ServiceCenter®

Version 3

SCAuto for Lotus Notes

February, 2001

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

Peregrine SYSTEMS®
The Infrastructure Management Company™

This edition applies to version 3 of the licensed program of **ServiceCenter**®
and version 2.0 of **SCAuto for Lotus Notes**.

Contents

Chapter 1 Introduction

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

This guide describes how to implement the SCAutomate interface for integration with Peregrine Systems' ServiceCenter.

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

Prerequisite Knowledge

This guide assumes you have:

- Working knowledge of ServiceCenter applications, ServiceCenter Client/Server, and Lotus Notes. While some procedures for these applications are explained, others are referenced. Refer to the appropriate ServiceCenter documentation for a more detailed explanation.
- Understanding of ServiceCenter Event Services and ServiceCenter's Change Management and Problem Management modules.
- Working knowledge of a GUI or text-based environment.
- Administrators should have a thorough knowledge of the operating system where the product will be installed and implemented, as well as a basic understanding of ServiceCenter applications and Event Services.

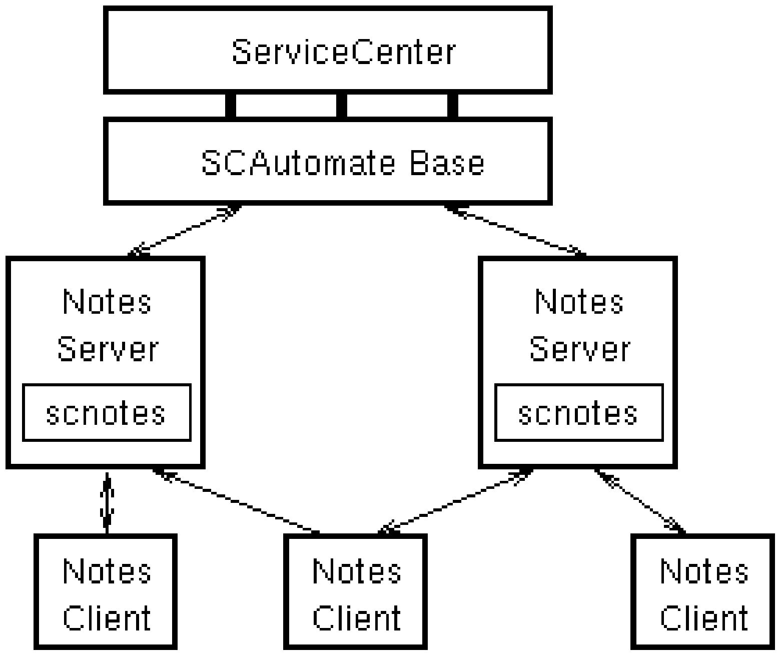
Important: ServiceCenter installation requirements are specific to the machine where ServiceCenter is being installed. These requirements are listed in the respective installation guides.

What is SCAuto for Lotus Notes

SCAuto for Lotus Notes provides a Lotus Notes (Notes) interface to ServiceCenter's Problem Management and Change Management applications. ServiceCenter problem and change tickets can be created, updated and closed by both Lotus Notes and ServiceCenter users. Bi-directional communication keeps tickets synchronized in both systems.

The interface consists of a server add-in task for Notes plus two Notes databases: the configuration database and the sample help desk change/problem management database. The supplied sample database can be customized using standard Lotus Notes application design facilities. The server add-in task communicates with ServiceCenter's Event Services, incorporating new change and problem events into Notes and creating events from notes that have been added or changed. Communication with ServiceCenter is handled by the SCAutomate product.

The following diagram illustrates the flow of data between ServiceCenter, SCAutomate, the Lotus Notes servers, and the Lotus Notes clients.



Contacting Peregrine Systems

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

North America, South America, Asia/Pacific

Telephone:	+(1) (800) 960-9998 (within US only, toll free)
	+(1) (858) 794-7402
Fax:	+(1) (858) 794-6028
Email:	support@peregrine.com
Headquarters:	Peregrine Systems, Inc.
	Attn: Customer Support
	3611 Valley Centre Drive
	San Diego, CA 92130

Europe, Africa

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

Documentation Web Site

For a complete listing of Peregrine documentation, see the Documentation pages on our Customer Support web site at:

http://support.peregrine.com

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

You can download .pdf files using Adobe Acrobat Reader (also available on the web site) or order printed copies of the documentation through your Peregrine Systems sales representative.

Chapter 2 Installation

Preparing for Installation

To communicate between Lotus Notes and ServiceCenter, the SCAutomate product from Peregrine Systems is used. This requires a TCP/IP network connection. The following components must be available and installed for SCAuto for Lotus Notes to function properly:

- ServiceCenter version 1.4 or higher, running the A9602 or later file system release.
- SCAutomate Base, with a version matching ServiceCenter.
- Lotus Notes Server *(*version 3 or higher*)*.
- TCP/IP; for the Windows version, the TCP/IP stack must be Winsock compatible.

SCAuto for Lotus Notes runs on the following systems:

- 32-bit Windows (95, 98, and NT).
- OS/2 Warp (3.0 and up). Lotus Notes version must be 4.0 or higher for OS/2.

Note: It is highly recommended that the ServiceCenter and SCAutomate Base installations be tested before attempting to install SCAuto for Lotus Notes.

Installing the Server Add-in

The Notes server add-in is the backbone of SCAuto for Lotus Notes. The add-in is started by the Notes server, either automatically or manually. Some versions of Notes require that server add-ins be in the same directory as the Notes server. The server add-in should be copied into that directory (along with the SCAuto DLL on systems that require it).

The name of the server add-in varies depending upon your platform type:

Platform	Name
32-bit Windows	`NSCNOTES.EXE`
OS/2	`ISCNOTES.EXE`

The location of the Notes server directory depends upon how Notes is installed. If you do not know the location, search in the Notes directory for the Notes server executable:

Platform	Name
32-bit Windows	`NSERVER.EXE`
OS/2	`ISERVER.EXE`

The Notes server directory most likely is the top-level directory in the Notes hierarchy.

SCAuto for Lotus Notes requires the *SCAuto.DLL* file. This should be copied into the Notes server directory along with the server add-in.

Windows example for Notes 4.x:

```
>COPY D:\WINNT\NSCNOTES4.EXE C:\NOTES\NSCNOTES.EXE
>COPY D:\WINNT\SCAUTO.DLL C:\NOTES
```

Windows example for Notes 5.x:

```
>COPY D:\WINNT\NSCNOTES5.EXE C:\NOTES\NSCNOTES.EXE
>COPY D:\WINNT\SCAUTO.DLL C:\NOTES
```

OS/2 example:

```
>COPY D:\OS2\ISCNOTES.EXE C:\NOTES
>COPY D:\OS2\SCAUTO.DLL C:\NOTES
```

Installing the Sample Databases

Two of the files distributed are Notes databases: the sample help desk Problem/Change Management database and the Configuration database. These databases can be used as is, but Notes or ServiceCenter administrators will more likely want to customize them for their users. These files must be installed on the same Notes server that will run the server add-in.

The sample databases are called *scprob.nsf* and *scconfig.nsf*. By default, SCAuto for Lotus Notes looks for these databases in the Notes data directory. These databases need to be copied to the Notes data directory. The Notes server cannot serve these files to Notes clients if it cannot find them. The Notes data directory should be in the system search path.

This user's guide is also a Notes database. It is in the *scman.nsf* file, and it can be copied to the Notes data directory.

Windows and OS/2 example:

```
-COPY SCPROB.NSF C:\NOTES\DATA\

-COPY $SCMAN.NSF C:\NOTES\DATA\
```

Replace the `C:\NOTES\DATA` path with the correct path to your server's data directory. This is dependent upon how Notes was installed; it is usually `\NOTES\DATA` or `\NOTES`.

After copying the databases, you need to create icons for them within Notes.

1. Choose **File>Database Open** from the **Lotus Notes** menu.
2. Select the name of the server you will use.
3. Select the database titled *SCAutomate Help Desk*. This action places an icon for the main SCAuto for Lotus Notes sample database onto the current Notes workspace. This should be done for each user's Notes client that needs to access SCAuto for Lotus Notes.
4. Perform steps 1 through 3 again, this time for the Configuration database, which is titled *SCAutomate Configuration*.

Warning! If more than one Lotus Notes server will be communicating with ServiceCenter, and those Notes servers replicate databases with each other, be sure the SCAuto for Lotus Notes databases are given different names (in this example, do not keep the default name *scprob.nsf*). Keeping the same name causes conflicts with SCAuto for Lotus Notes configuration.

Setting Security

Lotus Notes contains several built-in security facilities. Since SCAuto for Lotus Notes is accessed from a Lotus Notes server, security is a concern. Security must be set up correctly if you want to configure SCAuto for Lotus Notes or to have users respond to or create their own notes. It is beyond the scope of this guide to discuss Lotus Notes security fully.

Two important security concerns must be addressed during installation.

- At least one user must have manager access to the SCAuto for Lotus Notes databases.
- At least one user must be added to the *[configure]* role.

Note: Default security for the SCAuto for Lotus Notes databases allows manager access only to servers, not to users.

Permission to change security

To ensure that permission exists to change security on the database:

1. Change your user ID to a server ID if you have not done so (at least temporarily) by choosing **File>Tools>Switch ID**.
2. Select **server.id**. This step varies, depending on the exact setup at your site. If you have problems performing the following steps, consult your local Lotus Notes administrator.

Access to databases

To give a user manager access to both SCAuto for Lotus Notes databases:

1. Select the SCAutomate Help Desk database (click once on its icon).
2. Choose **File>Database>Access Control**.
3. Add at least one user or group name to the list, giving each name manager access.
4. Repeat steps 1 through 3 for the SCAuto for Lotus Notes Configuration database.

   The user or group is allowed full access to the database, including the ability to change security, delete any document, and so forth. This access level should be given out sparingly.

Notes requires that at least one user or group have manager access. The *LocalDomainServers* group should keep manager access if this fits within your existing Notes security scheme.

At this point, if you temporarily switched to a server ID, you can switch back to your user ID if you gave manager access to you or to a group you are in.

[Configure role]

Configuration parameters are modified through a document in the SCAuto for Lotus Notes Configuration database. At least one user must be given permission (having manager access is not enough; permission needs to be granted explicitly). By default, the *LocalDomainServers* group has the necessary permissions, so you can skip this step by switching user IDs when configuring SCAuto for Lotus Notes.

To add a user to the *[configure]* role:

1. Click on at least one user or group from the Access Control dialog box.
2. Click on the *[configure] role* to add this role to the selected user(s) and/or group(s).
3. Repeat steps 1 and 2 for the other SCAuto for Lotus Notes database (SCAutomate Configuration).

Manually Starting SCAuto for Lotus Notes

For initial installation and testing purposes, SCAuto for Lotus Notes should be started manually.

- To manually start SCAuto for Lotus Notes:
  1. Start the Lotus Notes Server.
  2. In the server console window, type the command:

     `LOAD scnotes`

     - or -

     `LOAD <`*`path-to-scnotes`*`>`, if scnotes is not in the Server's path (not recommended).

     Note: SCAuto for Lotus Notes can only be started from the server and cannot run as a standalone.

- To stop SCAuto for Lotus Notes:
  1. Go to the SCAuto for Lotus Notes Server.
  2. Type the following command in the server window:

     `TELL scnotes QUIT`

     Whenever the Notes server is stopped with the `QUIT` command, Notes attempts to stop SCAuto for Lotus Notes.

