ServiceCenter™

Release 1.4 & 2.0

SCAutomate Applications for SunNet

July, 1997

This edition applies to releases 1.4 and 2.0 of the licensed program

ServiceCenter™

and

SCAutomate™

June 1997

Table of Contents

Preface

Overview

SC Automate (SCAuto)™ for SunNet Manager, from Peregrine Systems, Inc., is one of a family of ServiceCenter automation products designed to enhance our HelpDesk and Inventory applications, and provide HelpDesk facilities to the SunNet user.

SCAuto-SunNet is composed of component daemons that handle inventory and problem reporting automation, as well as facilities to integrate your ServiceCenter applications into SunNet's operational environment. In running on the SunNet manager platform, SCAuto-SunNet utilizes supplied API's and standard operating facilities. No modifications are required to the network manager.

Integrated operational communications with ServiceCenter require the ServiceCenter client software, that allows SCAuto-SunNet to invoke a ServiceCenter client. You must have ServiceCenter client support on your network manager platform (which matches your server), and ServiceCenter application level A9401 or later (which contains the ServiceCenter Event Manager) Refer to *Chapter 2, Installation*, for more information on compatibility

This manual is organized as follows:

- **Chapter 1 Introduction** - A brief overview of Topology and Problem Management, Operations Integration and System Flow using the SCAuto for SunNet tool.
- **Chapter 2 Installatio**n - A summary of the installation process, including install verification and SCAuto modifications.
- **Chapter 3 Operations** - Detailed steps for starting and stopping SCAuto for SunNet, and descriptions of system utilities.
- **Chapter 4 Using SCAuto-SunNet** - An overview of system and knowledge requirements, application screens, menu fucntionalities and system navigation.
- **Chapter 5 Event Management** - A brief introduction to ServiceCenter Event Services, including major components used by SCAuto-SunNet.

SCAuto-SunNet Support Information

This chapter provides the information necessary to obtain Peregrine support for the SCAuto-SunNet product.

General Problem Isolation

Some common problems can occur when executing SCAuto-SunNet for the first time, changing platforms, etc. Use the following suggestions to isolate, or fix your problem prior to notification:

- SCAuto-SunNet ServiceCenter operational function relies on ServiceCenter client facilities in order to function. Run a ServiceCenter client from the $SNMHOME/ipas directory. This should have been set up during installation via a symbolic link (**ln -s...**) command.

  The ServiceCenter client command is: **scclient login** *userid* **-G**. If the client window appears, you have ServiceCenter client server connectivity. If the window does not appear, your client server specifications may be incorrect. Revalidate your ServiceCenter client installation. If you are still encountering problems, report problem according to ServiceCenter client server reporting procedures.
- Check the sn*IPAS_log* for any error messages.
- SCAuto requires the SCAuto base be up and running on the ServiceCenter server platform and its service name specified in /etc/services.
- Ensure permissions on $SNMHOME/ipas are consistent with the execution user, group. Also, ensure this directory is in your PATH environment variable.
- Compare the *SunNet event.log* to the *snIPAS.trapd.log*. The *snIPAS.trapd.log* should contain all log records from the SunNet log that correspond to the *snIPAS_keywords* specified. If not you may have a problem with the snIPAStd daemon. Ensure the daemon is active and that it has permissions to $SNMHOME/ipas.
- In some cases, system messages useful for are lost when SCAuto-SunNet daemons are executed under the SunNet Manager Console Window. Run the daemons under a user with proper permissions as follows:
  $SNMHOME/ipas/snIPASd& (SCAuto-SunNet ServiceCenter services daemon)
  – or –
  $SNMHOME/ipas/snIPAStd& (SCAuto-SunNet trap daemon)

Contacting Peregrine Systems

Peregrine Systems Inc. provides support for all SCAuto-SunNet users. Before contacting Peregrine Customer Support, review the following

section, *Obtaining Required Data/Information*, to see if additional data is required to help diagnose the problem.

You can contact Peregrine Systems support as follows:

- For SCAuto-SunNet problems or information that is needed immediately, call Peregrine Customer Support at (800 638-5231 or (619) 431-2400.
- For questions or information regarding SCAuto-SunNet, use a written FAX or email.

  Send all SCAuto-SunNet FAXes to (619) 431-0696.
- For information that was requested of your installation that is on tape, cartridge, etc., send to:

  Peregrine Systems Inc.

  attn: SCAuto-SunNet Support

  12670 High Bluff Dr.

  San Diego, CA 92130

Obtaining Required Data and Information

This section provides detailed instructions for gathering data and information needed for the Peregrine support staff to resolve your problem in the most efficient manner possible.

Environmental Information:

- ServiceCenter Release
- SCAuto-SunNet Release
- Operating System Release (i.e., Solaris, SunNet Mgr)
- Type of hardware base SCAuto-SunNet is running on
- Any error messages or error logs.

Error logs and files that the Peregrine support staff needs are listed on the next page.

The Peregrine Support staff can utilize the following error logs and files to resolve an SCAuto-SunNet problem:

- *$SNMHOME/ipas/snIPAS_chkpt*
- *$SNMHOME/ipas/snIPAS.trapd.log*
- *$SNMHOME/ipas/snIPAS_log*
- The output from a *$SNMHOME/ipas/sntopodump* command.
- If the problem resulted in a core dump, the resulting core file is helpful in determining the problem.

Error Messages

Messages printed to $SNMHOME/ipas/*snIPAS_log* are formatted as follows:

<*time*> <*module*> <*action code*>: <*message*>

where:

<*time*> - is the time when the message was printed (using the local time).

<*module*> - is the program module causing the message.

<*action code*> - is one of the following single letter codes:

A - operator action may be required.

W - warning message.

I - informational message only.

C - checkpoint record.

Error Return Codes

The following error return codes may be indicated in a message:

101...199

Indicates a client/server problem with ServiceCenter. Verify that ServiceCenter is operating and is the same release level as SCAuto-SunNet.

800

No connection established. Could indicate that initial connection to ServiceCenter failed. Verify ServiceCenter is running.

804

An error was printed to standard error. These messages are only displayed if you ran the daemons manually (not under OpenView).

807

Malloc failed. Check the standard error.

811

Error opening communication file. Check standard error.

814

Error adding event to ServiceCenter. Internal error.

Other errors indicate a systems error. If these occur, gather the necessary information and log files (sn*IPAS_log*, sn*IPAS_chkpt*, and *snIPAS.trapd.log*), then contact Peregrine Customer Support.

Sending Files to Peregrine

Files may be sent to Peregrine Systems either via magnetic tape or electronic mail. If multiple files are being sent, store the files in the *tar* format. Compressed files are acceptable.

Files can be sent on 8mm or 4mm magnetic tapes. If you are sending files with electronic mail, use the UNIX **uuencode** utility. Other formats or methods of transport must be arranged with Peregrine Customer Support.

Chapter 1 Introduction

Overview

SCAuto-SunNet major components include:

- A topology/client daemon which performs inventory functions.
- A trap daemon that gathers problem information and dynamic inventory changes.
- Operational integration facilities for the supported platforms.
- Utility programs to refresh inventory and archive SCAuto-SunNet files.

Figure 1-1 provides an overview of SCAuto components.

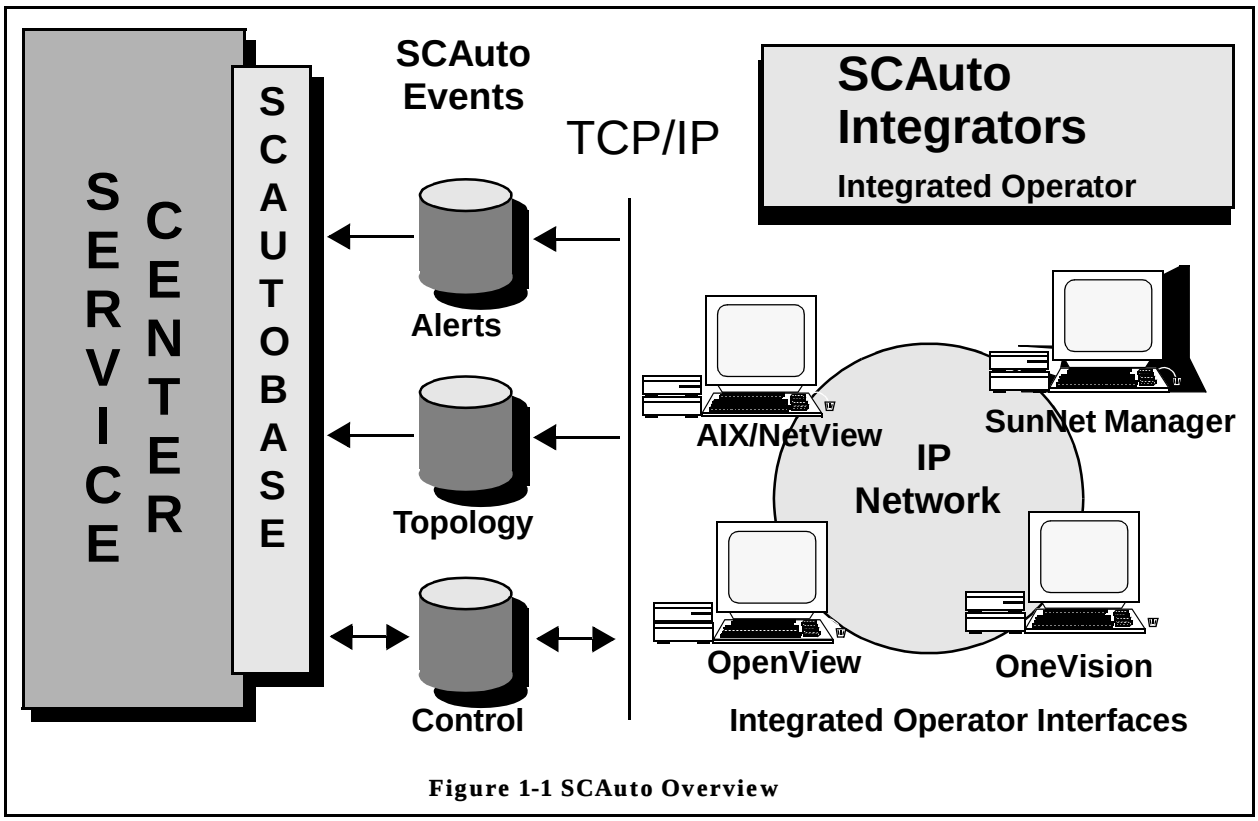

Figure 1-1 SCAuto Overview

Topology Management Overview

The primary function of the topology component, or daemon, is to create and maintain inventory records in ServiceCenter. The inventory records created correspond to the objects discovered by the network management platform. An inventory record is created and maintained for each device type occurrence in the network with the corresponding connections. Each record is maintained as a ServiceCenter device record, with special fields for the connection and control of information. The connection data is important to graphical, path determination, outage analysis, and dependency propagation applications. The connection relationships maintained are:

- Container (contained in relationship)
- Hierarchical (parent, child relationship)
- Point-to-point (peer relationship).

SCAuto-SunNet topology management transforms the SunNet management elements into corresponding device types in ServiceCenter. Different device types are required to represent the elements of the various networks. The major ServiceCenter inventory device types created by SCAuto-SunNet for SunNet manager are found in the ***elements.schema*** file in the ***$SNMHOME/struct*** directory of SunNet manager. ServiceCenter device types are represented as the SunNet element type minus the prefix (e.g. **component.router** element type in SunNet becomes the **router** device type in ServiceCenter).

The other fields extracted from the element are assumed to be of the form: *Name, IP_Address, User, Location* and *Description*. If the element is of a different form you must register it under the supplied alternate forms in the ***snIPAS_schema*** file.

The IP environment normally provides specific configuration information through the MIB (Management Information Base). If MIB information is maintained as a function of network administration, ServiceCenter inventory records can be maintained automatically Those records will contain current contact and location information. The dynamic changes to IP inventory are kept current by topology event records created by the network manager discovery/management programs. The dynamic updates ensure the IP network is up to date and the current status is reflected in your ServiceCenter database.

All topology information is forwarded dynamically to ServiceCenter event service facilities, providing the automated network discovery information for any ServiceCenter application. The dynamic update, add and delete of network component information in ServiceCenter results in more accurate and timely information in ServiceCenter databases. This process also provides a reduction in the labor intensive process of entering and maintaining this data.

SCAuto-SunNet topology utilizes standard inventory **add** and **update** events as described in ***ServiceCenter Base Utilities Chapter 19***. The Event Manager is the ServiceCenter component which maps the input events and gives control to the RAD applications that process

the events. In the case of inventory, an *icma* (inventory control management add) standard event is created and placed in the *eventin* file. The event scheduler reads the *eventin* file and maps the ServiceCenter event data into the device and attribute files. The scheduler would then perform a background inventory add operation. Refer to ***ServiceCenter Base Utilities Chapter 19*** for more information on the Event Manager and its standard facilities.

A firm understanding of the ServiceCenter Event Manager is helpful in reviewing the subsequent table which maps various element field names to ServiceCenter inventory records. The tables provide a quick reference of SCAuto-SunNet generated fields and their corresponding Event Manager eventmap fields in ServiceCenter.

Table 2-1 provides a cross reference of SCAuto for SunNet element object field names to ServiceCenter Event Manager mapping fields for an inventory event.

Table 1-1. Object Fields to ServiceCenter Mapping Field

Object *field name*	**eventmap field**
Name	logical.name
Element type - prefix	type
Contact	contact.name
SCAuto generated:	last.update
sc.ini #@ default location	location
Location	location
Name-network suffix	network.name
IP_Address1	end.point.1
IP_Address2	end.point.2
Generated by SCAuto	protocol
IP_Address	protocol.addr
SCAuto Generated SN - hostname	updated.by
Description	description Many vendor MIB description fields contain valid information which could be parsed and mapped to more appropriate ServiceCenter inventory field names

Problem Management Overview

The Problem Management component is comprised of a set of programs that interface with the network manager to dynamically open, close and update ServiceCenter problems. Problem actions are based on SNMP traps. Traps are issued in SNMP to notify the network manager of a specific event for a device or software. An event normally indicates a problem. SCAuto-SunNet allows user-specified keywords to identify specific traps for opening or closing a problem, or updating inventory.

