example, a problem was opened and an email event was expected), check the following:

- Review the Basic Trouble Shooting and Using Format Control to Send E-mail Messages sections in the Event Services online help.

There is no event in the eventout file and the SCAuto Application was not invoked (For example, fax event and fax not sent)

- 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 online help.

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).
- Service Manager log and Scheduler log(s) if specified.
- Service Manager msglog for affected applications, or services.
- Service Manager print of agent status (Event Services Menu).
- Unloaded Event records relevant to errors.
- Unloaded filter specifications if relevant.

Index

F

G

I

K

L

M

N

O

P

R

S

T

U

W