- To check if SCAuto for Lotus Notes is running, or to see its status:
  1. Go to the SCAuto for Lotus Notes Server.
  2. Type the following command in the server window:

     `SHOW TASKS`

     The actual name of the executable file is not *scnotes*. The file is named:

     `iscnotes.exe` **`for OS/2`**

     `- and -`

     `nscnotes.exe` **`for Windows NT`**

     The full executable name is not typed when loading it from the Lotus Notes server. Instead, use the filename minus its first letter and the extension.

     As shown, the command uses the default configuration database *scconfig.nsf*. To use a different configuration database, supply the database name (and path if needed) at the end of the load command. For example:

     `LOAD scnotes` ***`scconfig2`***

Automatically Starting SCAuto for Lotus Notes

The Lotus Notes Server can be configured to automatically load the SCAuto for Lotus Notes add-in task whenever it starts. To configure the Lotus Notes Server in this way:

Note: Refer to the Lotus Notes documentation for your system as you perform these steps.

1. Make a backup copy of the *NOTES.INI* file. This file is in the *Notes\DATA* directory on OS/2 and the *WINNT* directory on Windows NT.
2. Open the copied file for editing.
3. Find the line that begins with **ServerTasks=**. This line contains a comma delimited list of server tasks to be invoked whenever the Notes server is started.
4. Add **scnotes** to this list. Include the full path if the executable was not copied to the Notes software directory. Include the name of the SCAuto for Lotus Notes configuration database as a parameter (again, using a full path if it was not copied to the Notes data directory). This line should appear similar to the following:

   ServerTasks=Replica,Router,Update,scnotes scconfig
5. Save your changes to *NOTES.INI*.
6. Use the **QUIT** command to stop the Lotus Notes server. If you have Lotus Notes Workstation running on the same machine as the server, stop that also.
7. Restart the server.
8. Verify that SCAuto for Lotus Notes is running. It can be stopped in the same way as described on page 3-6.

Chapter 3 Configuration

Configuring SCAuto for Lotus Notes

After setting up security, permission is enabled to configure SCAuto for Lotus Notes.

1. Ensure that your current user ID is in the `[configure]` role.
2. Access the SCAuto for Lotus Notes Configuration database.
3. Select **Admin>Configure** from the **View** menu.
   - SCAuto for Lotus Notes ships with a default configuration note. You should never have more than a single record in this view.
   - If the configuration note is not there initially, select **SCNotesConfig** from the **Create** menu to create a new configuration note.

The configuration note describes the various configuration parameters. Fill in these parameters with appropriate values, as described below:

Note: Other parameters can exist in the configuration note, either for debugging or finetuning. The configuration note describes all of these parameters.

Database ID

Database file (default is scprob). Specify the name of your Help Desk database. The filename must be 16 characters or less. The entire specification, which includes the filename and possibly a path or node can be up to 127 characters in length.

SCAuto Server

Host and service name (separated by a period) that enables connection to the remote SCAuto Base server. This parameter is required and has no default. If the service name is not defined on your machine, you may specify the SCAuto port number instead.

Debug

If set to a non-zero value (e.g. 1), some basic debugging is enabled. Extra messages are sent to the server console window as well as to the Notes Event Log. If set to 2, then no checkpoint records are read on startup. This means if you stop SCAuto for Lotus Notes and start it again, the same events and notes are processed. This is useful for initial testing.

Create ID

Specifies the username ID for all newly created events. The default is scnotes.

Event Username Filter

List of usernames. Unless this field is blank or the keyword all, SCAuto for Lotus Notes will only retrieve events that were created by these users.

Event Type Filter

List of events to process. SCAuto for Lotus Notes will retrieve from ServiceCenter only the events listed here.

View List

There are two methods for selecting notes to process for creating output events to ServiceCenter. By default, SCAuto for Lotus Notes selects all notes that have been modified since the last checkpoint (last processing time). An alternative method is to specify a list of views that contain the modified notes to process.

It is the responsibility of the user to create and populate these views with the appropriate notes to process for output. This includes preventing processed notes from remaining in the view, otherwise they will be processed repeatedly. Following is an example of a view selection formula showing how the SCAuto Control Field can be used to filter out notes that have already been processed.

SELECT SCprob_domain = "Animals" & (scauto_flag != 1 & scauto_flag != 3)

SCAuto Control Field

SCAuto Control Field establishes the field name that will be used to control the processing of notes. Unless a View List is being used (see above), SCAuto for Lotus Notes will only process modified notes that contain this field and have its value set to 0 (zero).

Event Trigger Table

Defines the conditions for which events will be generated and sent to ServiceCenter. The first two columns define Notes field names and their corresponding values. The table is searched for each note to process. If the field name exists in the note and its value matches the Field Value, then an event is created of the type specified in the third column, Event Type. Every match generates an event, so a single note can generate multiple events.

There are two special cases:

- If the field value is empty, then any value in the note will match.
- If the field name and field value are both empty, then the event will always be generated.

Event Type Field Name	Field Value	Event Type
SCprob_status	open	pmo
SCprob_status	updated	pmu
SCprob_status	closed	pmc
cm3r_action		cm3rin
cm3t_action		cm3tin

Configuring ServiceCenter Event Services

The Event Services facility of ServiceCenter is used as the interface from Notes to ServiceCenter. As shipped, ServiceCenter does not create output events for problems or changes.

To prepare ServiceCenter to talk to SCAuto for Lotus Notes:

- Ensure that *scautod* (the SCAutomate Base Server) is installed and tested. The binary for *scautod* is in ServiceCenter's **RUN** directory.
- Ensure that the ServiceCenter applications are at release A9602 or higher.
- Place an authorization code that enables SCAuto for Lotus Notes capability in the *sc.ini* file. If you do not already have such an authorizaton code, contact your Peregrine account representative.

Events

For Problem Management, incoming *pmo* events should automatically create *pmo output* events. This process is necessary to correctly link up the new problem created by Notes with the problem number assigned by ServiceCenter. The user sequence number is carried through to the output event.

1. Enter the command **eventreg** on a ServiceCenter command line.
2. Select the record that has *Event type* set to *pmo* and *Application Name* of *axces.apm*.
3. Look at the parameter values and ensure that the parameter labeled **write eventout?** is set to *true*. Set it to this value and update the record if it is not already *true*.

Problem Management

The Problem Management application needs to create *eventout* records. This process is described in the ***ServiceCenter Event Services*** guide. For better integration with SCAuto for Lotus Notes, follow the instructions in this guide, but modify the *add* condition slightly.

Note: This step is done separately for each problem category. Also this step needs to be done for open, update, and close. For instance, enabling output events for the *Equipment* category, these changes must be made to the *problem.equipment.open*, *problem.equipment.update* and *problem.equipment.close* forms.

1. Get the *long* form of the subroutines display. This is done with the **Show Expanded Form** option on the **Options** menu.
2. Add the *axces.write* subroutine as shown in the ServiceCenter manual. However, instead of using the expression *true* for the *add* condition, use the expression:

   operator() ~# "scnotes"

   This ensures that the output event does not get created twice, once because of this subroutine, and once because of the event registration record.

Change Management

For Change Management, incoming *cm3rin* and *cm3tin* events should automatically create *cm3rinac* and *cm3tinac output* events, respectively. As with Problem Management, this process correctly links up the new request and task numbers assigned by ServiceCenter as well as the user sequence number.

1. Enter the command **eventreg** on a ServiceCenter command line.
2. Select the record that has *Event type* set to *cm3rin* and *Application Name* of *axces.cm3*.
3. Look at the parameter values and ensure that the parameter labeled ***write eventout?*** is set to *true*. Set it to this value and update the record if it is not already *true*.
4. Do the same for the record that has *Event type* set to *cm3tin* and *Application Name* of *axces.cm3*.

The Change Management application also needs to create `eventout` records for requests.

1. Type the command **db** on a ServiceCenter command line.
2. Type **cm3messages.g** in the *Form* field.
3. Click on the **Search** button to display a list of all of the Change Management event definitions.
4. Open the first record, *cm3r approval*, and locate the *Axces Reg* field.
5. Type **cm3rout** in this field.
6. Click the **Save** button to update this record.

7. Perform the steps 5 and 6 for each of the following events:

   cm3r approval mods, cm3r approved, cm3r closed, cm3r complete, cm3r deferred, cm3r disapproved, cm3r open, cm3r phase change, cm3r phase start, cm3r phase warning, cm3r reviewed, cm3r unapproved, cm3r update

You also need to create *eventout* records for Change Management tasks.

1. Type the command **db** on a ServiceCenter command line.
2. Type **cm3messages.g** in the *Form* field.
3. Click on the **Search** button to display a list of all of the Change Management event definitions.
4. Locate and open the record *cm3t approval* (note the *t* for task) and locate the *Event Services Reg* field.
5. Type **cm3tout** in this field.
6. Click the **Save** button to update this record.
7. Perform steps 5 and 6 for each of the following events:

   cm3t approval mods, cm3t approved, cm3t closed, cm3t complete, cm3t deferred, cm3t disapproved, cm3t open, cm3t phase change, cm3t phase start, cm3t phase warning, cm3t reviewed, cm3t unapproved, cm3t update

Configuring SCAuto for Lotus Notes Event Maps

SCAuto for Lotus Notes communicates with ServiceCenter using special messages known as *events*. These messages are passed between Lotus Notes and ServiceCenter. SCAuto for Lotus Notes uses special tables called *event maps* for defining the relationship between fields on a Notes form and the fields of a ServiceCenter event. These maps often come in pairs: one map for event messages sent to ServiceCenter and one map for event messages received from ServiceCenter. This is because the format of any given ServiceCenter event (such as *pmo*) differs depending on whether the event is an input or output event from the viewpoint of ServiceCenter. Other events (such as *cm3rin* and *cm3rout*) only travel in one direction.

For each event type that is to be processed by SCAuto for Lotus Notes, there needs to be a corresponding event map defined. An *event type* is defined as the combination of the event (e.g., *pmo* and *cm3rin*) and the direction it will travel (To ServiceCenter or From ServiceCenter). These event maps are located in the configuration database and are fully configurable.

The base SCAuto for Lotus Notes install includes basic event map definitions for both the Problem Management and Change Management applications. For Problem Management there are three pairs of event maps: Problem Open (*pmo*), Problem Update (*pmu*), and Problem Close (*pmc*). Change Management has six individual event maps: Request Input (cm3rin), Request Acknowledgment (cm3rinac), Request Output (cm3rout), Task Input (cm3tin), Task Acknowledgment (cm3tinac), and Task Output (cm3tout).

For each field in a Lotus Notes form that is to pass information to, or receive information from, ServiceCenter, there must be a corresponding map entry defined.

For a thorough discussion of how to create event maps, refer to the field mappings information in Chapter 6.

Additional Mappings for Change Management

ServiceCenter includes event map definitions for each of the necessary event types used with SCAuto for Lotus Notes' Problem Management and Change Management modules. Furthermore, the individual fields included in each map of a standard ServiceCenter installation are complete and sufficient for the needs of the single default form provided for the Problem Management module. The Change Management module, however, requires additional fields to be added to its maps to support the set of sample forms provided with SCAuto for Lotus Notes. As new forms are created, more fields will likely need to be added. The tables below detail the exact field mappings that need to be added to ServiceCenter's Change Management event maps. For array fields, use the pipe symbol (|)as the separator character.

The field **current.phase** is a required field. This field must be included in every ServiceCenter output event map used by Change Management. This is because SCAuto for Lotus Notes uses this field to select the proper Notes form used for displaying the information (the ticket).