Specialized or global filters also can be specified on the ServiceCenter server to block problem reporting. Filtering can be based on: time of day, event type, event data, frequency of occurrence, thresholds, and time considerations. Because filters are specified on the server, all problem management components associated with the server are centrally administered. The Event Manager's standard facilities supply filtering formats to create, update and delete event management filters.

If a problem is not solved by keywords or standard filter specifications, user-written RAD functions or expressions can be used.

The SCAuto base TCP/IP environment is used to forward problem reporting information to the ServiceCenter server, including ***pmo*** (problem management open) and ***pmc*** (problem management close) events. The server's Event Manager passes the event data through the standard Problem Management application to open, update or close a problem.

Only one problem is opened per reporting component per SCAuto-SunNet problem manager. All subsequent traps received from a component with an open problem ticket, are considered an update or a close of the opened problem.

For example, an SNMP trap is received with a keyword indicating a problem should be opened in ServiceCenter. SCAuto-SunNet checks the Event Manager filters. If all is well, a problem open event (pmo) for the device is created and sent to the Event Manager, which opens a problem in ServiceCenter.

A subsequent trap arrives that also contains a keyword indicating a problem should be opened. Again, the filters are checked and a problem open event is sent to the Event Manager. The Event Manager checks the event and notes it is from SCAuto. A check determines if a problem for this specific device was opened by this SCAuto-SunNet problem manager. If a problem is open, it is updated. If no problem exists, a new problem is opened.

A trap containing a keyword indicating the problem has been resolved. SCAuto-SunNet creates a problem close event (pmc) and sends it to the Event Manager. The Event Manager closes the problem for the specified device.

Problems opened by SCAuto-SunNet contain the following information (if available in the SNMP trap):

- Problem origin (reporting network manager).
- Date/time reported.
- Failing component.
- Description of the problem.

Operations Integration Overview

The Operations Management components provide assorted facilities that integrate ServiceCenter into the window or console facilities of your network management platform. This integration provides a uniform operator interface, and reduces the operator entered information required in utilizing ServiceCenter facilities from the network management platform.

Integration is done by including ServiceCenter functions on your network management platforms menus, and when possible, eliminating redundant operator entry of previously selected object information. ServiceCenter GUI client provides access to all ServiceCenter facilities and selected streamlined services.

Note: This capability is available only if the ServiceCenter client is installed, and the ServiceCenter cut-throughs are enabled when SCAuto-SunNet is installed.

The streamlined services allow you to bypass menus and application screens and invoke a requested application with passed variables to display a desired output window. For example, a Console operator selects a router element icon in the Console window and would like to request a list of ServiceCenter open problems for this element. The operator selects the ServiceCenter menu and chooses the open problems menu option. The selected router name is passed to a *streamlined* problem lookup. Any problems for that router are displayed in a separate ServiceCenter client GUI window.

Refer to *Chapter 4, Using SCAuto-SunNet*, for information on the facilities provided on a specific platform.

SCAuto-SunNet System Flow

SCAuto-SunNet runs as daemon processes under the userid of the console or database to be monitored. It can run as a standard foreground or as a background process. The foreground execution capability insures that all error messages are displayed and allows you to run as a standard user when testing.

The supplied method of starting the SCAuto-SunNet daemons is to select the ***_SCAuto_Start...*** function from the **Tools** menu. This selection runs *snIPASpm* the Process Monitor which starts the two daemons, sn*IPASd* and *snIPAStd*. Refer to Figure 1-2..The daemons run asynchronously to SunNet Manager and can be started or stopped with no loss of information when restarted.

The *snIPAStd* trap daemon first repositions itself and then monitors the SunNet Managers ***event.log*** file. As events are added, they are examined to determine if any SCAuto-SunNet keywords are present. When a trap or event is selected, it is analyzed, reformatted and time stamped. A standard record is written to the sn*IPAS.trapd.log* file for processing, and the daemon waits on the next event.

When the second daemon *snIPASd*, the topology client daemon, is started it connects to ServiceCenter via the SCAuto base TCP/IP facility. Upon connection, *snIPASd* daemon ensures that the user is

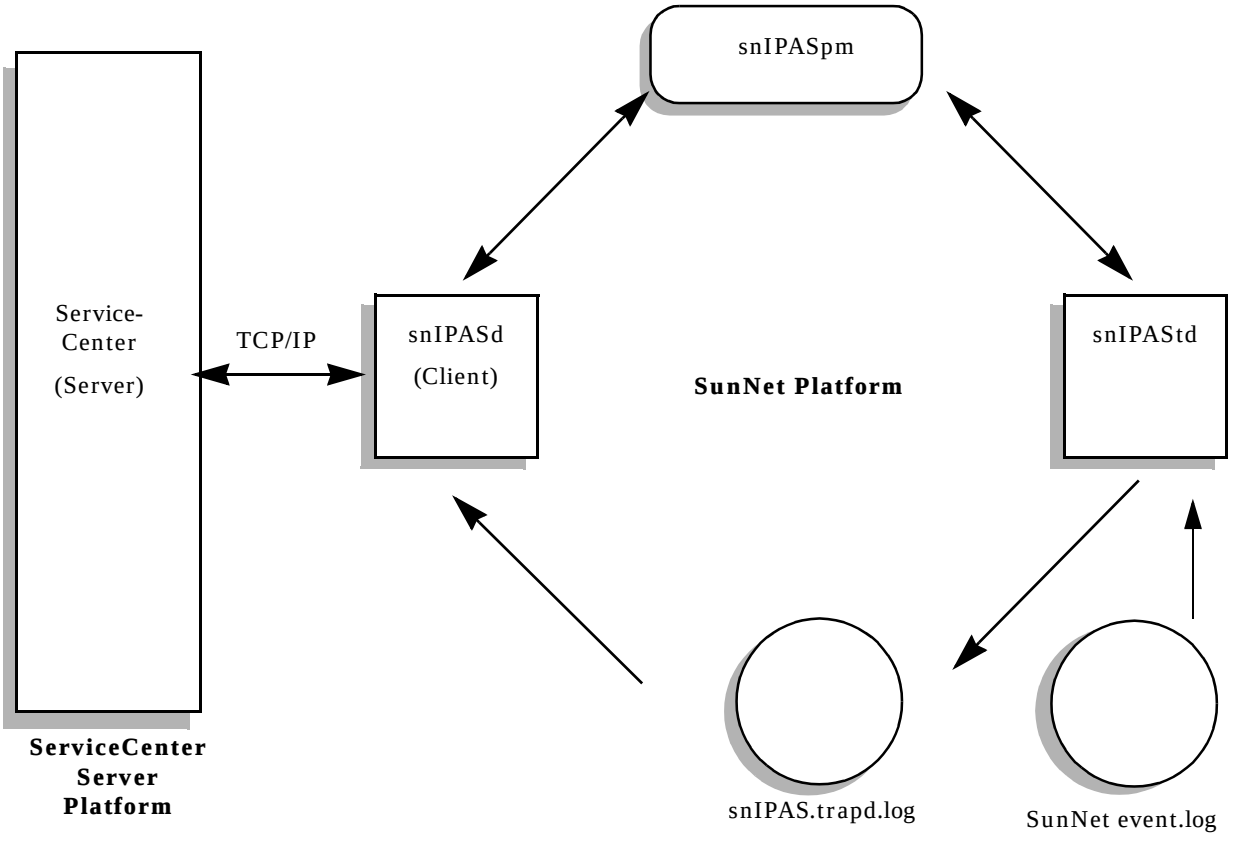

Figure 1-2 SCAuto-SunNet System Flow

licensed for use and opens or creates its required files. One such file, the *snIPAS_log* file is created or updated with each daemon execution and is especially useful in problem determination.

Next, the daemon determines if a refresh is requested (via the *snIPAS_chkpt* file), or a resynchronization is needed. If a refresh is requested, all supported objects are read and objects are updated or added in ServiceCenter. If resynchronization is requested, a checkpoint record is read and the *snIPAS.trapd.log* is positioned. Log records are then processed from the checkpoint forward to open or close problems, or update inventory. Once the initial processing is completed, the daemon waits for a new problem or inventory event to occur.

Whenever *snIPASd* has updated the *eventin* table in ServiceCenter, an Event Manager scheduler examines the new records and performs the appropriate actions to open, update or close a problem ticket, or update inventory files. This sequence continues until the daemons are killed or stopped using the ***_SCAuto_Stop..*** selection on the **Tools** menu.

Chapter 2 Installation

Overview

This chapter provides information, instructions, and verification procedures for installing ServiceCenter SCAuto for SunNet.

Installation Notes

- SCAuto-SunNet requires approximately 10 MB of hard disk space.
- SCAuto-SunNet is a subdirectory on your ServiceCenter client/server tape.
- If you plan to use ServiceCenter functions from SunNet please validate that a standard ServiceCenter client is installed and runs with your ServiceCenter server from the platform you intend to run SunNet.
- SCAuto-SunNet must run on the same platform as your network manager.
- Obtain an SCAuto-SunNet authorization code from your Peregrine Systems Account Executive or Customer Support and restart your ServiceCenter server with this code in the ServiceCenter servers current *sc.ini* file.
- The installation asks for the name of a user that will own the SCAuto-SunNet files. If you wish to create a new user for this, do so before starting the installation. It is suggested that this be the same user that owns the ServiceCenter files. The *root* user cannot be used for this purpose. You can also specify a group for the SCAuto files by entering **user.group** when prompted for a user name.

Installation Procedures

This section provides the SCAuto-SunNet installation instructions.

1. SCAuto-SunNet is a subdirectory on your ServiceCenter client/server tape. Use the **tar** command to restore your ServiceCenter client/server tape to the desired platform.
2. Perform the ServiceCenter client installation procedures regarding any kernel parameters, etc. (If only ServiceCenter Functions from SunNet are required).
3. If the ServiceCenter client is installed, test a ServiceCenter client and ensure it can connect to the desired server from the network manager platforms (Only ServiceCenter Functions from SunNet are required).
4. Obtain an SCAuto-SunNet authorization code from your Peregrine Account Executive or Peregrine Customer Support at (800) 638-5231.
5. Restart your target ServiceCenter server with the proper authorization code.
6. The SCAuto/snm product tape subdirectory contains all files required to install and run SCAuto for SunNet. You may need root authorization to complete the installation. An install program (**INSTALL**) is provided which creates your execution directories. Once you have the proper file authorizations execute the following command:

   /<tar directory>/SCAuto/snm/*INSTALL*

   Note: Ensure you change to the *<tar directory>/SCAuto/snm* directory before running the INSTALL program. If you are not in this directory, the installation will fail.

7. The install application INSTALL prompts you for the answers to some key requirements. The responses must be accurate for successful operation.
8. Review the output produced from INSTALL and correct any error situations, permissions, etc.. You can rerun INSTALL without a problem. Even though INSTALL always attempts to do a complete install, no problems will occur on steps that were previously accomplished.
9. Update the *sc.ini* file in *$SNMHOME/ipas* with your server name and any ServiceCenter client specifications. See *ServiceCenter Quick Installation Guide, Client/Server For UNIX*.

   You should also update your.*sc.ini* with your SCAuto specifications at this time. The SCAuto parameters are identified with a **#@** prefix:

   #@scinv:yes|no

   Do you want the automated inventory functions? The default is **yes**.

#@scauto:

The service name in /etc/services of your SCAuto base. If the base is on the SunNet platform only the service name is required. If on a separate platform specify hostname.servicename

#@sclog:

The directory of the SunNet event.log normally /var/opt/SUNWconn/snm.

#@scprob:yes|no

Should SCAuto automatically open and close problems? The default is **yes.**

#@scdatefmt: 1|2|3

The date format corresponding to ServiceCenter date formats to be used by SCAuto-SunNet.

d**#@scevsufx:event suffix**

Alphanumeric suffix to be appended to events created by this SCAuto. The default is no suffix.

Note: If you specify a suffix your event tables in ServiceCenter, Event Manager must be modified to include a set of new event types, *pmosuffix*, *pmcsuffix*, *icmasuffix*, and *icmdsuffix*. This may require you to provide some of your own RAD coding. Therefore, do not specify a suffix unless you fully understand the implications.

#@sclocation:default location

The specified location is utilized by SCAuto when no location is specified in the MIB.

#@sccategory:problem category

The specified category is used by SCAuto on problem open and close.

10. Define symbolic links to ServiceCenter binaries(if not done by installation script):
    - **ln -s /<tar directory>/RUN/scclient $SNMHOME/ipas/scclient**
    - **ln -s /<tar directory>/RUN/scgui.uid $SNMHOME/ipas/scgui.uid**
    - **ln -s /<tar directory>/RUN/scguimtf $SNMHOME/ipas/scguimtf**
11. Ensure that all network manager operators have $SNMHOME/ipas directory specified in their PATH environments.

Refer to the next section to verify the installation.

Installation Verification

This section includes installation checks and possible solutions for installation problems.

1. Start your ServiceCenter server.
2. To start SCAuto, enter:

   $SNMHOME/ipas/snIPASpm -start

   This command normally is executed when you select the ***SCAuto start*** option from your **SunNet Console Tools** selections. This command starts the SCAuto daemons snIPAStd, which is the SCAuto trap daemon, and snIPASd which is the ServiceCenter client daemon. Verify that the two daemons are running as follows:

   a. The snIPAStd daemon creates SCAuto log records in *$SNMHOME/ipas/snIPAS.trapd.log*. If log records are located in this file after events are detected, snIPAStd is operating correctly.

   b. Start the SCAuto Base on the ServiceCenter Server platform. The snIPASd daemon connects to the SCAuto Base and determines if the ServiceCenter server is licensed for SCAuto-SunNet. If so, the events are created and inserted in the *event.in* table in ServiceCenter server. If the *event.in* records are present at the server, check the Event Manager installation using the ServiceCenter Event Manager installation procedures.

   c. Verify that events are appearing in the *event.in* table by logging in as a ServiceCenter client and entering **db** at the command line.

   d. Enter **event.in** as the format name and press **<enter>** twice to display all the events.

   e. Search for the events with the **user-id** field of *SN-<hostname>* where hostname identifies the machine SCAuto is running on. If these records are found, SCAuto and ServiceCenter are communicating.

   If you are having problems with snIPASd connection, verify that the /etc/services file has your SCAuto Base specification and that the SCAuto Base is up and running on the ServiceCenter server platform.

3. If the ServiceCenter cut-throughs were enabled when SCAuto-SunNet was installed, verify the SCAuto ServiceCenter operational facilities by issuing:
   $SNMHOME/snm

   Your network manager platform user windows are displayed.

4. Verify the **Console Tools** menu bar contains the ServiceCenter options. Select a component on one of your network views and check the **Element Tools** menu for Service Center selections. Refer to the *Chapter 4, Using SCAuto,* to verify the numerous options in this facility.

   If the options are not available perform the following:

   - *Select* File and Load Management Database Option.
   - *Set* directory to $SNMHOME/ipas/Ipas.schema and load
   - Quit your SunNet Console session and restart.

   The ServiceCenter options should now be available.

You should become familiar with the utilities and filtering functions after the basic installation is complete. These post-installation capabilities are discussed later in this manual.

SCAuto Modifications

SCAuto dynamically creates four types of events in ServiceCenter:

- pmo (problem open)
- pmc (problem close)
- icma (inventory add)
- icmd (inventory delete).

The event type and subsequent processing is determined by the log record information in sn*IPAS.trapd.log*. The parse of the log record determines if the record is ignored (filtered), or what type of operation should be performed.

The file that determines the parse is *snIPAS_keywords*. sn*IPAS_keywords* contains four categories which relate to the four event types *PROBOPEN* (pmo), *PROBCLOSE* (pmc), *TOPOADD* (icma), and *TOPODEL* (icmd).

If the parse phrases in snIPAS_keywords do not contain the codes or descriptions of traps you want processed by SCAuto, you can update the specific category with a new *word*, *phrase*, or *generic specific code*. If you want SCAuto to ignore certain traps, you can remove a certain *word*, *phrase*, or *code* from this file.

You can also modify the configuration file in the network manger to produce unique trap records to be parsed by SCAuto.

The network manager *trapd* daemon receives the SNMP trap and converts the SNMP information into a *event.log* record. The log record is created from the variables received in the trap.

When using the sn*IPAS_keywords* file to specify unique words or phrases to create SCAuto events, review your *event.log* file and select the appropriate keywords for SCAuto events.

Please review the keywords that are shipped in the sn*IPAS_keywords* file at install to determine what modifications should be made for your operation. In conjunction with your filter specifications in the event manager you may not have to tailor initially. The keywords supplied are intended for standard IP trap and topology events.

Whenever you update the snIPAS_keywords file, you must stop and start the SCAuto daemons to pick up your new specifications.

Chapter 3 Operations

Overview

This chapter covers operation procedures for starting and stopping SCAuto-SunNet.

Starting SCAuto-SunNet

SCAuto-SunNet is normally controlled by **snIPASpm**, a supplied application.

1. Start the SCAuto base on your ServiceCenter sever platform
2. To start the SCAuto-SunNet daemons you may select ***SCAutostart*** from the **SunNet Console Tools** menu or issue the following command:

   $SNMHOME/ipas/snIPASpm -start

   You may need special permissions to execute this command.
3. You can also start the daemons outside of the snIPASpm process management by issuing the following commands:

   $SNMHOME/ipas/snIPASd

   $SNMHOME/ipas/snIPAStd

Stopping SCAuto-SunNet

This section provides the steps for stopping SCAuto-SunNet.

1. You can stop all agents by issuing the following command.

- To stop all agents, issue this command:

  $SNMHOME/ipas/snIPASpm -stop

2. To stop the daemons separately, first issue the following command to get the necessary data to complete a **kill** command:

   /bin/ps -e |grep snIPAS*

   Extract the snIPASd or the snIPAStd process ids, or both. Substitute those values in the following command:

   kill -9 *pid1* [*pid2*]

   Where *pid1* is the snIPASd process id, and *pid2* is the snIPAStd process id.

SCAuto-SunNet Utilities

This section describes the SCAuto-SunNet utilities and their operation.

SCAuto-SunNet Archive Utility

An **Archive** utility is provided to archive the *snIPAS.trapd.log* file, which can be deleted or put to external media. The sn*IPAS.trapd.log* could grow infinitely based on the disk available. Periodically, you should remove processed records and save the disk. The archive utility produces two files: an archive file containing all processed records, and a new sn*IPAS.trapd.log* with all unprocessed records.

In order to run the archive, the daemons (snIPASd and snIPAStd) must be stopped using the following command:

$SNMHOME/ipas/snIPASarc

The utility prompts you, then executes after you have responded. After the process is complete, a new date/time stamped archive file is placed in the *$SNMHOME/ipas* directory which can be removed, copied, etc.

SCAuto-SunNet Inventory Refresh Utility

The **SCAuto-SunNet Inventory Refresh Utility** allows the entire inventory or segments of the inventory to be refreshed in ServiceCenter. Initially, the inventory is created by the snIPASd daemon when the **#@scinv:yes** parameter is specified and a checkpoint (snIPAS_chkpt) cannot be located on the log tape (sn*IPAS.trapd.log*). You can update or recreate this inventory without impacting the snIPASd daemon and its running environment.

When snIPASd initializes, *snIPAS_chkpt* is opened. If it is not present or the record it contains cannot be found in *snIPAS.trapd.log*, the snIPASd performs a restart. A restart assumes that SCAuto-SunNet is to start over by initializing inventory and processing the log from this new point. This should only happen at the initial start of SCAuto-SunNet after installation.

The SCAuto-SunNet Inventory Refresh utility can run concurrently with the SCAuto-SunNet daemons (snIPASd and snIPAStd). The utility attaches to the specified server and creates inventory events (icma) to update or create the specified inventory.

The Inventory Refresh Utility can be started by:

$SNMHOME/ipas/snIPASr -s *ServiceCenter server* -t *inventory type*

Where:

-s *ServiceCenter server*

The ServiceCenter server you wish to update. This should be specified as **host.servname** if the server is on a remote host.

-t invtype|all

The specific inventory type to refresh or all types. Refer to the inventory tables in *Chapter 1* for specific inventory types created by SCAuto-SunNet.

For example, the following command refreshes the *router* type inventory records for the *IP* network in ServiceCenter server *sc01*.

```
$SNMHOME/ipas/snIPASr -s sc01 -t router
```

Note: The SCAuto Base must be running on the ServiceCenter server platform for the refresh utility to function.

Chapter 4 Using SCAuto-SunNet

Overview

SCAuto-SunNet provides an enhanced operator interface to ServiceCenter that integrates ServiceCenter facilities on the SunNet Manager Console. From a SunNet Manager window, you can access a number of ServiceCenter screens to gather information related to the current window or selected element.

This capability is only available if the ServiceCenter cut-throughs were enabled during the SCAuto-SunNet installation. If the cut-throughs were not enabled, you must start a client from the UNIX command line instead.

Note: SCAuto-SunNet operates under the A9503 or later version of ServiceCenter 1.3.x. Refer to the appropriate ServiceCenter documentation for more information on using ServiceCenter.

SCAuto-SunNet gives you access to ServiceCenter through two toolbars. The primary ServiceCenter tools appear within the **Console Tools** pulldown menu in a SunNet Manager window (Figure 4-2). Another set of tools appear within the **Element tools** right-mouse-button popup menu for a selected element (Figure 4-2).

Note: In order to access a ServiceCenter window, SunNet Manager must not be running from the *root* user account.

Important: Screen captures and functions may change from release to release. Reference the current ServiceCenter documentation for your specific platform for the latest operational details.

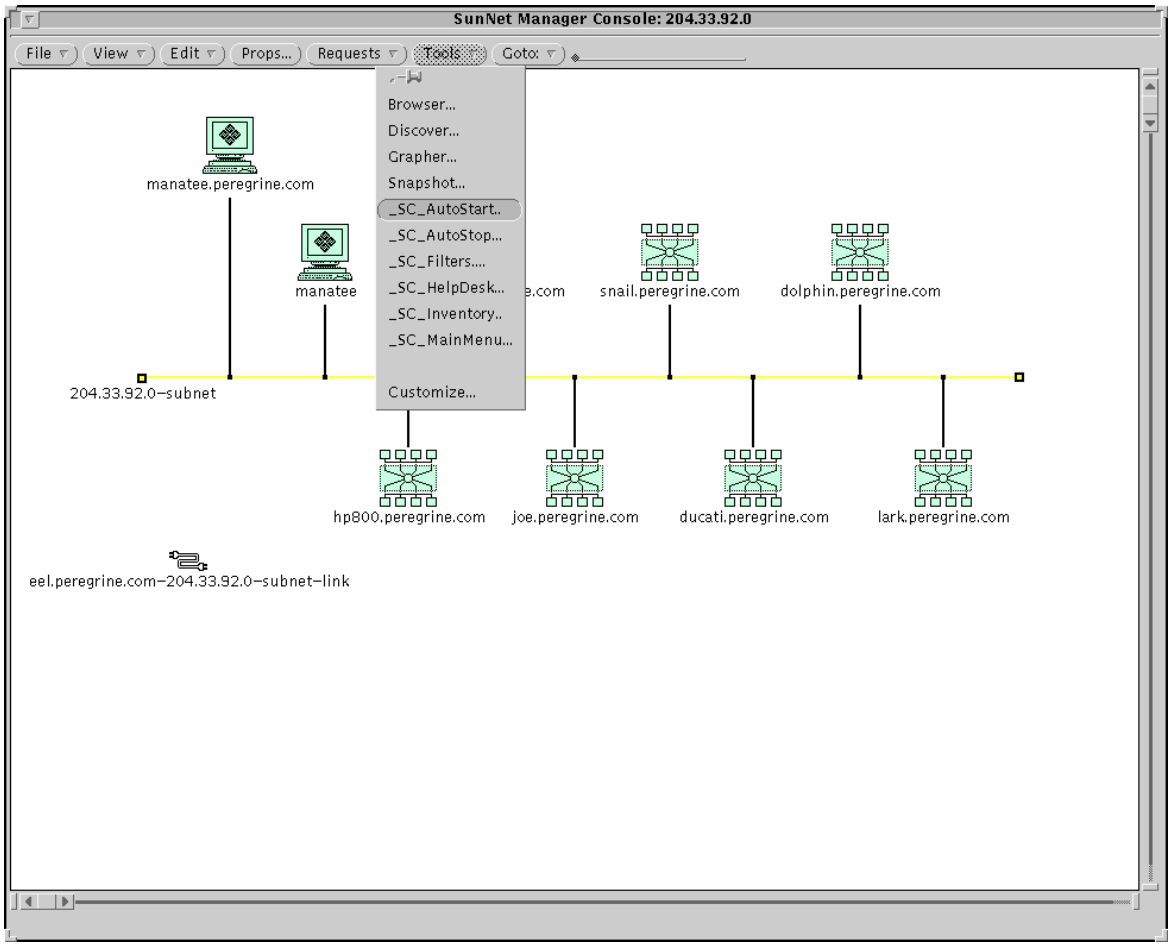

Figure 4-1. SCAuto-SunNet Primary Tools

Access the primary SCAuto-SunNet tools from the SunNet Manager **Tools** menu as shown.

Primary tools give you general access to ServiceCenter and allow you to control the SCAuto-SunNet daemons.

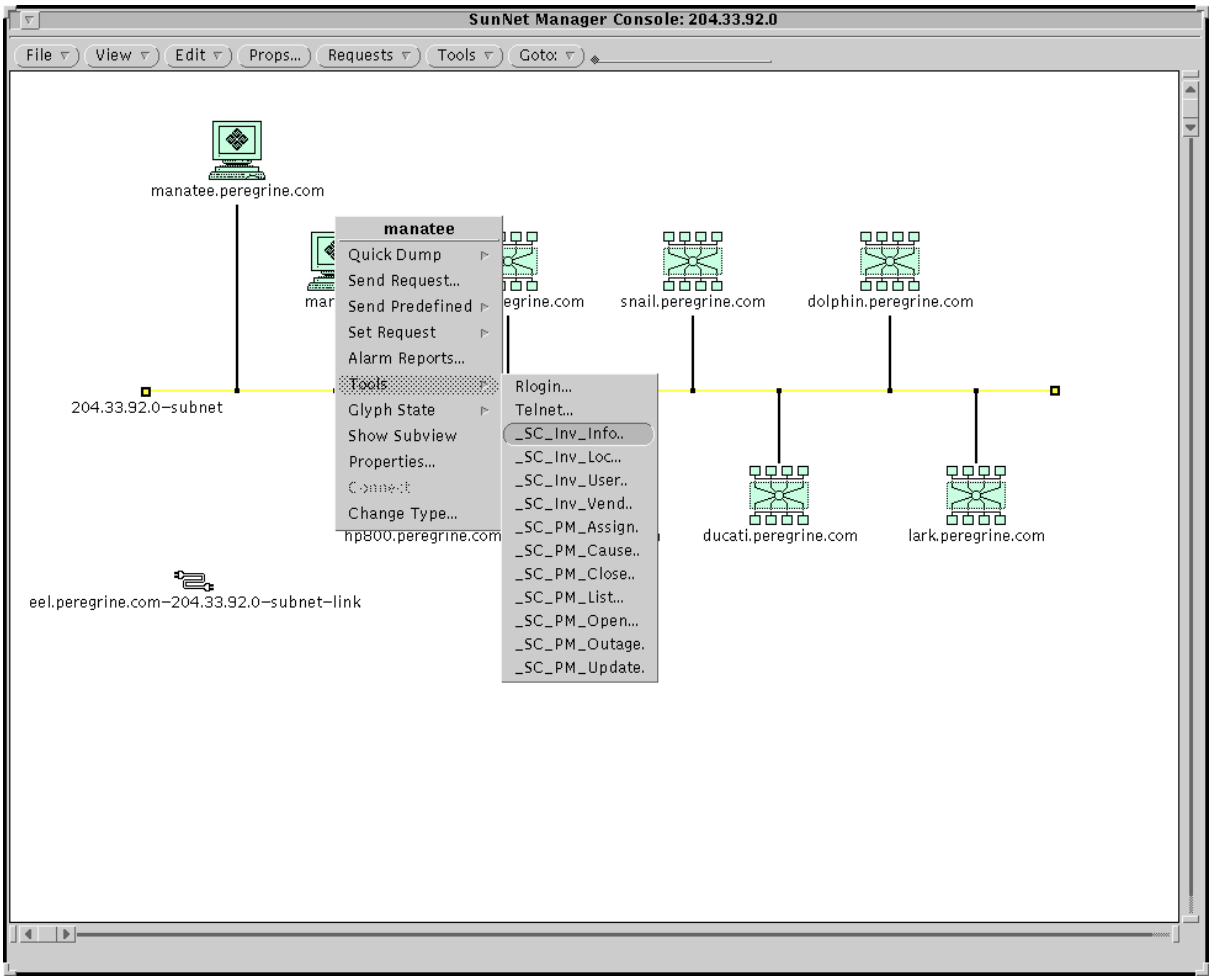

Figure 4-2. SCAuto-SunNet Element Tools

Access element-oriented SCAuto-SunNet tools by selecting and element an d clicking the right mouse button.