Table 3-1 cm3r – Input

Position	Field	Post-Map Instructions
94	account.id	
95	actual.outage.end	
96	actual.outage.start	
97	outage.comments	
98	sched.outage.end	
99	sched.outage.start	
100	user.name	
101	down.start	
102	down.end	
103	account.type	
104	$move.flag	
105	$add.flag	
106	$change.flag	

Table 3-2 cm3r – Output

Position	Field	Post-Map Instructions
94	current.phase	
95	outage.comments	
96	sched.outage.end	
97	sched.outage.start	
98	user.name	
99	down.start	
100	down.end	
101	account.type	
102	account.id	
103	actual.outage.end	
104	actual.outage.start	

105	$add.flag (character)	cleanup($add.flag)
		if (add.flag in $axces.source=true)
		then ($add.flag="Add")

106	$change.flag (character)	cleanup($change.flag)
		if (change.flag in $axces.source=true)
		then ($change.flag="Change")

107	$move.flag (character)	cleanup($move.flag)
		if (move.flag in $axces.source=true)
		then ($move.flag="Move")

Table 3-3 cm3t – Input

Position	Field	Post-Map Instructions
100	current.phase	
101	dept	
102	version.old	
103	user.id	
104	manufacturer.old	
105	license.number	
106	contact.name	
107	contact.phone	
108	asset	
109	asset.comments	
110	ram.current	
111	hard.disc.current	
112	location.code.current	
113	down.start	
114	down.end	
115	tape.name	
116	tape.location	
117	tape.date	
118	tape.description	
119	brief.desc	
120	ram.new	
121	location.code.new	
122	size	
123	hard.disc.new	
124	install.date	

Table 3-3 cm3t – Input

Position	Field	Post-Map Instructions
125	work.start	
126	work.end	
127	work.notes	
128	account.group	
129	account.name	
130	account.type	
131	application.name	
132	application.name.old	
133	create.step	
134	delete.step	
135	restore.step	
136	test.step	
137	volume.step	

Table 3-4 cm3t – Output

Position	Field	Post-Map Instructions
100	current.phase	

101	$create.step	cleanup($create.step)
		if (create.step in $axces.source=true)
		then ($create.step="Create new netware partition")

102	$delete.step	cleanup($delete.step)
		if (delete.step in $axces.source=true)
		then ($delete.step="Delete netware partition")

Table 3-4 cm3t – Output

Position	Field	Post-Map Instructions

103	$restore.step	cleanup($restore.step)
		if (restore.step in $axces.source=true)
		then ($restore.step="Restore from tape")

104	$test.step	cleanup($test.step)
		if (test.step in $axces.source=true)
		then ($test.step="Test")

105	$volume.step	cleanup($volume.step)
		if (volume.step in $axces.source=true)
		then ($volume.step="Create new volumes")

106	user.id	
107	version.old	
108	dept	
109	manufacturer.old	
110	ram.current	
111	hard.disc.current	
112	location.code.current	
113	down.start	
114	down.end	
115	tape.name	
116	tape.location	
117	tape.date	
118	tape.description	
119	brief.desc	
120	ram.new	
121	location.code.new	
122	size	
123	hard.disc.new	

Table 3-4 cm3t – Output

Position	Field	Post-Map Instructions
124	install.date	
125	work.start	
126	work.end	
127	work.notes	
128	account.group	
129	account.name	
130	account.type	
131	application.name	
132	application.name.old	
133	asset	
134	asset.comments	
135	contact.name	
136	contact.phone	
137	network.name	
138	license.number	
139	vendor.id	
140	istatus	
141	y2k.status	
142	building	
143	floor	
144	room	

Controlling Time/Date Formatting

Unlike Problem Management, Change Management events must send time and date fields back to ServiceCenter. This can create problems because of the various formats that may be used by Lotus Notes. For example, dates may use the slash (/) or the hyphen (-) to separate the day, month, and year. Or the time may be in either 12-hour or 24-hour mode.

To solve these format differences, ServiceCenter's mappings for each date/time field must be enhanced to convert to a standard form To enhance the ServiceCenter mappings:

1. Modify the Request input map date fields.
   a. Type the command **eventmap** on a ServiceCenter command line.
   b. Type **cm3r** in the *Map Name* field and select **Input** for the *Type*. Press **Enter**. This will display a list of the fields that make up this map file.
   c. Select the *request.date* field (position 12).
   d. Select the Expressions tab. In the *Initialization* section, type the following lines:

      $r.date=12 in $axces.fields

      $r.date=translate($r.date, "-", "/")

      $am=index("AM", $r.date)

      $pm=index("PM", $r.date)

      if ($am>0 or $pm>0) then ($junk=strclpr($r.date, 3))

   e. Type the following lines in the *Post-Map Instructions* section:

      if ($pm>0) then (request.date in $axces.target=val($r.date, 3)+'12:00:00')

      if ($am>0) then (request.date in $axces.target=val($r.date, 3))

   f. Click the **Save** button to update the map.

2. Repeat steps 1c though 1f for each of the remaining date fields listed in Table 3-5. Modify the *Initialization* section slightly for each specific field as follows:

   a. For each map field, change the first line of the *Initialization* section so that the number immediately to the right of the equal sign displays the position number of this field. In the above example, *request.date* is in the 12th position of the map, so the number *12* must appear after the first equal sign. For the *assign.date* field, which occupies the 16th map position, the first line would be:

      $r.date=16 in $axces.fields

   b. Change the variable name (field name) in both lines of the **Post-Map Instructions** section to match the name of the field being modified. For example, for the field **assign.date**, the two lines would need to be changed to:

      if ($pm>0) then (assign.date in $axces.target=val($r.date, 3)+'12:00:00')

      if ($am>0) then (assign.date in $axces.target=val($r.date, 3))

   c. Make the mapping changes for every date field that is used by the Change Management forms in SCAuto for Lotus Notes. Table 3-5 contains the default list of date fields in the *cm3r map* that must be changed.

Table 3-5 cm3r map date fields

Position	Field Name
12	request.date
16	assign.date
20	coord.date
21	planned.start
24	planned.end
32	date.entered
37	orig.date.entered
41	close.time
95	actual.outage.end
96	actual.outage.start
98	sched.outage.end
99	sched.outage.start

Table 3-5 cm3r map date fields

Position	Field Name
101	down.start
102	down.end

3. Modify the Task input map date fields:
   a. Type the command **eventmap** on a ServiceCenter command line.
   b. Type **cm3t** in the *Map Name* field and select **Input** for the *Type*. Press **Enter**. This will display a list of the fields that make up the *cm3t map* file.
   c. Perform the same changes that were done in steps 3a and 3b for each of the date fields of the **cm3t map** listed in Table 3-6:

Table 3-6 cm3t map date fields

Position	Field Name
16	request.date
20	assign.date
24	coord.date
25	planned.start
28	planned.end
36	date.entered
42	orig.date.entered
47	close.time
113	down.start
114	down.end
117	tape.date
124	install.date
125	work.start
126	work.end

Chapter 4 Using Problem Management

SCAuto for Lotus Notes uses the Event Services facility of ServiceCenter to propagate problem tickets into a Lotus Notes database. SCAuto for Lotus Notes also accepts problem tickets from Lotus Notes and sends them to ServiceCenter. This functionality allows the Lotus Notes user to view ServiceCenter tickets from within a familiar Notes client as well as to open, update, and close problem tickets. Filtering can be set up so that only a subset of ServiceCenter tickets are imported into Notes (see Chapter 6 for more information). Typical Notes functions, such as text searching and creating private views, can be used on these problem tickets as well. Users are advised to become familiar with the capabilities and uses of Lotus Notes in order to make full use of SCAuto for Lotus Notes.

When problems are opened, updated, or closed from within ServiceCenter, events are created in the *eventout* file managed by Event Services. The SCAuto for Lotus Notes server add-in communicates with ServiceCenter to retrieve these events. The events are then translated into notes and added to the SCAuto for Lotus Notes database where any Notes client with proper access and permissions can access the notes and modify them. The SCAuto for Lotus Notes server add-in detects those problem tickets that are newly opened from within Notes, or those existing tickets that have been updated or closed. SCAuto for Lotus Notes translates these notes into ServiceCenter events and places them into the *eventin* file of ServiceCenter.

If the SCAuto for Lotus Notes server add-in is not running, or communication to ServiceCenter is temporarily unavailable, the notes in the SCAuto for Lotus Notes database can still be viewed, searched, and modified by Lotus Notes users.

Only a subset of the fields used in the Problem Management application of ServiceCenter is exported to SCAuto for Lotus Notes (and other external applications that read ServiceCenter events). SCAuto for Lotus Notes communicates with ServiceCenter only through the Event Services facility. SCAuto for Lotus Notes is not a replacement for a ServiceCenter client.

The SCAuto for Lotus Notes database provides a default set of forms, views, and macros that allow easy manipulation of problem tickets. The SCAuto for Lotus Notes server add-in has little knowledge of this database format. Notes developers can customize a significant portion of the SCAuto for Lotus Notes user interface, either to adapt it to local usage and language, implement Notes security, or add any number of other features.

Selecting Views in the Database

Lotus Notes displays groups of notes by using *views*. The notes in a view are displayed in column format, with optional categorization. To select a different view than the default, use the **View** menu and select one of the choices at the bottom of the menu. From within a view, rows represent problem tickets. These tickets can be viewed by double clicking on them. Problem tickets that have not been viewed by the current user are highlighted in a different color and have a star icon in the leftmost column. After viewing a note, pressing the **Esc** key closes the window containing the note and returns to displaying the view.

The following Problem Management views are predefined in the SCAuto for Lotus Notes Help Desk database and are available from the Lotus Notes menu:

by Number	This is the most basic view. By Number lists all problems sorted by problem number with no categorization.
by Category	Problem tickets are categorized based upon the ServiceCenter category that they were assigned to, such as *Equipment, Software*, etc.
by Status	Problem tickets are categorized by their status. That is, *Opened*, *Closed*, *Alert stage 1*, etc. The group of tickets within a category can be hidden or expanded by double clicking on that category's row in the view.
by Assignment	Problem tickets are categorized by the user they are assigned to.

Users are allowed to create their own private views that can be customized on a per-user basis. For instance, problems may be sorted based upon the last update time or categorized via the problem's priority. Further information on how to do this can be found in the ***Lotus Notes Application Developer's Reference*** manual.

Composing a New Problem Ticket

To open a new ServiceCenter problem ticket from within Lotus Notes:

1. Click on **New Problem** from the **Create** menu.

   This option creates a blank note and presents the note with editing enabled.

2. Fill in the various fields. The mouse or the arrow keys can be used to move the editing cursor.

   Note: The **priority** field only allows a limited set of values, and these may be cycled through by pressing the space bar repeatedly.

3. Click the **Save** button to save the changes and return to the previous view or form. (Pressing the **exit** button causes Lotus Notes to ask if you want to save your changes.)

   The new note is assigned a problem number of **0** initially. This number is updated once ServiceCenter has actually opened the problem ticket and assigned a number to it.

Updating, Closing and Deleting Problem Tickets

Selecting **Update** puts the current document into editing mode, so that the fields may be updated. The cursor is automatically placed in the **Updates** field. Editing is done in the same manner as for new problems, with **Save** and **Exit** buttons used to finish editing.

To close a ticket, click on the **Close** button. This button functions the same as updating a ticket, except that the status is filled in with **closed**, and the cursor will initially be in the **Resolution** field. Note that the status must be left as **closed**. If it is not, the changes are treated as an update.