Element tools give you direct access to ServiceCenter functions that operate on the selected element.

Requirements

Before using SCAuto-SunNet, you should have a good working knowledge of the following:

- ServiceCenter applications.
- ServiceCenter Client Server.
- SunNet Manager Console operations.

Note: While some procedures for these applications are explained, others are referenced. Refer to the appropriate documentation for more detailed documentation.

Mouse Conventions

The mouse buttons are used as follows when operating ServiceCenter under SCAuto-SunNet, assuming a right hand mouse.

- Left (1) button - activates and selects the options from the **ServiceCenter** menu in the SunNet Manager windows. In ServiceCenter windows, button 1 is used to place cursor in a field.
- Middle (2) button - not active.
- Right (3) button - activates the popup menu in a ServiceCenter window. To select an option, keep the button depressed until the option is highlighted, and release the button.

ServiceCenter Menu Options

The ServiceCenter menu options take you directly to the ServiceCenter applications from SunNet Manager. These express services save the time of logging in and navigating through ServiceCenter to get to these applications. The following sections provide a brief description of the screens.

Note: While some of the ServiceCenter popup menu options are mentioned in this manual, you should refer to the ServiceCenter documentation for complete instructions on using the ServiceCenter options.

To use a ServiceCenter application under SCAuto-SunNet:

1. Select the **ServiceCenter** menu in the SunNet Manager window and select the appropriate menu option. Some ServiceCenter menu options are not available unless an object is selected in the SunNet Manager window.
2. Use the mouse and the tab keys to navigate through a screen.
3. To leave the application, select **F3 end** from the popup menu or press the **F3** key. This takes you to the previous screen or a logout screen.
4. When the *login.exit* screen is displayed (Figure 4-3), select **F1 EXIT** from the popup menu or press the **F1** key to exit the ServiceCenter session.

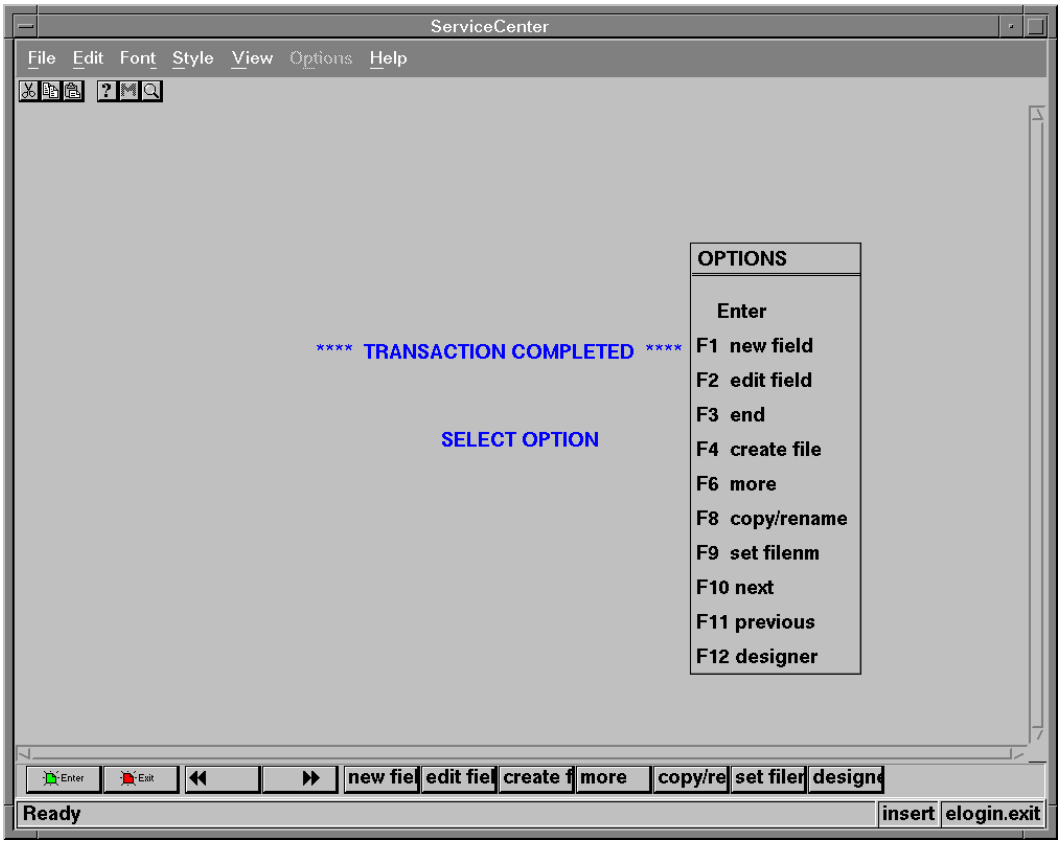

Figure 4-3. Login.exit Screen

Problem List(_SC_PM_List)

The ***Problem List*** menu option provides a list of problems open in ServiceCenter for the selected object. When this option is selected, a **problem list** is displayed (Figure 4-6). A popup menu provides options for the problem list.

To exit the screen, select **F3 end**. This takes you to the logout screen. Select **F1 exit** in the logout screen to end this ServiceCenter session.

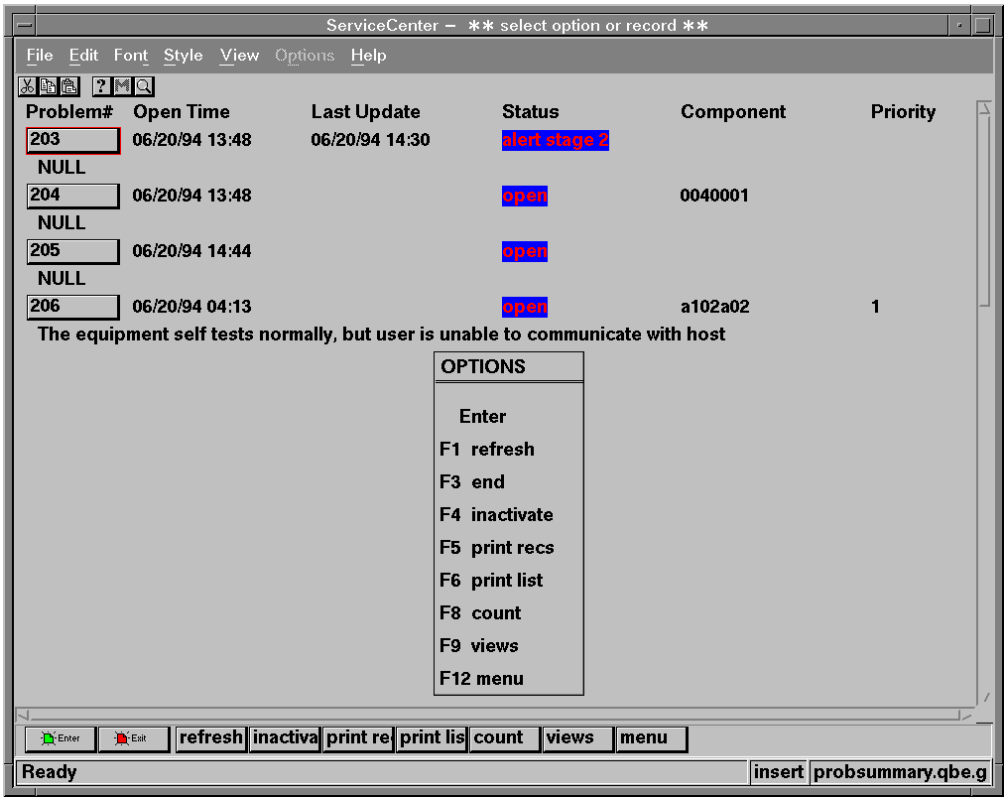

Figure 4-4. Problem List Screen

Open A Problem(_SC_PM_Open)

The ***Open A Problem*** menu option allows you to open a problem in ServiceCenter for the selected object. When the option is selected, the **problem open** ServiceCenter screen is displayed (Figure 4-6). The popup menu allows you to open a problem, plus other standard ServiceCenter functions associated with opening a problem.

To exit the screen, select **F3 end**. This takes you to the logout screen. Select **F1 exit** to end this ServiceCenter session.

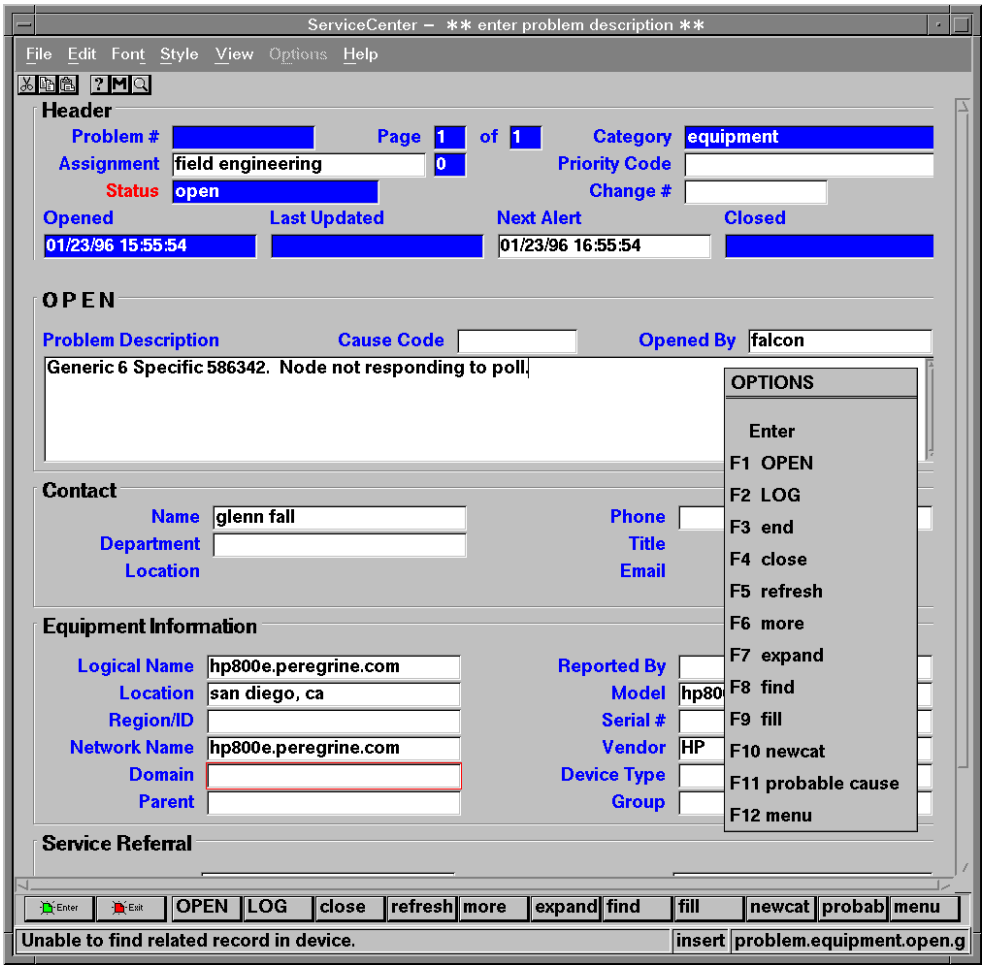

Figure 4-5. Open a Problem Screen

Update A Problem(_SC_PM_Update)

The ***Update A Problem*** menu option allows you to update an open problem in ServiceCenter for the selected object. When the menu option is selected, a list of ServiceCenter problems for the device (Figure 4-6) is displayed if multiple problems are open for the device. Double click on the desired problem in the list. The **problem.update** ServiceCenter screen is displayed (Figure 4-6). If only one problem has been opened for the selected device, the problem list screen is skipped and the problem update screen is displayed after the menu option is selected. If no problems are open for the selected device, a screen is displayed with a message stating that no problems are open for the device. The problem update screen contains a popup menu that allows you to update a problem, plus other standard ServiceCenter functions associated with problem management.

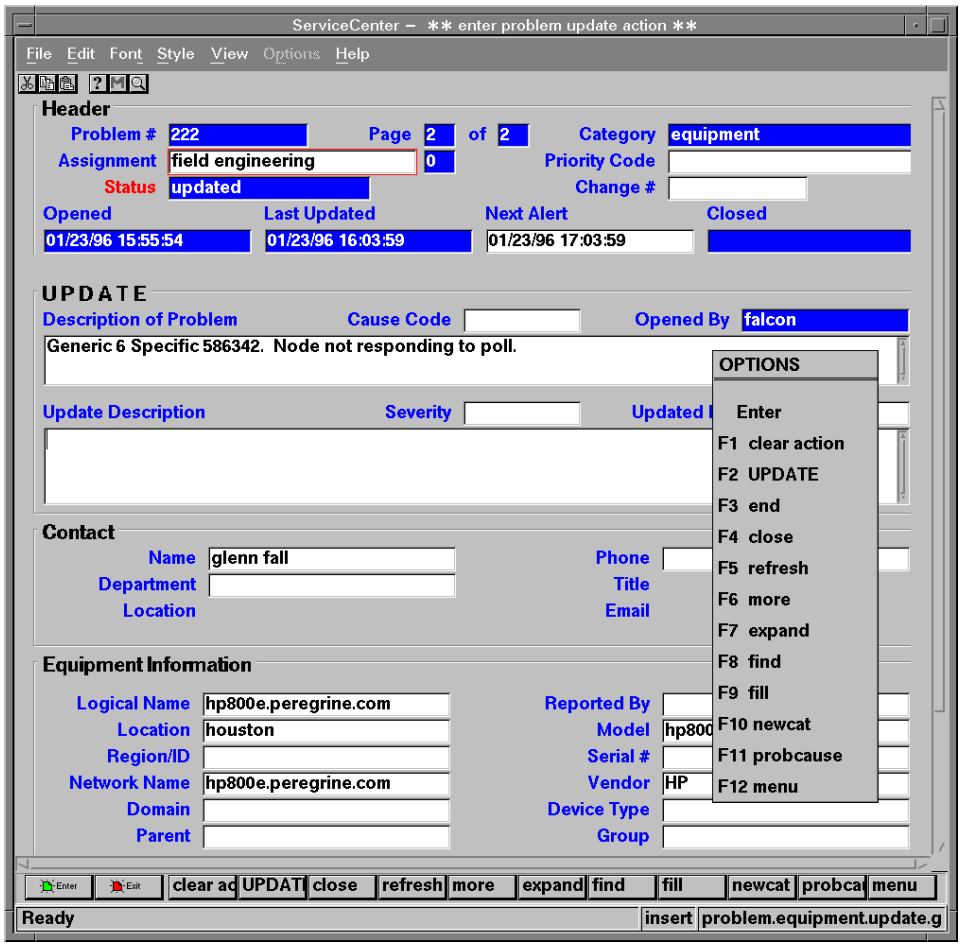

Figure 4-6. Update a Problem Screen

Close A Problem(_SC_PM_Close)

The ***Close A Problem*** menu option allows you to close an open problem in ServiceCenter for the selected object. When the option is selected, **problem close** ServiceCenter screen is displayed (Figure 4-6). The popup menu allows you to close an open problem, plus other standard ServiceCenter functions associated with the problem.

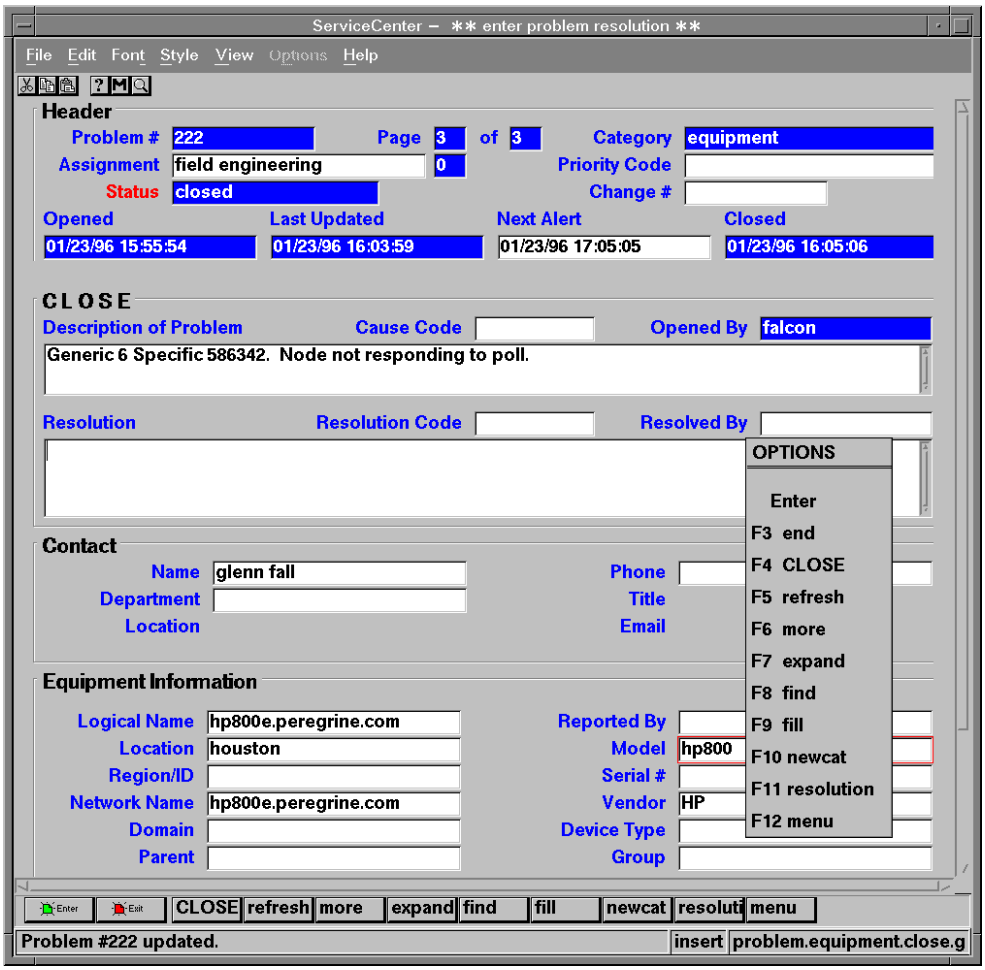

Figure 4-7. Close a Problem Screen

Probable Cause(_SC_PM_Cause)

The ***Probable Cause*** menu option allows you to query ServiceCenter for the probable cause of a problem. When first accessed, a blank *probable cause* screen appears. If you press **Enter**, a **probable cause** list appears. You can select one of the listed probable causes by double-clicking on it.

You can also query ServiceCenter by entering syntax in a field, such as **Key Words**, **Description**, or any of the other fields.

Once a query or selection is completed, a complete *probable cause* screen is displayed (Figure 4-8). The **Resolution** field lists any solution that has been determined for the problem.

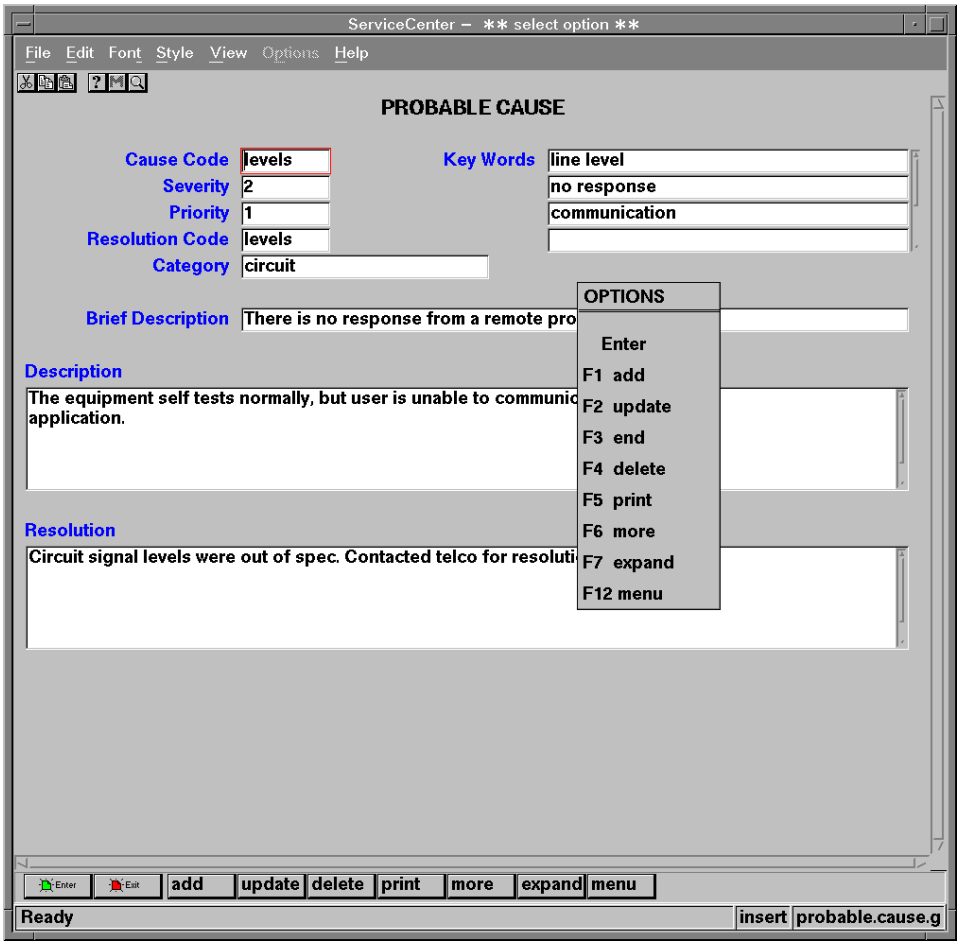

Figure 4-8. Probable Cause

Device Inventory(_SC_Inv_Info)

The ***Device Inventory*** menu option takes you to the ServiceCenter **device inventory** screen (Figure 4-9). This screen allows you to add update, and delete an object, plus find its parents and children.

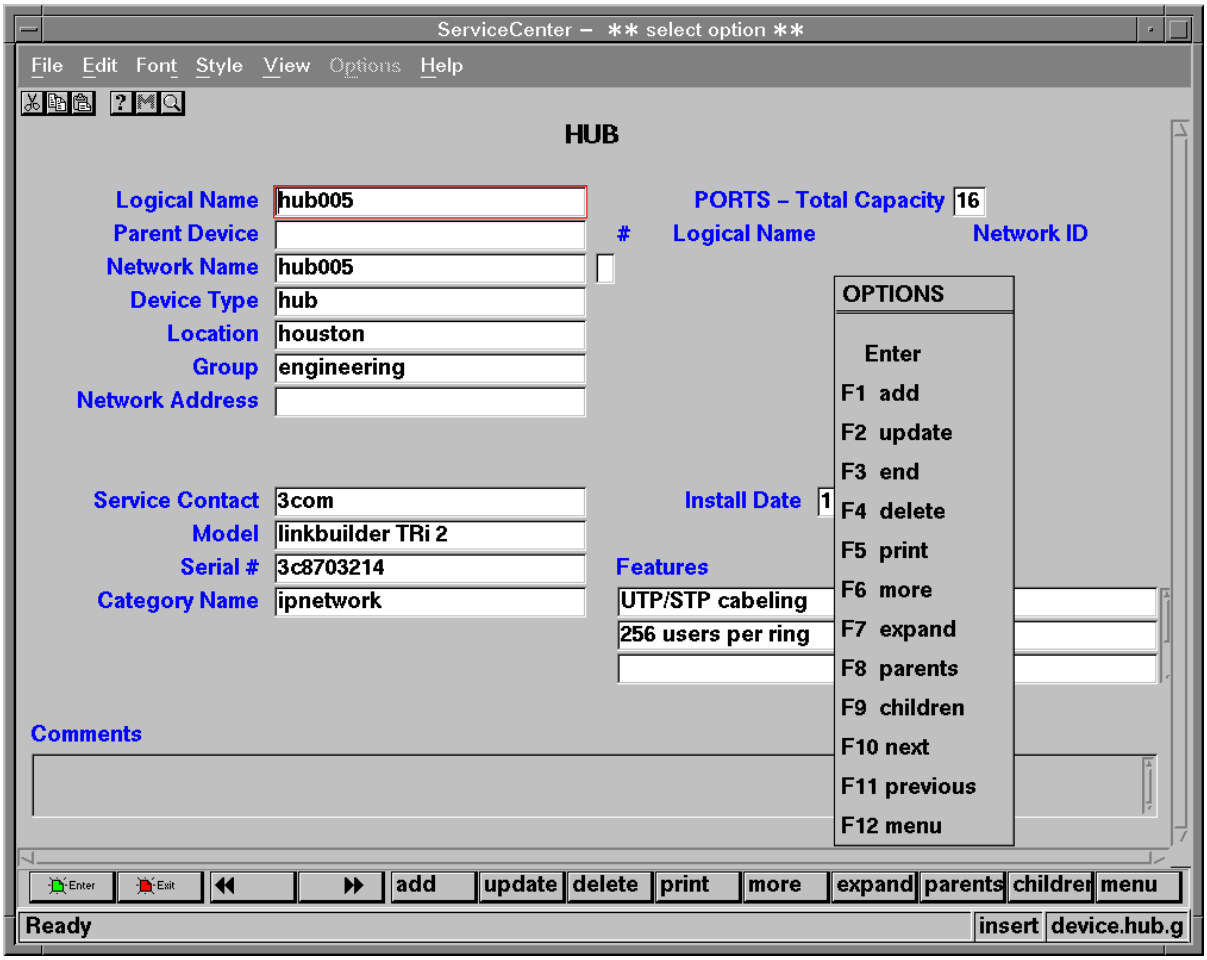

Figure 4-9. Device Inventory Screen

Service Information(_SC_Inventory)

The ***Service Information*** menu option takes you to the ServiceCenter **inventory** screen (Figure 4-10). The popup menu allows you to search for and edit specific ServiceCenter information about the selected object, such as location, vendor, model, problems or availability.