Closed tickets are not automatically removed from the SCAuto for Lotus Notes database. In order to delete a ticket that is no longer desired, select the ticket in a view and press the **Del** key. This marks the note to be deleted, but it will not actually be removed until you leave or refresh the view. Alternatively, you may use the **Edit>Cut** menu option, which deletes the selected note immediately. ServiceCenter is not informed when a ticket is deleted so this action does not close the ticket or prevent further updates of the ticket from appearing. Set up security so that unprivileged users cannot delete problem tickets.

Chapter 5 Using Change Management

SCAuto for Lotus Notes uses the Event Services facility of ServiceCenter to propagate change request and task tickets into a Lotus Notes database. It can also accept these tickets from Lotus Notes and send them to ServiceCenter. The communication and interaction is fully bi-directional. See Chapter 4 for a general discussion of how SCAuto for Lotus Notes operates and how tickets are processed.

Change Management uses three pairs of single-direction events for processing change tickets. Each pair includes one event for requests and another for tasks. The names of the events are the same except for the use of an *r* in the name to designate requests, and a *t* for tasks:

cm3rin cm3tin

These events are used to send all request and task ticket information to ServiceCenter. The actions that are performed by these events are: open, update, approve, disapprove, unapprove, close, and reopen.

cm3rinac cm3tinac

These are acknowledgment events sent from ServiceCenter in response to receiving a *cm3rin* or *cm3tin* event. Every *cm3rin* or *cm3tin* event will generate one *cm3rinac* or *cm3tinac* event, respectively. These events perform two important functions:

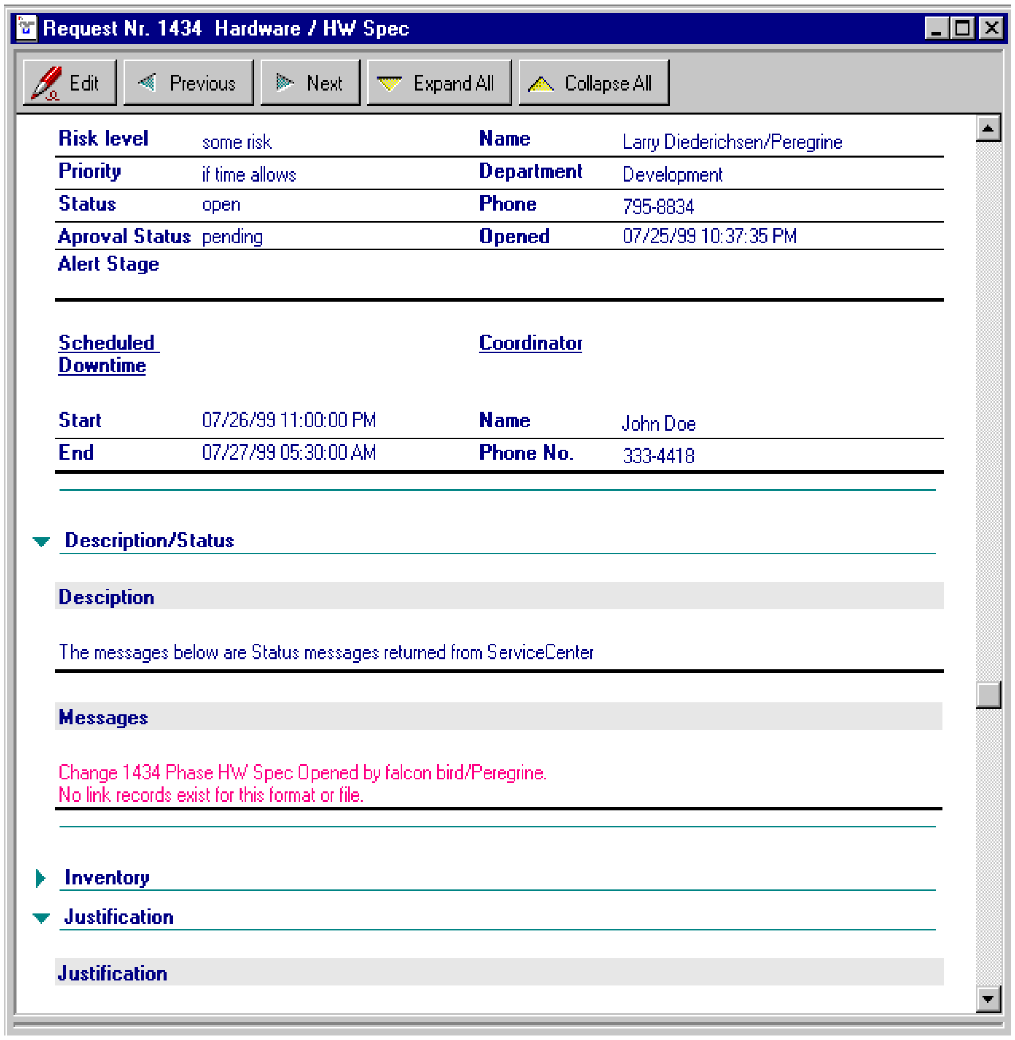
- When opening a new request or task, returns the ticket number that is assigned by ServiceCenter.
- Returns status information and/or error messages. These are displayed in the form under the **Description/ Status** section. Figure 5-1 shows a request form with this section expanded.

cm3rout cm3tout

These two event messages transfer most of the ticket information that comes from ServiceCenter. After sending the acknowledgment event (*cm3rinac* or *cm3tinac*), ServiceCenter typically follows this with a series of *cm3rout* or *cm3tout* events. Sometimes there will only be one of these events sent. If there are errors with the original open, there may be no *cm3rout* or *cm3tout* sent. Usually there are two or more, each carrying different information about the actions that ServiceCenter is taking. For example, approval information for requests and tasks is always sent down in a separate event.

For requests that automatically open tasks, there is a *cm3tout* event sent from ServiceCenter for each task. These events follow the *cm3rinac* event and all of the *cm3rout* events. SCAuto for Lotus Notes then creates a new task note for each *cm3tout* message.

Figure 5-1 is a sample of a request form showing the status messages returned from ServiceCenter (in red):



Risk level	some risk	Name	Larry Diederichsen/Peregrine
Priority	if time allows	Department	Development
Status	open	Phone	795-8834
Aproval Status	pending	Opened	07/25/99 10:37:35 PM
Alert Stage			

Scheduled Downtime

Coordinator

Start	07/26/99 11:00:00 PM	Name	John Doe
End	07/27/99 05:30:00 AM	Phone No.	333-4418

Description/Status

Desciption

The messages below are Status messages returned from ServiceCenter

Messages

Change 1434 Phase HW Spec Opened by falcon bird/Peregrine.
No link records exist for this format or file.

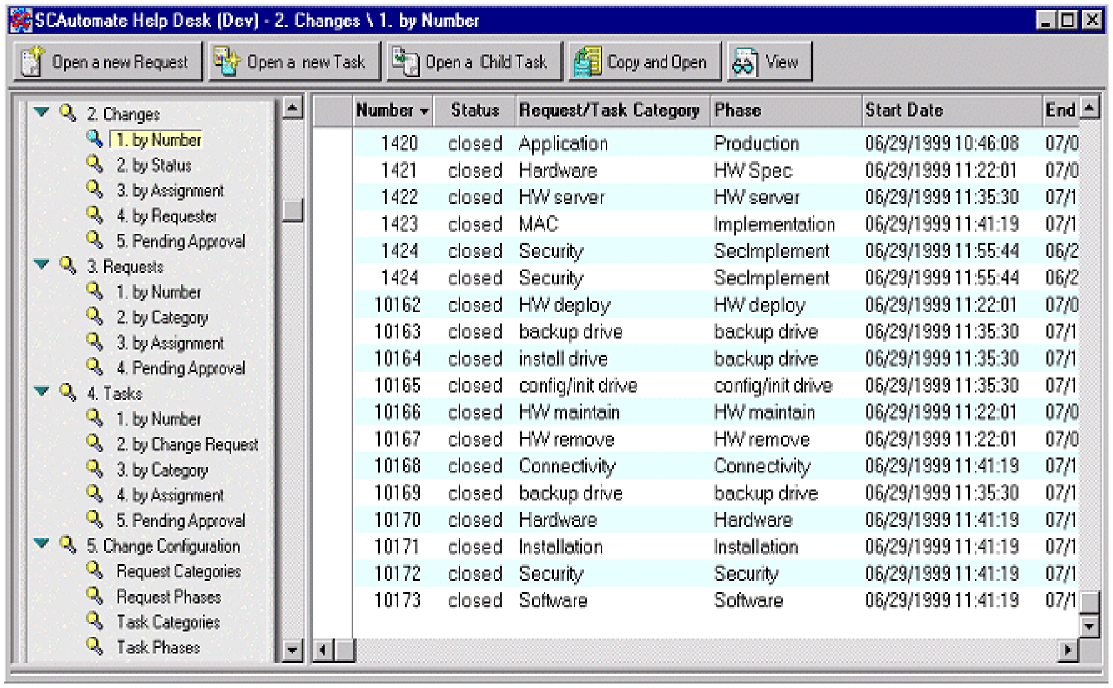
Inventory

Justification

Justification

Figure 5-1. Request form

Selecting Views in the Database



- 2. Changes
  - 1. by Number
  - 2. by Status
  - 3. by Assignment
  - 4. by Requester
  - 5. Pending Approval
- 3. Requests
  - 1. by Number
  - 2. by Category
  - 3. by Assignment
  - 4. Pending Approval
- 4. Tasks
  - 1. by Number
  - 2. by Change Request
  - 3. by Category
  - 4. by Assignment
  - 5. Pending Approval
- 5. Change Configuration
  - Request Categories
  - Request Phases
  - Task Categories
  - Task Phases

Number	Status	Request/Task Category	Phase	Start Date	End
1420	closed	Application	Production	06/29/1999 10:46:08	07/0
1421	closed	Hardware	HW Spec	06/29/1999 11:22:01	07/0
1422	closed	HW server	HW server	06/29/1999 11:35:30	07/1
1423	closed	MAC	Implementation	06/29/1999 11:41:19	07/1
1424	closed	Security	SecImplement	06/29/1999 11:55:44	06/2
1424	closed	Security	SecImplement	06/29/1999 11:55:44	06/2
10162	closed	HW deploy	HW deploy	06/29/1999 11:22:01	07/0
10163	closed	backup drive	backup drive	06/29/1999 11:35:30	07/1
10164	closed	install drive	backup drive	06/29/1999 11:35:30	07/1
10165	closed	config/init drive	config/init drive	06/29/1999 11:35:30	07/1
10166	closed	HW maintain	HW maintain	06/29/1999 11:22:01	07/0
10167	closed	HW remove	HW remove	06/29/1999 11:22:01	07/0
10168	closed	Connectivity	Connectivity	06/29/1999 11:41:19	07/1
10169	closed	backup drive	backup drive	06/29/1999 11:35:30	07/1
10170	closed	Hardware	Hardware	06/29/1999 11:41:19	07/1
10171	closed	Installation	Installation	06/29/1999 11:41:19	07/1
10172	closed	Security	Security	06/29/1999 11:41:19	07/1
10173	closed	Software	Software	06/29/1999 11:41:19	07/1

Figure 5-2. Views

The following Change Management views are predefined in the SCAuto for Lotus Notes Help Desk database. These views are grouped as changes, requests, tasks, or change configuration.