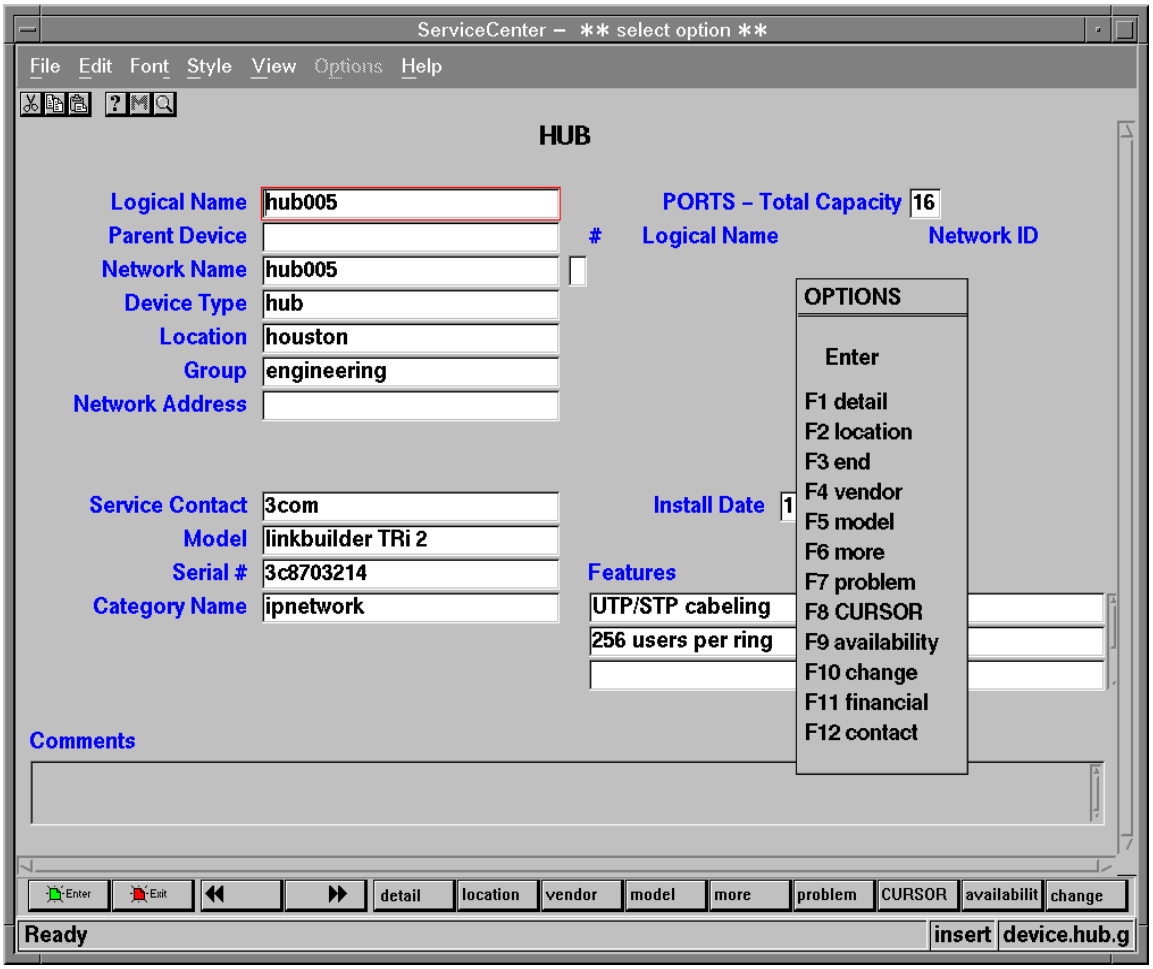

Figure 4-10. Service Information

Down Time(_SC_PM_Outage)

The ***Down Time*** menu option provides the availability time for the selected object. When you first access Down Time, a blank **availability** screen appears. Enter the object's logical name in the **Logical Name** field. The **Logical Name** is the same as the object's label on the object on the submap, or the network name assigned to the object. A list of availability times for the named object is displayed.

If you do not know the **Logical Name** of the object, press **Enter** to get a general *availability qbe* list.

Double-click on the desired entry in the list to get the availability information.

Once an availability record is selected, an availability screen appears (Figure 4-11).

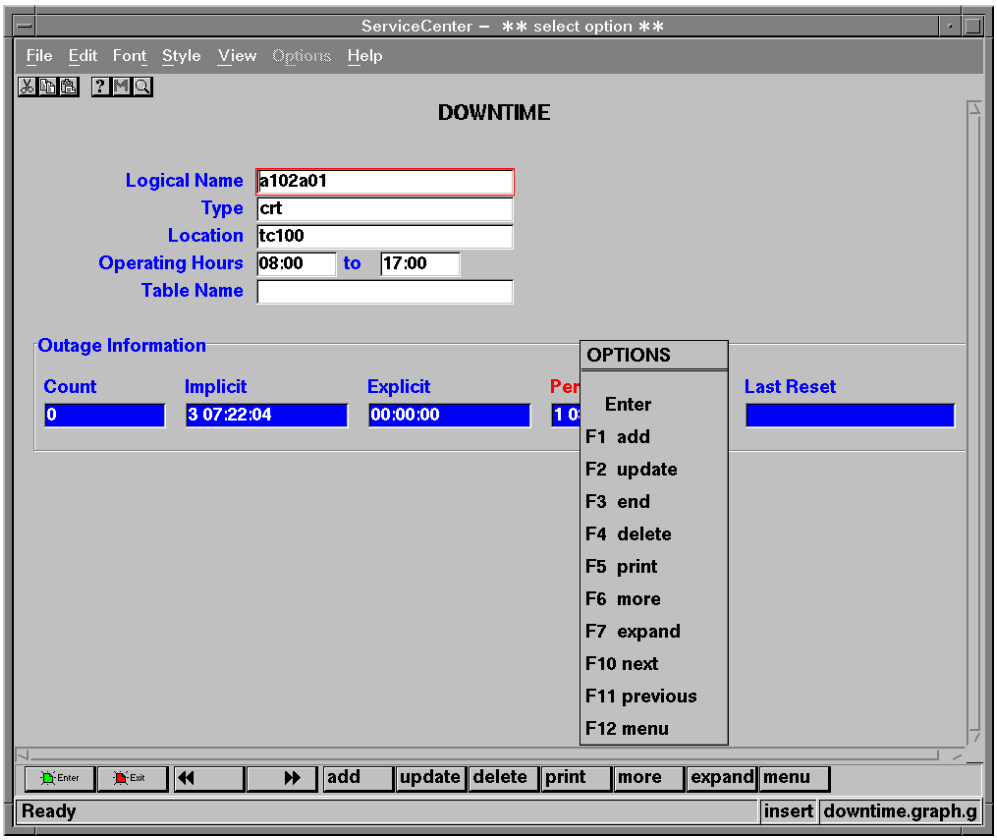

Figure 4-11. Availability Screen

Assigned Problems(_SC_PM_Assign)

The ***Assigned Problems*** menu option provides a summary of the problems assigned to the operator using the current SunNet Manager session. This summary is displayed in the Review Open Problems screen (Figure 4-12). Problems can be opened or updated utilizing the popup menu.

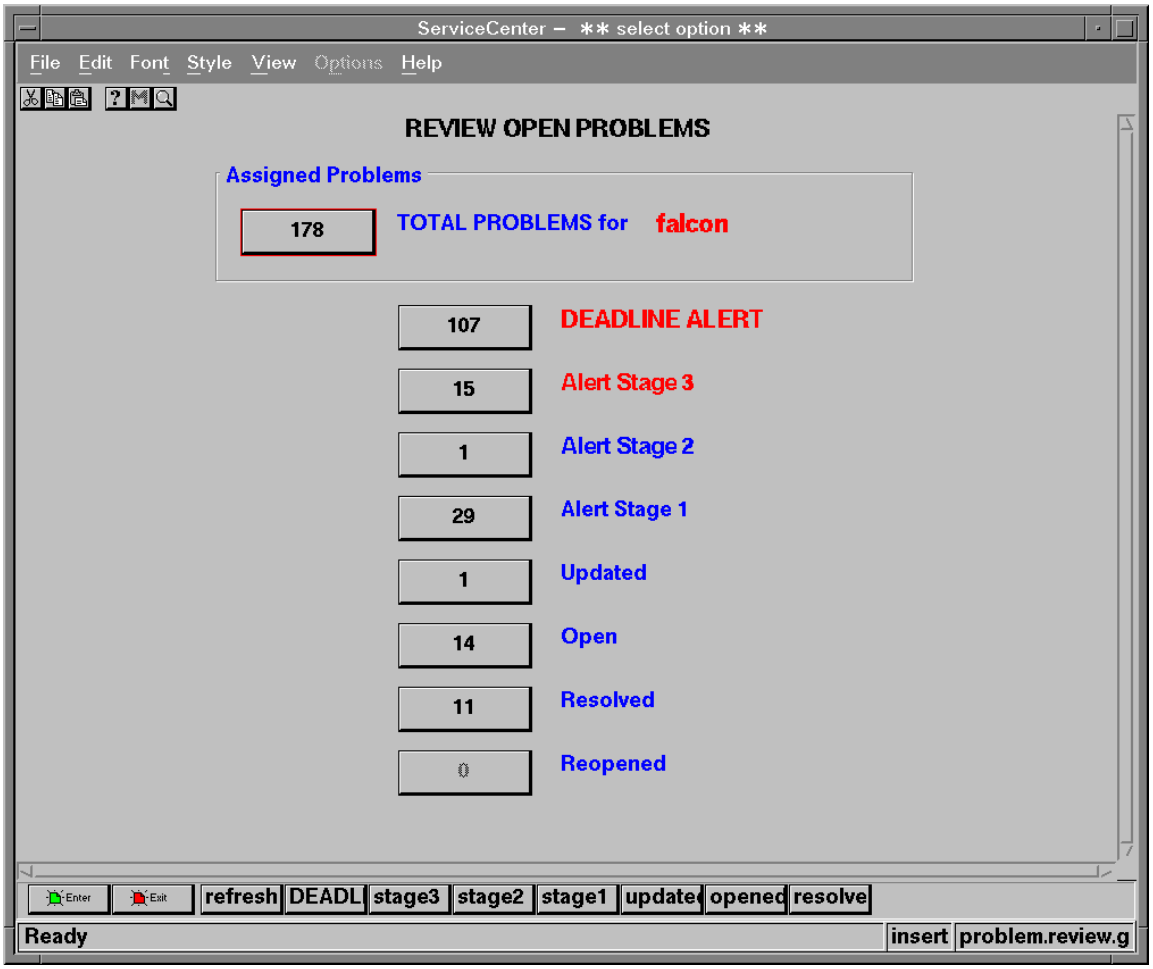

Figure 4-12. Assigned Problems Screen

Start and Stop Daemons(_SC_AutoStart,_SC_AutoStop)

The SCAuto-SunNet implementation requires two daemons running to connect and provide inventory and problem information to ServiceCenter.These daemons may be started using the selected menu items. Please check the logfile or your console X_window for any error messages. The daemons are snIPASd and snIPAStd which were described earlier. If any problem arises using the menu functions, the daemons may be started as a background or foreground UNIX process.

Location Information(_SC_Inv_Loc)

The ***Location Information*** menu option provides location records, much like an address book. When the **location** screen (Figure 4-13) is first accessed, the screen is blank. To find location data, enter the **location name** and select ***find*** from the popup menu.

To view a list of locations, press **Enter** while in the blank screen. A summary list is displayed. Double-click on the desired location to see the data for that location. The location screen will also allow to search, edit, add to and update location information.

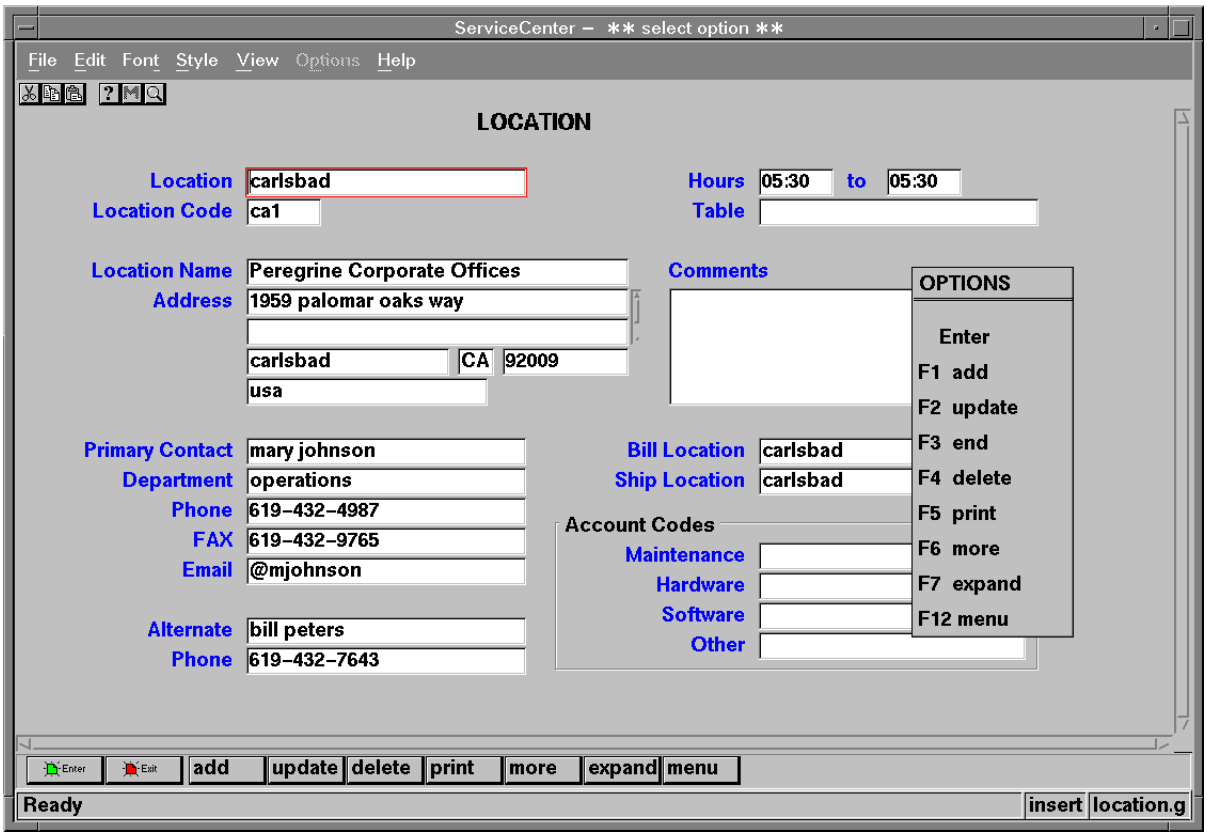

Figure 4-13. Locations Screen

Vendor Information(_SC_Inv_Vend)

The ***Vendor Information*** menu option takes you to the ServiceCenter *vendors* table (Figure 4-14). When Vendor Information is first accessed, a blank **vendor** screen is displayed. Press **Enter** to display a vendor summary list. Double-click on the desired vendor to access the data for that vendor. The vendor screen is displayed with the ServiceCenter information for that vendor. These fields can be edited and updated.

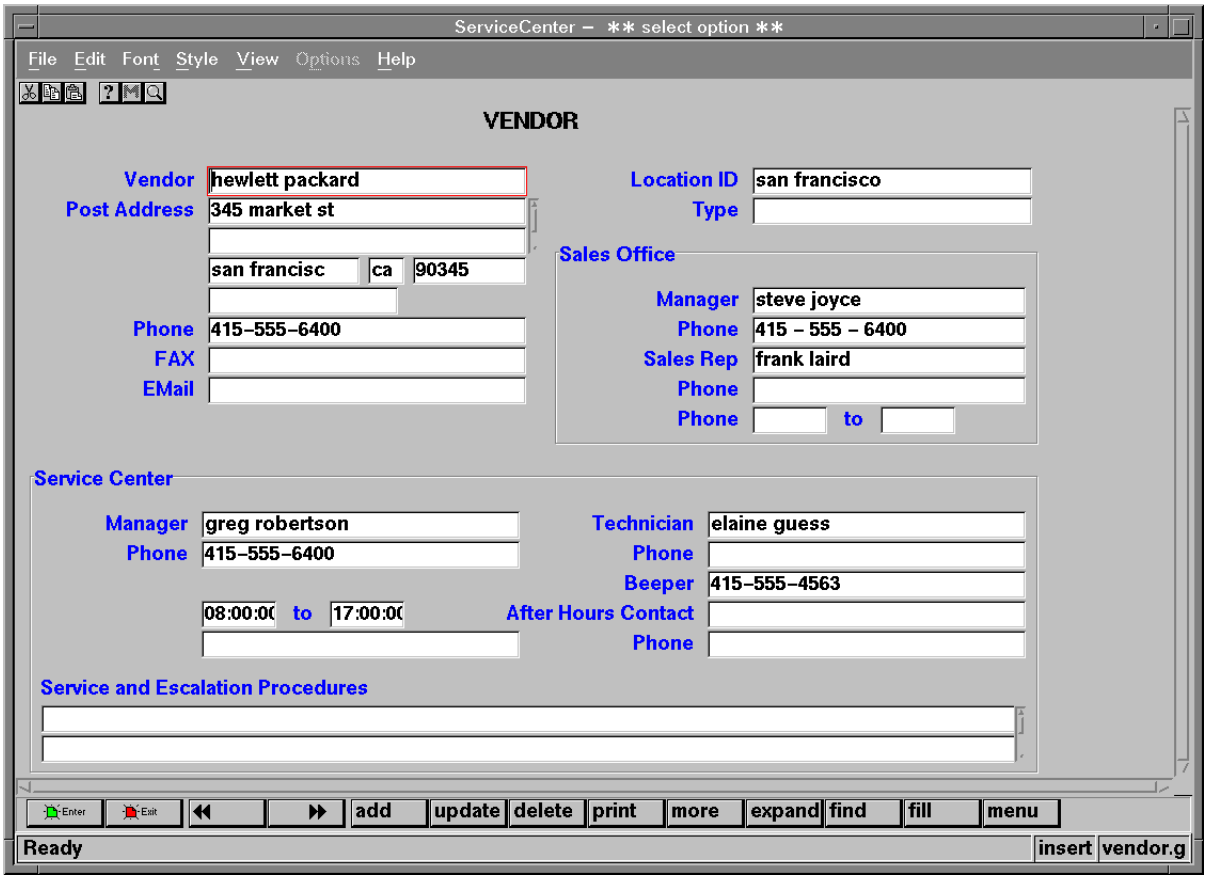

Figure 4-14. Vendors Screen

User Directory(_SC_Inv_User)

The ***User Directory*** menu option allows you to add or query for a particular user. When ***User Directory*** is first accessed, a blank *user directory* ServiceCenter screen appears (Figure 4-15). To query for user information, enter known data in the appropriate field and select **query** from the popup menu.

To get a user list, press **Enter** after the blank screen appears. A user list is displayed. Double-click on the desired user to get the User Directory information for that user.

You can also use the ***User Directory*** option to add or update user information in the ServiceCenter User Directory.

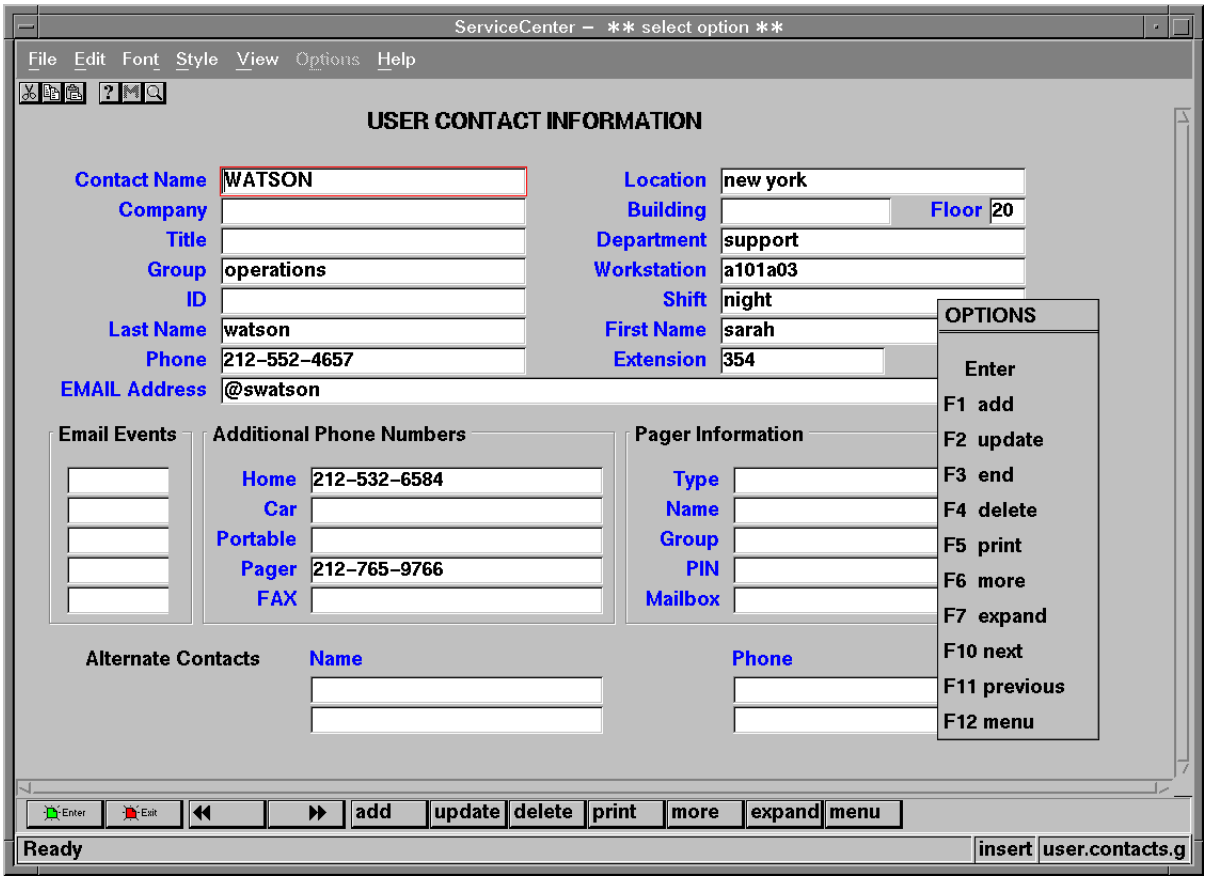

Figure 4-15. User Directory Screen

Filtering(_SC_Filters)

The ***Filtering*** menu option takes you to a **filter setup** screen (Figure 4-16). In this screen, you can set ServiceCenter and SCAuto event filters, or query for an existing filter. All fields in the setup screen are optional, therefore you can either set one field, or all fields, or a combination of fields. This provides flexibility in creating filters. Multiple filters can be set to seek problems under different conditions. Refer to *Chapter 5, Event Management* for more information on the Event Manager application.

The filter setup screen contains the following fields.

Event Type

Allows you to specify an existing or custom event code to define the filter. ServiceCenter contains eight standard events.

User Name

Allows you to specify the user name as defined in the **EV User** field in the event record. A blank user name will match any user name.

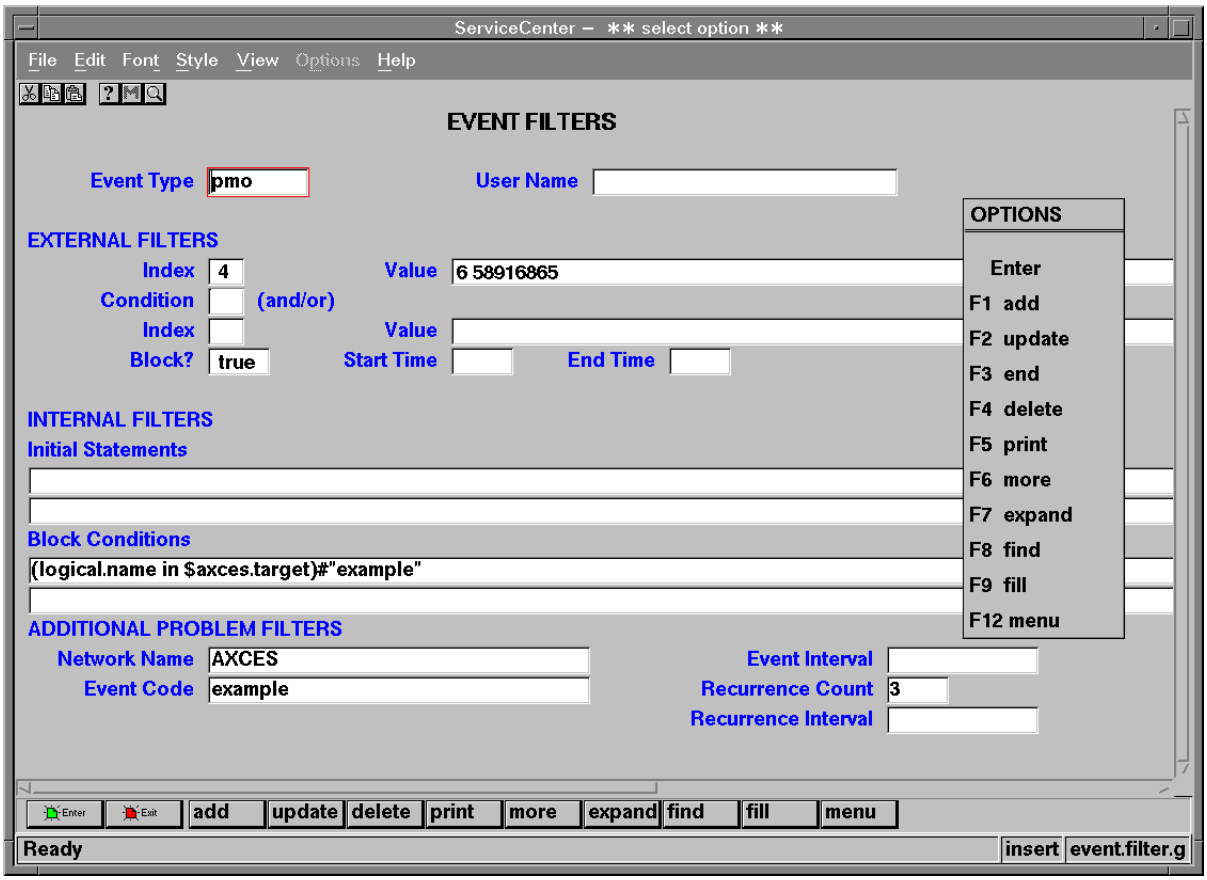

Figure 4-16. Filter Setup Screen

FIELD LEVEL FILTERS

Allows you to specify the EV fields in the Event Record structure. These fields provide specific separator characters. Up to eight EV fields can be used to define a filter.

Index

Allows you to specify the field in the EV Events Record that is to be read by the filter. In Figure 4-16, **3** represents the third field in the record. One or two indexes can be defined for a filter.

Value

Allows you to specify the value the filter is to compare to in the field specified by the index. In Figure 4-16, the value **2** represents the generic SNMP trap code the filter is to look for. The value **58916865** is the specific SNMP trap code the filter is to look for.

Condition

Allows you to specify a relational operator, **and** or, if a second index and value are to be used in the filter.

Block

Enter **true** to completely block the event. Enter false to allow the **Recurrence Count** to take effect.

Start Time

Enter a time for the filter to begin monitoring or block the alert specified by the filter. The format for the **Start Time** field is **hh:mm**.

End Time

Allows you to enter a time for the filter to stop monitoring for the alert specified by the filter. The format for the **End Time** field is **hh:mm**.

Note: If the times are not specified, then the filter continuously remains in effect.

Note: For best performance, only the above fields should be used. The fields below will cause additional event processing overhead when used.

Network Name

Specify the part of the system you want the filter to be applied. For a system-wide filter, enter **AXCES** in this field. For a specific host enter the host name.

Event Code

Enter an SNMP trap code for the filter to search for. If you are not familiar with SNMP trap codes, refer to SNMP documentation for information. The values available for this field are:
0 0, **1 0**, **2 0**, **3 0**, **4 0**, **5 0**, or **6** ***specific***
where *specific* is a vendor specific code.

Event Interval

Allows you to specify a time period an event is active before a problem is opened if the filter condition occurs. The format for the **Event Interval** field is **hh:mm:ss**.

Recurrence Interval

Allows you to specify a time period to open a problem if the filter condition occurs. The format for the **Recurrence Interval** field is **hh:mm:ss**. This parameter is used in conjunction with the **Recurrence Count**.

Recurrence Count

Allows you to set the number of alerts the must occur for the filter before a problem is opened. If the **Recurrence Count** is used in conjunction with the **Recurrence Interval**, a problem is opened if the count value is reached in the set interval.