Changes

by Number	Lists all changes (requests and tasks) sorted by number with no categorization.
by Status	Change tickets are categorized according to the current status of the ticket.
by Assignment	Change tickets are categorized by the user they are assigned to.

by Requester	Tickets are categorized by the user name of the original requester of the change.
Pending Approval	Tickets are categorized by the first approval group that has yet to approve the change. The list of approval groups assigned to a change, and their status, can be seen by viewing the Approval Information section of the ticket.

Requests

by Number	Lists all requests sorted by request number with no categorization.
by Category	Request tickets are categorized based upon the change request category they were created under, such as *Hardware, Security,* etc.
by Assignment	Request tickets are categorized by the user they are assigned to.
Pending Approval	Request tickets are categorized by the first approval group that has yet to approve the request. The list of approval groups assigned to a request, and their status, can be seen by viewing the Approval Information section of the ticket.

Tasks

by Number	Lists all tasks sorted by task number with no categorization.
by Change Request	All tasks are associated with a child of a parent request. This view categorizes the tickets by their parent change request number.
by Category	Task tickets are categorized based upon the task category they are assigned to.
by Assignment	Task tickets are categorized by the user they are assigned to.
Pending Approval	Task tickets are categorized by the first approval group that has yet to approve the task. The list of approval groups assigned to a task, and their status, can be seen by viewing the Approval Information section of the ticket.

Change configuration

Request Categories

Lists all of the request category definitions. Displayed columns are *Name*, *Description*, and *Phases* (the phases that constitute this category).

Request Phases

This view lists all of the request phase definitions. Displayed columns are *Name*, *Description*, and *Form* (the Notes form needed to view this type of request phase).

Task Categories

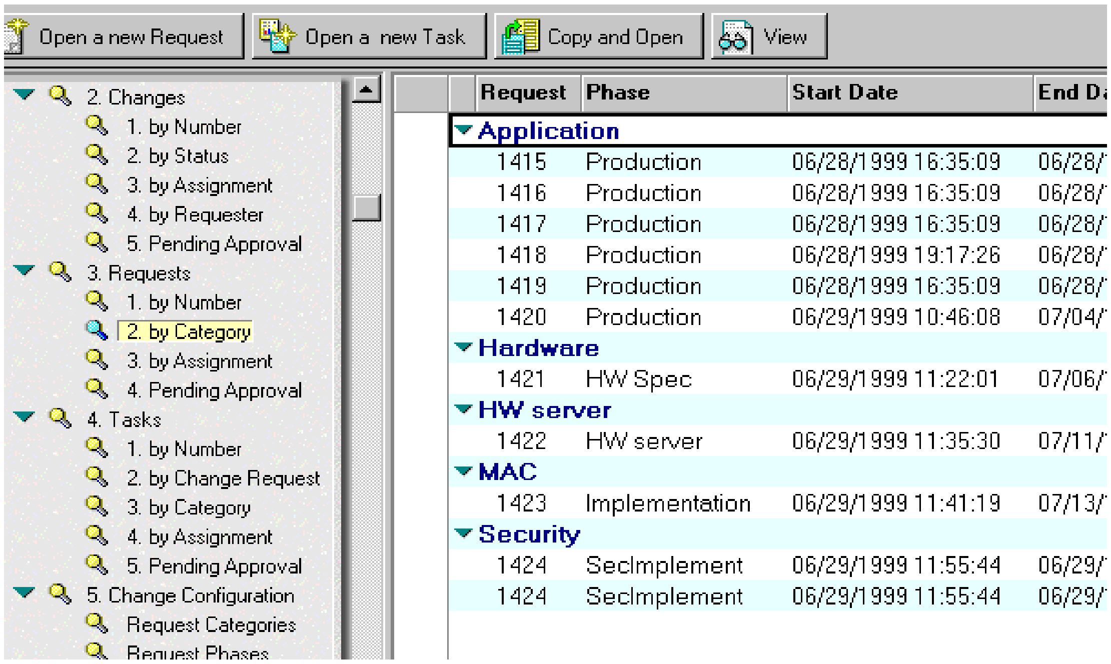
Lists all of the task category definitions. Displayed columns are *Name*, *Description*, *Used in Request Phases* (lists the request phases that can use this task category) and *Phases* (the phases that constitute this category).

Task Phases Lists all of the task phase definitions. Displayed columns are *Name*, *Description*, and *Form* (the Notes form used to view this task phase).

Opening a New Change Ticket

Invoking actions on change requests is done slightly different than with problems. For problems, some actions are performed using menu items (for instance, opening a new problem) while others are done using buttons that are placed directly on the form. All functions within Change Management are implemented with action buttons that reside at the top of the window, just below the title bar and above the view or form window.

To open a new request or task, you must start by selecting one of the Change Management views. See *Selecing Views in the Database* on page 5-4 for a detailed list of all of the available views. Figure 5-3 shows the view *Requests by Category*. Notice the buttons at the top for opening a request or task. The buttons available on a view or form will change depending on the type of view, the type of ticket (request or task), and the state of the ticket. For example, the view *Changes by Number*, described on page 5-4, includes **Open a Child Task**, which is only applicable for views that show tasks.



Request	Phase	Start Date	End Da
Application			
1415	Production	06/28/1999 16:35:09	06/28/
1416	Production	06/28/1999 16:35:09	06/28/
1417	Production	06/28/1999 16:35:09	06/28/
1418	Production	06/28/1999 19:17:26	06/28/
1419	Production	06/28/1999 16:35:09	06/28/
1420	Production	06/29/1999 10:46:08	07/04/
Hardware			
1421	HW Spec	06/29/1999 11:22:01	07/06/
HW server			
1422	HW server	06/29/1999 11:35:30	07/11/
MAC			
1423	Implementation	06/29/1999 11:41:19	07/13/
Security			
1424	SecImplement	06/29/1999 11:55:44	06/29/
1424	SecImplement	06/29/1999 11:55:44	06/29/

Figure 5-3. Requests by category

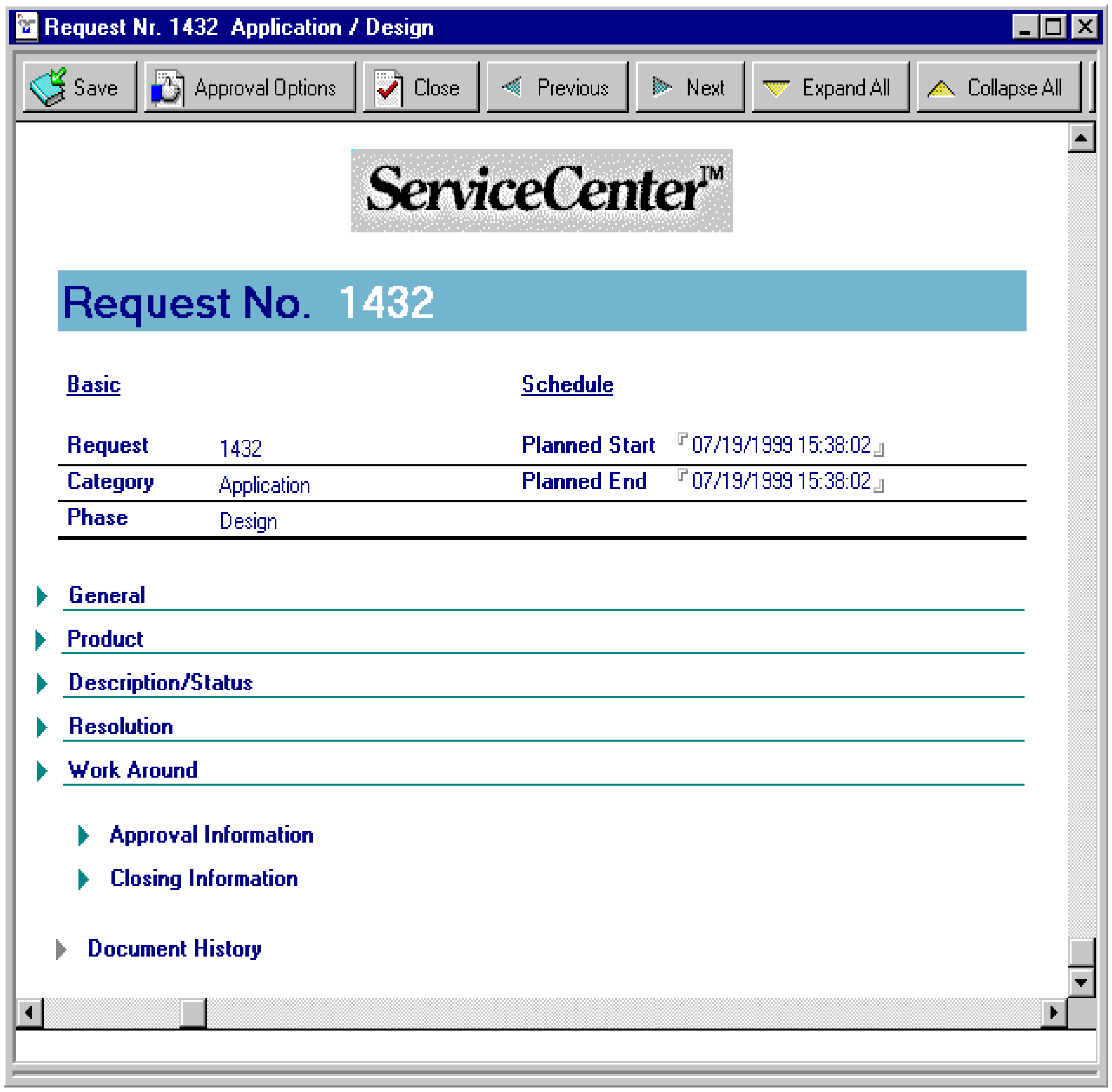
To open a new request or task, click on one of the open buttons. This brings up a dialog requesting that a category be selected. Every request and task must be assigned to a category. The list of categories presented is set by the categories defined. See the section *Defining Change Categories and Phases* in the Chapter 6 for details.

Once the category has been selected, SCAuto for Lotus Notes searchs the category and phase definitions to find the name of the first phase of the category. Every category is composed of one or more phases, or steps. Each phase of the category may use a different Notes form for viewing and editing. Once the first phase is identified, it is used in another lookup to get the name of the form associated with this phase. With this form, SCAuto for Lotus Notes can finally create a new note and open it.

After filling in the appropriate fields, click on the **Save** button to save the form, close the window, and send the ticket to ServiceCenter.

Note: Always use the action buttons for saving and closing forms. Closing a window (form) in any other way will prevent proper processing of the ticket!

Figure 5-4 shows an example of a change request form that has been opened and assigned a ticket number by ServiceCenter.


Request Nr. 1432 Application / Design

Request No. 1432

Basic

Request	1432	Planned Start	07/19/1999 15:38:02
Category	Application	Planned End	07/19/1999 15:38:02
Phase	Design		

Figure 5-4. Change request form

Updating and Approving Changes

This section describes the procedures for updating and approving changes.

Updating tickets

To update a change ticket:

1. Select the ticket to update from one of the views.
2. Click the **View** button to open the ticket for viewing, or double-click the entry in the view.
3. Click the **Edit** button to put the form into edit mode.
4. Click on the **Save** button after making your changes.

Note: Always use the action buttons for saving and closing forms. Closing a form in any other way will prevent SCAuto for Lotus Notes from processing the ticket properly.

Approval options

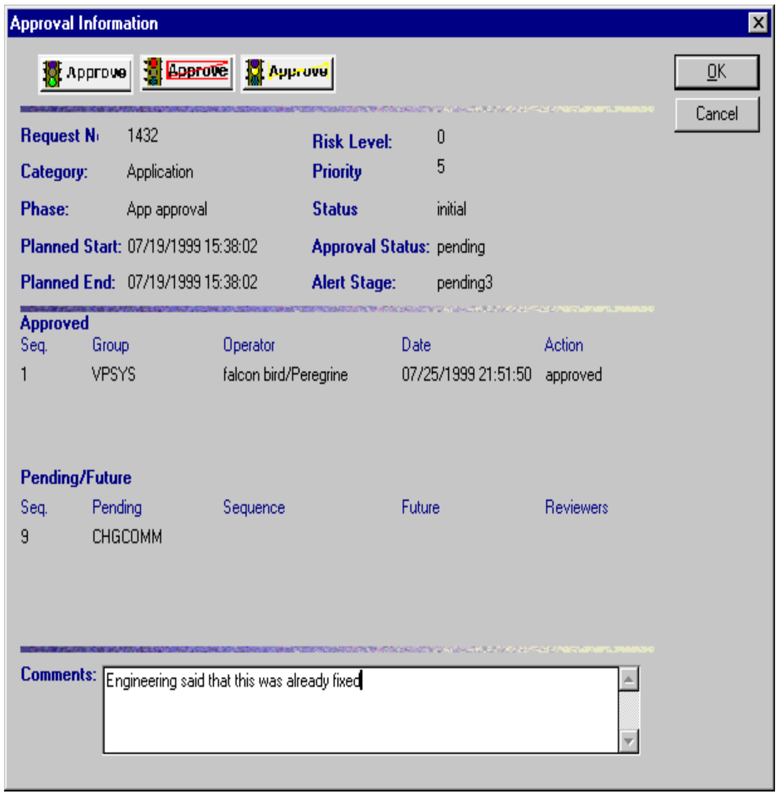
If the request or task ticket requires approvals from one or more groups, then the **Approval Options** button is visible while the ticket is in Edit mode. The approval information can be viewed in the ticket by expanding the section labeled *Approval Options* towards the bottom of the form. This section shows both the pending approvals (still to be approved) and also those that have already been approved.

To make a change to the approvals, click on the **Approval Options** button. This brings up the *Approval Options* dialog. This dialog presents some basic information about the ticket as well as a listing of the groups that have already been approved, plus the next pending approval group.

Action buttons are displayed at the top of the form:

Approve	Approves the change request.
Disapprove	Disapproves the change request.
Unapprove	Undoes the previous approval, listed in the Approved section.

1. Click on the appropriate button to perform the desired action.
2. Click the **OK** button to save your changes, or click **Cancel** to discard any changes.



Request N	1432	Risk Level:	0
Category:	Application	Priority	5
Phase:	App approval	Status	initial
Planned Start:	07/19/1999 15:38:02	Approval Status:	pending
Planned End:	07/19/1999 15:38:02	Alert Stage:	pending3

Approved

Seq.	Group	Operator	Date	Action
1	VPSYS	falcon bird/Peregrine	07/25/1999 21:51:50	approved

Pending/Future

Seq.	Pending	Sequence	Future	Reviewers
9	CHGCOMM			

Comments: Engineering said that this was already fixed

Figure 5-5. Approval options

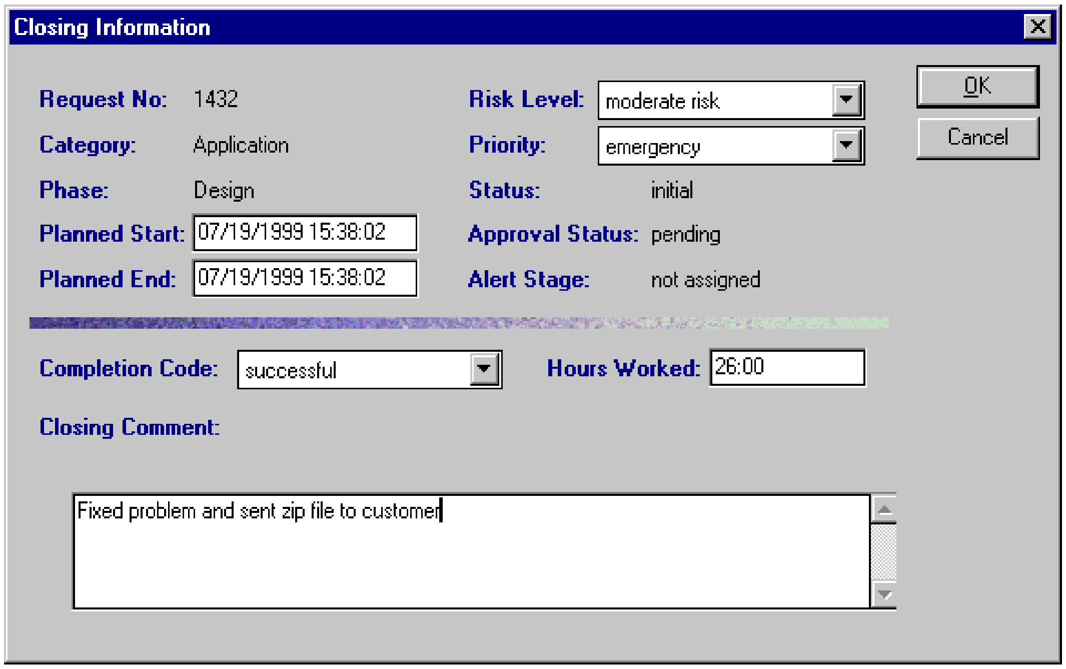
Closing and Reopening Change Tickets

Closing tickets

To close a ticket:

1. Click the **Edit** button.
2. Click the **Close** button at the top of the form.

   The Closing Information dialog box is displayed:



Request No: 1432
Category: Application
Phase: Design
Planned Start: 07/19/1999 15:38:02
Planned End: 07/19/1999 15:38:02
Risk Level: moderate risk
Priority: emergency
Status: initial
Approval Status: pending
Alert Stage: not assigned
Completion Code: successful
Hours Worked: 26:00
Closing Comment: Fixed problem and sent zip file to customer

Figure 5-6. Closing information dialog box

3. Fill in the form, including the Closing Comments field.
4. Click the **OK** button to save the changes.

   The main form is displayed.
5. Click the **Save** button to process the close request.

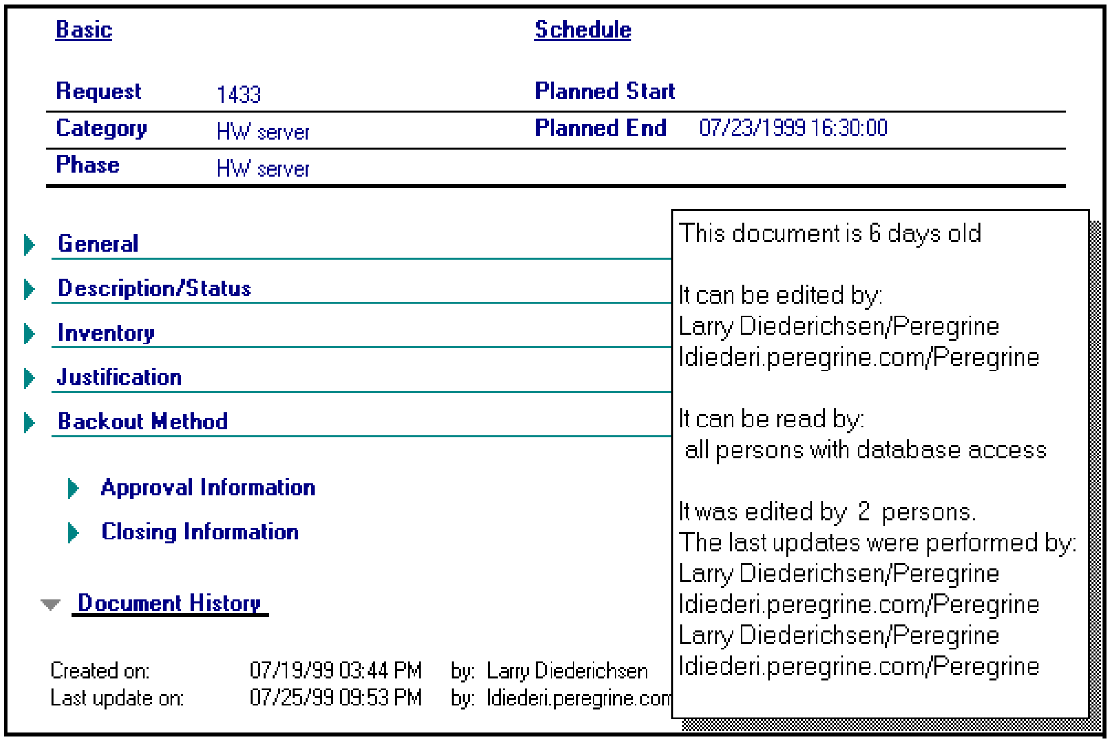
Reopening tickets

To reopen a ticket that has a status of *closed*:

1. Click the **Edit** button to put the ticket in edit mode.
2. Click the **Reopen** button.
3. Fill in the appropriate information.
4. Click the **Save** button to process the request.

Document History

The Document History section at the bottom of each form displays details about the modification history and security settings of the document. Clicking on the green hot spot labeled *Document Info* causes a popup window to be displayed that shows more detailed information.



Basic

Request	1433	Planned Start	
Category	HW server	Planned End	07/23/1999 16:30:00
Phase	HW server		

General

Description/Status

Inventory

Justification

Backout Method

Approval Information

Closing Information

Document History

Created on:	07/19/99 03:44 PM	by:	Larry Diederichsen
Last update on:	07/25/99 09:53 PM	by:	ldiederi.peregrine.com

This document is 6 days old

It can be edited by:
Larry Diederichsen/Peregrine
ldiederi.peregrine.com/Peregrine

It can be read by:
all persons with database access

It was edited by 2 persons.
The last updates were performed by:
Larry Diederichsen/Peregrine
ldiederi.peregrine.com/Peregrine
Larry Diederichsen/Peregrine
ldiederi.peregrine.com/Peregrine

Figure 5-7. Document history

Action Buttons

Following is a list of all action buttons and their functions:

View actions

Open a New Request

Opens a new change request. This button brings up a dialog box that lists the available request categories you can select.

Open a New Task

Opens a new task. This button presents a dialog requesting the user to select a task category from the list shown. Tasks are always associated with a specific change request, so a valid request, with an assigned ticket number, must be selected prior to opening a new task.

Open a Child Task

Similar to the Opening a New Task button except that it requires that a valid **task** (not request) must be selected first, and it must have a valid ticket number assigned. In this case, SCAuto for Lotus Notes will first get the parent request of this task and then simulate an *Open a new Task*.

Copy and Open

Provides a convenient way to open a new request or task that is to contain a lot of information that is the same as another ticket. This can save time and effort, plus reduce errors. After copying the original ticket, SCAuto for Lotus Notes then clears the *number* and *messages* fields, and then sets *status* to *initial* and *action* to *open*.

View Opens the selected ticket for viewing.