If **Recurrence Count** is used without a time period set in the **Recurrence Interval**, then the count continues over an indefinite period, while the filter is active.

The count is reset to zero (0) if a problem is opened by the filter.

When the filter is configured, select **add** from the setup screen popup menu. Filters can be added, updated and deleted.

Help Desk(_HelpDesk)

The ***Help Desk*** menu option takes you to the initial ServiceCenter **Help Desk** screen (Figure 4-17). This is the same function as selecting the **Help Desk** option in the ServiceCenter Main Menu. When you first access this option, a blank *problem.initial* screen appears. To query for a problem, you can either enter data in a field or simply press **Enter** to access a problem summary list

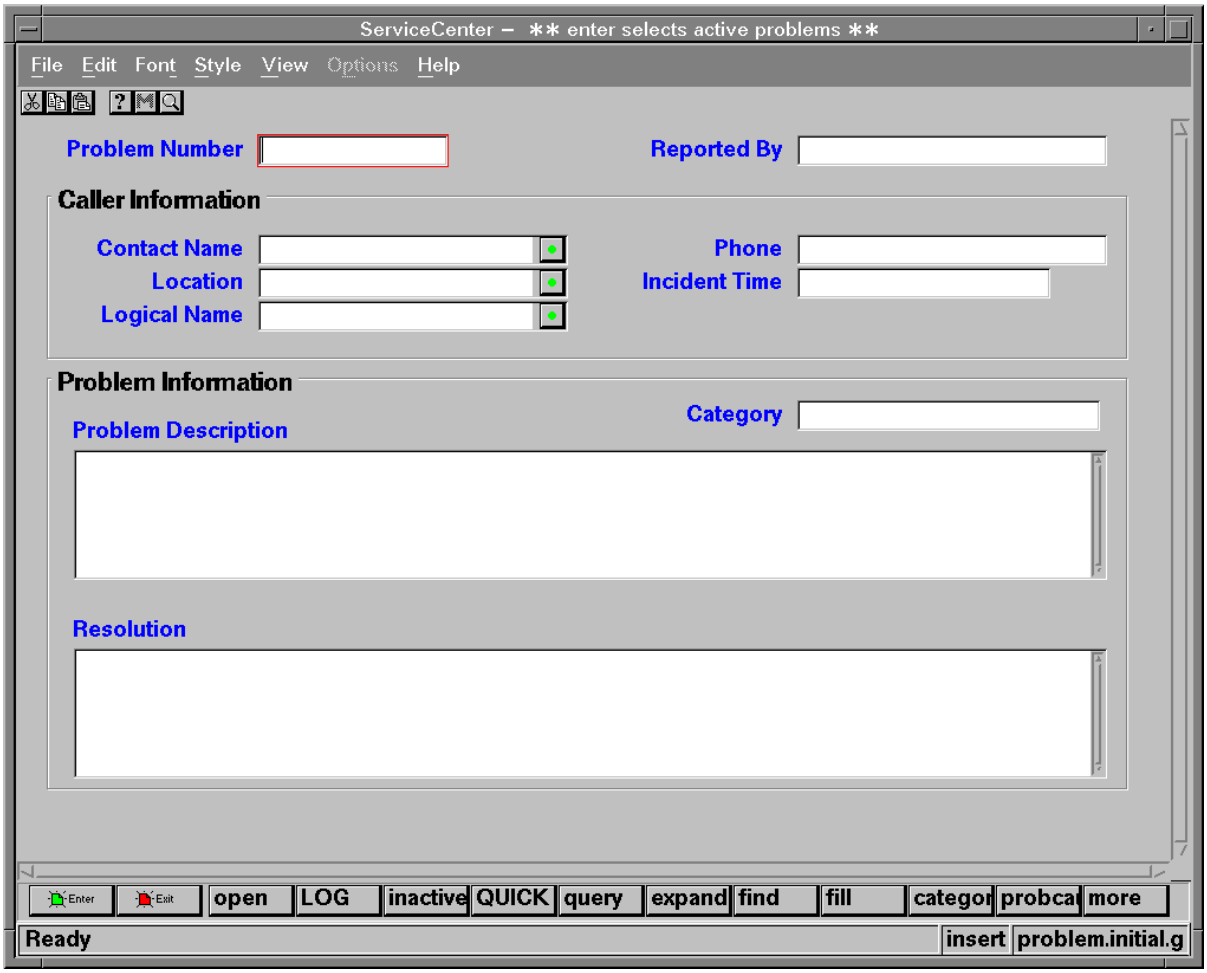

Figure 4-17. Help Desk Screen

Main Menu(_SC_MainMenu)

The ***Main Menu*** option takes you to the ServiceCenter **Help Desk Main Menu** (Figure 4-18). From here, you can access all ServiceCenter functions by utilizing menu buttons, the popup menu, standard ServiceCenter F Keys, or the command line.

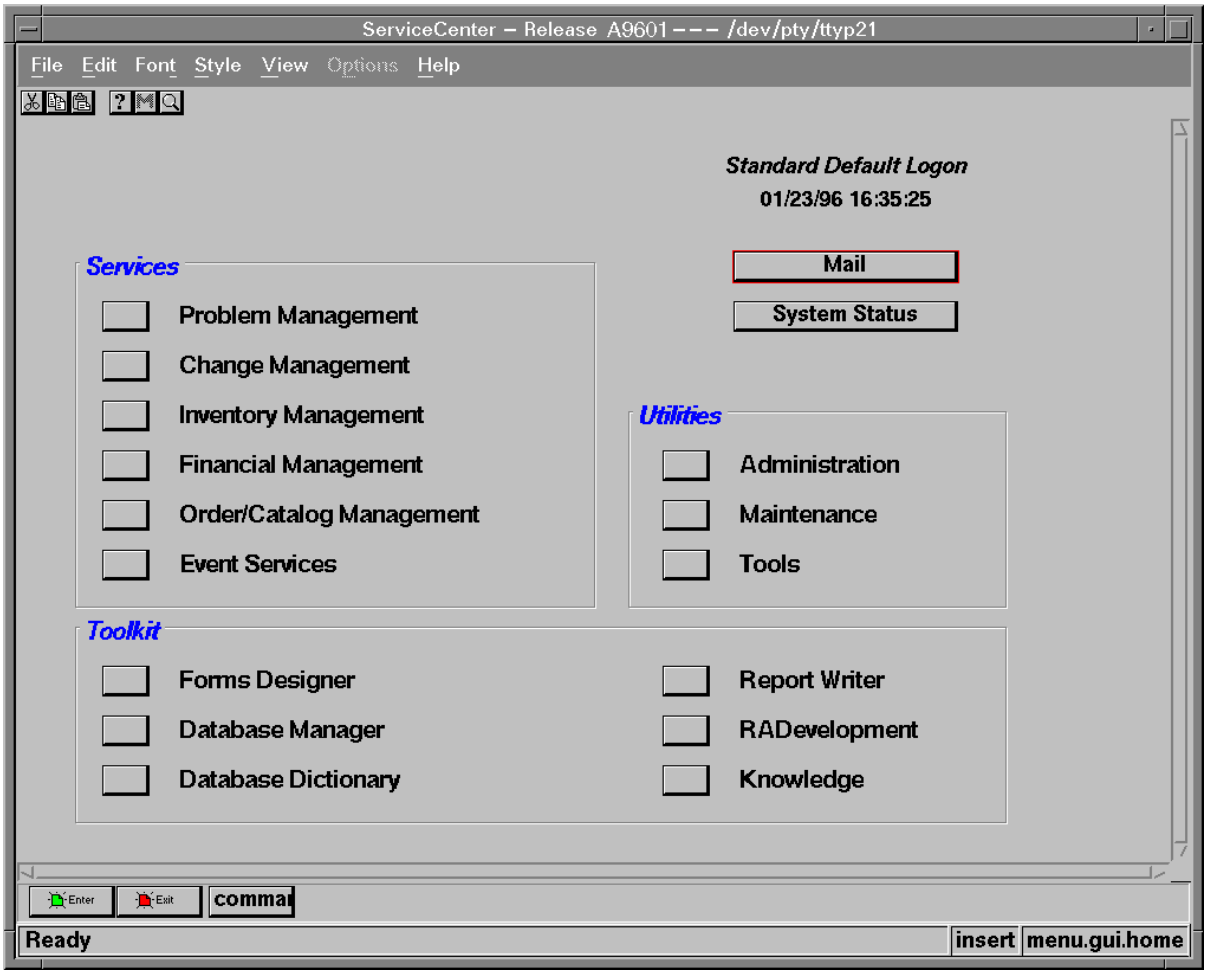

Figure 4-18. ServiceCenter Main Menu

Chapter 5 Event Management

Overview

This chapter provides a short overview of *ServiceCenter Event Services*. For more information on Event Services, refer to ***ServiceCenter Base Utilities Chapter 19***.

Event Services is designed to provide a simple interface between ServiceCenter applications and external programs. The external program must add information to an event services file; event services applications process the information based upon user-defined rules. Although the primary applications include email, problem management and inventory management, additional applications can be written by either the client or by Peregrine Systems' Professional Services.

Event Services is a requirement to install SCAuto.

Major Components

Files

eventin

Contains information added by the external program (or by an RAD routine that reads an external file).

eventregister

Contains instructions for what to do with a record in the *eventin* file.

eventmap

Contains instructions for mapping data passed from the external program to a ServiceCenter record.

eventfilter

Can be used to eliminate ServiceCenter processing based upon several variables such as information content, elapsed time or incident count.

eventout

Contains information that will be read by an external program.

Applications

axces.read

Defined in the event ***schedule*** record's **application** field. This application reads the *eventin* file, looking for records with an Event Time (*evtime*) less than the current time. If a record is found, axces.read checks the *eventregister* file for a valid input event for the Event Type (*evtype*) in the *eventin* record. If an eventregister record exists, axces.read calls the translate module, then the application designated in the eventregister record. After processing has completed in the called application, axces.read cleans up the *eventin* file according to directions in the eventregister record.

axces.translate.input

Called by **axces.read**. The application converts the delimiter-separated information in the *eventin* record's **evfield** field to an array (*evlist*), and converts the data to all upper or lower case depending upon instructions in the *eventregister* record.

axces.appl.error

The default application called when SCAuto detects an error of any kind. The application sends a message using

the *application error* type.

axces.problem

Prepares for mapping data from the *eventin* record to a data record. This application calls **axces.map.fields** and **axces.problem.setup**, and is called by the event register.

axces.map.fields

Maps the data from *evlist* in the ***eventin*** record to fields in a problem record, converting data types and evaluating all expressions in the map records.

axces.problem.setup

Determines whether or not a problem exists to be updated or closed. The application also inserts a line in the action, update.action or resolution of the problem with a time stamp. This application calls **axces.problem.filter**.

axces.problem.filter

Filters events based on instructions in the *eventfilter* file.

axces.problem.cover

Writes the logical name and network name to *eventout* for use by Peregrine Systems's COVER product.

axces.problem.email

Sends external email, rather than ServiceCenter mail, for problem messages.

axces.inventory

Adds to and updates the ServiceCenter device file. Data is mapped in the same manner as for problem open, update and close. Records to be deleted are marked for delete in the **estatus** field; format control can call a subroutine to delete the record based on the value in the estatus field. The **last.update** field is always updated when SCAuto touches a device record.

axces.inventory.delete

Removes records from the device and, where appropriate, attribute files. The application can be called by format control.

axces.email

Converts ServiceCenter mail to electronic mail output format and adds the information to the *eventout* file with an evtype of *email*.

axces.email.receive

Converts email information found in the *eventin* file to ServiceCenter mail.

axces.build.maps

Converts a dbdict record to *eventmap* records. Fields can then be removed from the mapping and field numbers

resequenced.

axces.query.setup

Used to cut-through from COVER and OpenView. This application resolves a query of either *logical.name* or *network.name*, since either can be passed arbitrarily.

axces.write

Converts a ServiceCenter record to SCAuto format using either a format or a mapping name passed in the *eventregister* file. The data is written to the *eventout* file according to instructions in the registration record.

elogin

Used to cut-through from external applications to ServiceCenter. The cut-through executes the application passed to it, using any additional parameters.

The application expects parameters in the following format:

scenter elogin:*user name*:*application name*::*parm name*::*parm value*!

For example, to open a problem on device *a101a01*, use the command:

scenter elogin
:falcon:pm.access::name::*a101a01*::text::opn!

To review problems assigned to falcon:

scenter elogin :falcon:pm.access::text::review!

To select problems with logical names beginning with *a*:

scenter elogin
:falcon:pm.access::query::logical.name#^a^::text::qu!

To access the problem profile database:

scenter elogin :falcon:database::name::pm.profile!

To access the **PROBLEM MANAGEMENT** menu:

scenter elogin
:falcon:menu.manager::name::PROBLEM_MANAGEMENT
!

To access the second screen of the device whose network name is *a101a01*:

scenter elogin :falcon:icm.access::names::*a101a01*::text::2!

Replace **“** with ^ in any query, and replace a space with **_**. Only one parameter may be passed to **elogin**, and it must be one contiguous string beginning with a colon and

ending with an exclamation point. The delimiters in this parameter are double colons EXCEPT for the single colon immediately following the login name. All parameters are passed as strings. Upon exit from the called application, **elogin** prompts to exit

external.query

Allows a standard query to be passed from an external application. The name of the format is passed with the *name* parameter, and the query itself with the *query* parameter. For example, to select all locations in San Diego, the command is:

scenter elogin :falcon:external.query::name::location::query::city=^San_Diego^!

Miscellaneous

- The *event* scheduler polls the *eventin* file at regular intervals, looking for records whose Event Time is less than the current time. The event scheduler is started in the same manner as other ServiceCenter schedulers (e.g., report).
- The *number* file has a new class called *event*.
- The *number* file has many new fields, including a length parameter.
- The **getnumb** application now can convert a number to a string.
- The *msgclass* file has a new message class called *event management errors*.
- The *msgclass* file has a new message type called *email*.

Index

No index is availible for this document at this time