Form actions

Edit	Puts the form into edit mode to allow changes.
Previous	Changes the ticket being displayed to the previous one listed in the view.
Next	Changes the ticket being displayed to the next one listed in the view.
Expand All	Expands every section in the form so that all ticket data can be seen. Each section is indicated by a green triangle on the left followed by the name of the section. Clicking on the triangle or the name alternately expands or collapses that individual section.
Collapse All	Collapses every section in the ticket so that only the header information and the section names are visible. Each section is indicated by a green triangle on the left followed by the name of the section. Clicking on the triangle or the name alternately expands or collapses that individual section.
Time	Inserts the current time and date (in proper format) into the current field.
Save	This button: • Saves the document • Closes the window (form) • Causes an event to be sent to ServiceCenter. This button must be used to save and close a form. Closing a form in any other way will prevent proper processing of the ticket.
Approval Options	Brings up the *Approvals* dialog.
Close	Invokes the *Close* dialog.
Reopen	Sets the action to *reopen* so that the ticket will be reopened by ServiceCenter. This button is only available on tickets with a status of *closed*.

Chapter 6 Customization

Filtering Events

SCAuto for Lotus Notes provides two separate mechanisms for filtering the events that are processed:

- Filtering with configuration variables
- Filtering with a macro formula

Filtering with configuration variables

The first, and most basic, way to filter out events is to use the three filtering fields contained in the configuration note.

Create ID This field specifies the event user name evuser for all newly created events. This name can be used for selecting (filtering) the set of events that will be retrieved from ServiceCenter. See the following field description, Event Username Filter, for details. If left blank, the default name of scauto is used.

Event Username Filter

This is one of the two fields that can be used for selecting which events will be retrieved from ServiceCenter. Only events that were created by one of the members in this list of usernames are retrieved. If this field is the keyword all or is left blank, then events created by all users will be retrieved. To list multiple usernames, separate them with commas and enclose them in parentheses. For example: John,Lucy,George

Event Type Filter

This field contains the list of event types that will be retrieved from ServiceCenter. Separate the types with commas and enclose the string in parentheses. For example, (pmo,pmu,pmc) defines the three Problem Management events: open, update, and close. Only events of these types are retrieved. If left blank, the default filter (pmo,pmu,pmc,cm3rin,cm3rinac,cm3rout,cm3tin,cm3tinac,cm3tout) is used.

Filtering with a macro formula

Incoming events from ServiceCenter are passed through another filtering function before being added to Lotus Notes. This function allows Notes to only deal with a relevant subset of the events from ServiceCenter. This filter is optional and fully customizable by anyone with designer access to the SCAuto for Lotus Notes database. This filter has more capability than just filtering.

The macro formula filter is stored in the *SelectEventsMacro* macro in the configuration database. This filter is not a Lotus Notes filter macro. Although this filter is accessed as a macro, it is not executed as a macro, and all macro flags are ignored. It is stored in a macro only for convenience (so it is easy to find and customize).

Before any incoming problem or change event from ServiceCenter is added to Notes, the *SelectEventsMacro* formula is evaluated in the context of that note. If this problem or change already exists in the SCAuto for Lotus Notes database, then it is updated with the modifications from ServiceCenter. If this is a new problem or change, then it is only added to the database if this filter selects the note with the SELECT statement. Thus, new problems and changes are only added to Notes if they pass this filter, whereas existing notes are updated regardless of the filter.

To change the macro formula filter:

1. Open the SCAuto for Lotus Notes configuration database.
2. Select **Agents** from the main view.
3. Double-click on the **SelectEventsMacro** note in the view pane.
4. In the default formula, give values to the variables *category*, *assignment*, *priority*, and *severity*. If a value is given, then only those problems or changes matching those values are added to SCAuto for Lotus Notes. For example, to restrict SCAuto for Lotus Notes to only problems in the *equipment* category, change the appropriate line to read:

   category := "equipment";

Note: This formula may do more than just filter out problems. It has the capability of performing many other actions. For instance, it may add or change fields, set security based upon the assignment group, and many other actions. However, since this filter is called from a server add-in, some functions are not available. In particular, *@Command*, *@DbLookup*, *@DbColumn*, and *@MailSend* do not work.

Mapping Fields

SCAuto for Lotus Notes event maps provide the association (mapping) between Lotus Notes fields and ServiceCenter's Event Services event maps. The event maps are located in the configuration database. For an overview of event maps, see *Configuring SCAuto for Lotus Notes Event Maps* in Chapter 3.

To view the maps

1. Choose **Admin>Event Maps** from the **View** menu.
2. Double-click on a map in the view to open it for viewing.

To edit a map :

1. Click the **Edit Document** button at the top of the form. This puts the document in edit mode and exposes the following action buttons:

Save	After completing your changes, clicking this button saves the document and closes the form.
Exit	This button closes the form without saving.
Import Map	This button will allow for the definition of this map to be read in from an external ASCII file which was previously created with the Export Map button described next. Clicking Import Map will bring up a dialog that will allow the user to specify the import file name and location. Importing a map definition will fill in every field of the map, including all of the special control fields in the map header.
Export Map	This button performs the opposite action from Importing. It will create an ASCII text file that includes the entire definition of the map. The file created can later be used for importing. A dialog box is displayed so that the filename and location can be specified.

2. Click the desired action button.

Map header

The first two fields, together, identify the map:

Event Type Identifies the ServiceCenter event for which this map will be used. For new events, this keyword field accepts undefined keywords (or new keywords can easily be added to the list).

Event Direction

Set the direction in which the event message will travel: *To-ServiceCenter* or *From-ServiceCenter*.

The remainder of the header contains a number of fields that control various aspects of the processing of events that use this map. These fields are only available in maps with an event direction of *From-ServiceCenter*:

Lookup Field Slot

Identifies the field (via its slot number) that SCAuto for Lotus Notes will use to locate an existing note for updating. If an existing note contains the same value for this field as the incoming event, then this note is updated. If there is no match, then the Note IDs are compared. If that fails to find a match, then a new note is created.

Lookup View This field allows for specifying a user-managed view that provides the source of the notes that are searched when locating a note to update. For large databases, this may give better performance. If left blank (the default), then the entire database is searched.

Form Designates the form to associate with an incoming event. There is a mechanism provided for setting the form dynamically. If the value is a number enclosed in angle brackets (<>), then this number is interpreted as a slot in this map. The field for this slot is used to query the incoming event. The value of this field (in the incoming event) is used as the lookup key in the view identified by *Form View*. The resulting unique document contains the form name in the field given by *Form Field*.

Form View When setting the form dynamically, this field names the view that will be used for the lookup of the form name (see the previous field *Form*, for details). Two event maps, *cm3rout* and *cm3tout*, use this mechanism for dynamically setting the form.

Form Field	For dynamically assigned forms (see the previous two fields), specify the name of the field that contains the form name. This field must exist in each document that is selected by the view specified by *Form View*.
Doc Type	Specifies the type of document. This is used as a filter in some of the view formulas.
Delete Events	Determines the fate of the events in ServiceCenter's output event queue. The default is *No* which tells SCAuto for Lotus Notes to leave the events in the queue. If set to *Yes*, SCAuto for Lotus Notes deletes each event (that uses this event map) from the queue after it is successfully processed. The event is not deleted if there are processing problems or errors.

Map body

The body of the map defines that actual field mappings between Lotus Notes and ServiceCenter. See the ***ServiceCenter Base Utilities*** manual for complete information on Event Services and event mappings.

Rows

Each slot in a table corresponds directly to a subfield position in the *evfields* field of the event structure that makes up a ServiceCenter event message. When looking at ServiceCenter event queue entries, this field is identified as the *External Information String*. The table is initially created with 250 slots but can be increased to a maximum of 999 slots. When adding slots, follow the same field-naming scheme.

Columns

The table contains two columns. The second column, although always visible, is only functional for incoming event maps (Event Direction = From-ServiceCenter):

Field Name	Contains the name of a Lotus Notes field. Literal fields can also be defined for outgoing messages by enclosing the text in double-quotes. In this case, the double-quotes are removed and the remaining string is placed, as is, directly into the event message.
Default if Null	This column provides control over the action taken when event fields coming from ServiceCenter are empty (null). When a field of an event is empty, one of three possible actions can be taken with the corresponding field in the receiving note:

Value	Action Taken
(empty field)	Field will not be modified.
NULL	Field will be replaced will an empty string
Replacement-Text	Field will be replaced with this replacement text

Security

Only minimal security is implemented in the default SCAuto for Lotus Notes databases. The capability exists for more security by using the facilities within Lotus Notes. Incoming events do not currently create any fields with the types *Author Names* or *Reader Names*.

At a minimum, you should restrict author access to the SCAuto for Lotus Notes databases to only privileged users. ServiceCenter events do not map well into the Document/Response model used by Lotus Notes. Different users may need to edit the same note within SCAuto for Lotus Notes, and these notes may have been created by the SCAuto for Lotus Notes server add-in. Users who are updating tickets from Lotus Notes will likely need editor access instead of just author access to edit notes owned by other users.

Defining Change Categories and Phases

Each change request and task is assigned to a category when opened. These categories define the type of request or task. Included in the definition of each category is a list of the phases (steps) that comprise that category. Every category must contain at least one phase. SCAuto for Lotus Notes needs to know these phases so that it can select the proper phase-specific Notes form to use for displaying the ticket information. Each request and task phase defined must have an accompanying Notes form designed specifically for that phase. Phases may, however, share a form, if appropriate.

To view the categories and phases, select **Change Configuration** from the View menu. There are four views provided:

- Request Categories
- Request Phases
- Task Categories
- Task Phases

For each view, there are two action buttons displayed at the top: one for editing an existing definition, the other for creating a new definition. When defining new categories and phases, make sure that you include the following:

Request Categories

In the *Phases* field, list, in order, each of the phases that make up the category; one phase name per line.

Request Phases

In the *Form* field, enter the name of the Notes form that will be used for viewing this phase. See *Creating Request Forms* and *Creating Task Forms* later in this chapter for a discussion of how to create these forms.

Task Categories

List the phases of the task, one per line and in proper order, in the field *Task Phases*. In the field *Available Change Phases*, list each change request phase that can use this task category.

Task Phases List the name of the Notes form to use for viewing this task phase.

The base install of SCAuto for Lotus Notes includes the following category and phase definitions along with their associated Notes form.

Table 3-1 Request Categories

Request Category	Description	Phases
Application	Production Application Changes	Design App approval QA Production
Hardware	Hardware Move/Add/ Changes	HW Spec
HW Server	Hardware Server	HW Server
MAC	General Purpose Move/ Add/Change	HW Server
Security	Request Change/Add/ Delete of User Accounts	Analysis Approval Testing Implementation

Table 3-2 Request Phases

Request Phase	Description	Form
Analysis	Analysis Of Work	cm3r.mac
App approval	Approving application changes	cm3r.application
Approval	Supervisor Approval	cm3r.mac
Design	Steps for the application upgrade	cm3r.application
HW server	Hardware Server	cm3r.HW.server
HW Spec	Hardware Specification Planning	cm3r.hardware
Implementation	Place Changes Into Service	cm2r.mac
Production	Implementation into Production	cm3r.application
QA	Quality Assurance Testing	cm3r.application
SecApproval	Approve Chg/Rem/Add of user accts	cm3r.security
SecImplement	Implement Chg/Rem/Add of user accts	cm3r.security
Testing	Test Changes	cm3r.mac

Table 3-3 Task Categories

Task Category	Description	Phases	Used in Request Phases
backup drive	Backup Drive	backup drive	HW server
config/init drive	Configure/ Initialize Drive	config/init drive	HW server
Connectivity	Network Maintenance	Connectivity	Analysis Approval Testing Implementation
Hardware	General Hardware Changes	Hardware	Analysis Approval Testing Implementation
HW deploy	Hardware Deployment	HW deploy	HW Spec Approval Implementation
HW maintain	Hardware Maintenance	HW maintain	HW Spec Approval Implementation
HW remove	Hardware Removal	HW remove	HW Spec Approval Implementation
install drive	Install Drive	install drive backup drive	HW server
Installation	Installation Procedures	Installation	Analysis Approval Testing Implemementation
Security	Security Maintenance	Security	Analysis Approval Testing Implemementation

Table 3-3 Task Categories

Task Category	Description	Phases	Used in Request Phases
Software	General Software Changes	Software	Analysis Approval Testing Implememtation

Table 3-4 Task Phases

Task Phase	Description	Form
backup drive	Backup Drive	cm3t.backup.drive
config/init drive	Configure/Initialize Drive	cm3t.config/init.drive
Connectivity	Network Maintenance	cm3t.connectivity
Hardware	General Hardware Changes	cm3t.hardware
HW deploy	Hardware Deployment	cm3t.HW.deploy
HW maintain	Hardware Maintenance	cm3t.HW.maintain
HW remove	Hardware Remove	cm3t.HW.remove
install drive	Install Drive	cm3t.install.drive
Installation	Installation Procedures	cm3t.installation
Security	Security Maintenance	cm3t.security
Software	General Software Changes	cm3t.software

Creating Request Forms

Forms

Request and task forms are stored in the main Help Desk database. The easiest way to create a new form is to copy an existing request form, change its name, and then edit it.

To copy a form:

1. Click on **Design** from the **Forms** menu to select the form you want to copy.
2. Use the **Cut** and **Paste** functions under the **Edit** menu to copy the form.

Note: All request form names begin with *cm3r* and task forms begin with *cm3t*.

To change the form's name:

1. Open the form by double-clicking on the form's name from the **Design>Forms** view.
2. Right-click at the very top of the form and select *Form Properties* from the menu. Be careful not to click on the first subform, which starts just one line down from the top of the form.
3. Enter the new name of the form in the Properties dialog box. The form name is the first field of the dialog and is highlighted.

If you are creating a form from scratch, you probably want to change the Window Title of the form to match the pre-built forms.

1. From the open form, bring up the **View>Design** Pane.
2. Make sure that the *Define* field is displaying the main form. The name shown will have *(Form)* appended to it.
3. Set the *Event* field to *Window Title*.
4. Change the **Run** radio button to *Formula*.
5. Enter the following formula in the *edit* field: **@If(@IsNewDoc; "New Change Request"; "Request Nr. "+ number + " " + category + " / " + current.phase)**

Subforms

Request forms are composed of the main form plus the following set of subforms:

ControlHeader

This subform is required and must be placed at the top of the form. It contains some internal data about the request itself as well as some of the needed control fields for processing**.** The company logo is also located in this subform. Text displaye in red is used to indicate hidden fields and text.

RequestHeader

This required subform provides the main header for all request forms. It includes the form header layout, more processing control fields**,** plus the request information common to all requests: *number*, *category*, *phase*, *planned start*, and *end dates*. Place this subform immediately following the ControlHeader subform.

***<phase>*.Phase**

This subform supplies the body of the information that is specific to this phase. For the prebuilt forms provided in the base install, there exists one of these subforms for each request form, where <*phase*> is the name of the request phase. For example: *Hardware.Phase* for the Hardware phase, and *HW.server.Phase* for the HW.server phase.

The information contained in these subforms need not be placed in a subform. It could be put directly into the main form, which is how the pre-built task forms are done.

ApprovalInformation

This required subform presents the information pertaining to the approval process of a request.

ClosingInformation

This is the last required subform. It contains three fields that are filled in when the request is closed.

History

This is an optional subform that tells who created the request and when it was last updated. Clicking on **Document Info** will display additional data about the document.

Note: The main request form must also define the control field *Form*, which holds the name of the Notes form to use for viewing tickets for this request phase. The *value* of this field must be set to the name of the form.

Fields

There are several important guideline that should be followed when adding fields to a form:

- All fields must be of the type *text* or *keyword* (which yields text). This includes date fields which must be declared as *text*.
- When adding date fields, make sure that the corresponding field in ServiceCenter's event map contains the necessary additional RAD code to handle differences in date formats. See *Additional Mappings for Change Management* in the Chapter 3 for detailed instructions about how this is done.
- If the field being added is new to this event type, there must be a corresponding map entry added to the SCAuto for Lotus Notes event map for this event type. See *Configuring SCAuto for Lotus Notes Event Maps* in the Chapter 3 for details.

Creating Task Forms

Forms

Task forms are stored in the main Help Desk database. The easiest way to create a new form is to start by copying an existing task form. See the previous section on creating request forms for instructions on how this is accomplished. The method is the same except that the window title for task forms should be changed to the following formula:

@If(@IsNewDoc; "New Task"; "Task Nr. "+ number + " "+category + " / " + current.phase)

Unlike the request forms included in the base installation of SCAuto for Lotus Notes, the pre-built task forms do not put the main body of the task phase-specific information in a subform. Instead, this information is placed directly into the form itself. This is not a requirement; just a design choice. The choice is up to the designer of the form.

Subforms

Task forms are composed of the main form plus the following set of subforms:

Note: The main task form must also define a *Form* field. This control field holds the name of the form, and is used to select the proper form for viewing tickets of this task phase. The *value* of this field must be set to the name of the form.

ControlHeader

This subform is required and must be placed at the top of the form. It has several fields that contain internal information about the task phase as well as some of the needed control fields for processing**.** Hidden areas are indicated with red-colored text.

TaskHeader

This required subform provides the main visible header for all task forms. It includes the form header layout, processing control fields**,** and the data common to all tasks: *number*, *category*, *phase*, *planned start*, and *planned end dates*. Place this subform immediately following the ControlHeader subform.

ApprovalInformation

This required subform holds all of the fields necessary for processing task approvals.

ClosingInformation

This is the last required subform. It contains fields that get filled in when the task is closed.

History

This is an optional subform that gives some information about who created the document, when it was created, and who can access it.

Fields

There are several important issues to note when adding fields to a form:

- All fields are of type text or keyword (which yields text). This includes date fields which must be declared as text.
- When adding date fields, make sure that the corresponding field in ServiceCenter's event map contains the necessary additional RAD code to handle differences in date formats. See *Additional Mappings for Change Management* in Chapter 3 for details.
- If the field being added is new to this event type, there must be a corresponding map entry added to the SCAuto for Lotus Notes event map for this event type. See *Configuring SCAuto for Lotus Notes Event Maps* in the Chapter 3 for details.

Chapter 7 Troubleshooting

Common Problems

If the time on the Lotus Notes server machine is out of sync with respect to the time on the ServiceCenter machine running the event scheduler, delays can occur in processing the events coming from SCAuto for Lotus Notes. In particular, ServiceCenter cannot process events if their time stamps appear to be in the future. To avoid these problems, ensure that the time on your Lotus Notes server host is the same as that on your ServiceCenter host.

If you test SCAuto for Lotus Notes on one ServiceCenter system for testing, then use it with a different ServiceCenter system, the event checkpoints are incorrect. This problem can also happen if the ServiceCenter database is reinitialized. The next section describes how to clear these checkpoints.

If the server add-in does not start (it does not show up in the Notes server's task list) and no error message is given, this may be because it cannot find the SCAuto DLL. This DLL is required on Windows systems and should be copied into the Notes server directory along with the server add-in, or appear in a directory along the system search path. The problem of no error messages appearing only occurs with some versions of Notes.

Saved Checkpoints

SCAuto for Lotus Notes saves two variables as external Lotus Notes environment variables. The location of these depends upon which system Notes is running on and is documented with your Lotus Notes system. See the ***Application Developer's Reference*** manual for details.

The two variables that are saved are checkpoints, so that SCAuto for Lotus Notes can continue where it left off whenever it is restarted. Without these variables, SCAuto for Lotus Notes would process all events from ServiceCenter (even those it had processed before) and assume that every note in the database needs to be sent to ServiceCenter. Normally you should not modify these variables, but you may wish to clear or view them for various reasons.

These variables are named **scauto_sysseq_<add-in>_<database_id>** and **scauto_srch_<add-in>_<database_id>**, where <*add-in*> is the server add-in task name, and <*database_id*> is the name of the sample problem/change database specified in the configuration note. These variables may be viewed or cleared from the Lotus Notes Server with the following commands (type them in the server window or console):

SHOW CONFIGURATION scauto_sysseq_<xxx>_<yyy>

SET CONFIGURATION "scauto_sysseq_<xxx>_<yyy>="

Error Messages

All SCAuto for Lotus Notes error messages put into the Lotus Notes server log are prefixed with the name of the server add-in for easy identification (default is *scnotes*). This is the same name that is displayed in the server console window from the **show tasks** command. The following messages, among others, may appear in this log.

scnotes: SCAuto connection to *xy.zzy* failed: <reason>

SCAuto for Lotus Notes could not contact or communicate correctly with the SCAutomate base server. Ensure that the server is running and that you have specified the correct host name and service name pair in the configuration note of the configuration database. The **`<reason>`** text will give further information about the exact problem encountered.

scnotes: Can't open database 'xyzzy'

The SCAuto for Lotus Notes Help Desk database cannot be found. Ensure that the correct filename was specified in the configuration note; include the full pathname, if necessary.

scnotes: Can't open configuration database

The SCAuto for Lotus Notes configuration database cannot be found. This database contains the configuration note along with all of the event maps. The default name for this database is *scconfig.nsf.* To specify a different name, include its name as a parameter when starting the server add-in. For example, to use the configuration database 'myconfig', start the add-in with the following server console command: **load scnotes myconfig.nsf**.

scnotes: Can not find configuration note; please check installation

No configuration note was found in the SCAuto for Lotus Notes configuration database. Ensure that such a note was composed and that the correct database was specified to the SCAuto for Lotus Notes server add-in.

scnotes: SCAuto server name not found in configuration note

The configuration note was found, but no SCAutomate base server name was found. This parameter is required and has no default value.

scnotes: Disconnected from SCAuto base server

The network connection to the *scautod* is no longer active. This may be due to a network error; the *scautod* server may have terminated; or the server machine may have rebooted. SCAuto for Lotus Notes does not terminate immediately, but tries to reconnect periodically in case the server restarts.

scnotes: Invalid Event Type: <event>

Every event must have an associated event map. This error is received whenever the map for a given event type cannot be found. An event type is the combination of the actual event keyword along with the direction the event will travel (input or output). Event maps are located in the configuration database.

scnotes: SCAuto Control Field not defined

There is a required field that must be defined in each form. This field is used for processing control as well as general filtering. The actual field name and its values are defined in the configuration note in the configuration database. The default name is *scauto_flag*.

Customer Support

Prior to notifying Peregrine Systems of problems, please check any log files for useful error messages and diagnostics. This includes the ServiceCenter log file and the SCAutomate Base log file. Check the Lotus Notes server log as well.

To isolate the problem, try running SCAuto for Lotus Notes with debugging turned on (a parameter in the configuration note). Debugging mode results in many more messages being logged to the Lotus Notes server log.

Before contacting Peregrine Systems customer support, gather the following information:

- ServiceCenter version number (for example, 1.4.0).
- SCAutomate version number.
- ServiceCenter applications release (for example, A9602).
- SCAuto for Lotus Notes release (printed in the Notes server log).
- Type of hardware and operating systems being used.
- Any error messages or diagnostics.

See the *Preface* of this user's guide for information on contacting Peregrine Systems' customer support.

Index

Peregrine SYSTEMS®

The Infrastructure Management Company™