

Peregrine

# ServiceCenter

---

# Event Services

Release 5.1

Copyright © 2002-2003 Peregrine Systems, Inc. or its subsidiaries. All rights reserved.

Information contained in this document is proprietary to Peregrine Systems, Incorporated, and may be used or disclosed only with written permission from Peregrine Systems, Inc. This book, or any part thereof, may not be reproduced without the prior written permission of Peregrine Systems, Inc. This document refers to numerous products by their trade names. In most, if not all, cases these designations are claimed as Trademarks or Registered Trademarks by their respective companies.

Peregrine Systems® and ServiceCenter® are registered trademarks of Peregrine Systems, Inc. or its subsidiaries.

This document and the related software described in this manual are supplied under license or nondisclosure agreement and may be used or copied only in accordance with the terms of the agreement. The information in this document is subject to change without notice and does not represent a commitment on the part of Peregrine Systems, Inc. Contact Peregrine Systems, Inc., Customer Support to verify the date of the latest version of this document.

The names of companies and individuals used in the sample database and in examples in the manuals are fictitious and are intended to illustrate the use of the software. Any resemblance to actual companies or individuals, whether past or present, is purely coincidental.

If you need technical support for this product, or would like to request documentation for a product for which you are licensed, contact Peregrine Systems, Inc. Customer Support by email at [support@peregrine.com](mailto:support@peregrine.com).

If you have comments or suggestions about this documentation, contact Peregrine Systems, Inc. Technical Publications by email at [doc\\_comments@peregrine.com](mailto:doc_comments@peregrine.com).

This edition applies to version 5.1 of the licensed program.

Peregrine Systems, Inc.  
Worldwide Corporate Headquarters  
3611 Valley Centre Drive San Diego, CA 92130  
Tel 800.638.5231 or 858.481.5000  
Fax 858.481.1751  
[www.peregrine.com](http://www.peregrine.com)



# Contents

---

	<b>About This Guide . . . . .</b>	<b>11</b>
	Knowledge Requirements . . . . .	12
	Sample Forms and Examples . . . . .	12
	Documentation Web Site. . . . .	12
	Contacting Peregrine Systems . . . . .	13
	Peregrine’s CenterPoint Web Site . . . . .	13
	Contacting Education Services . . . . .	13
<b>Chapter 1</b>	<b>Introduction . . . . .</b>	<b>15</b>
	Overview . . . . .	15
	How Event Services Works . . . . .	17
	Components . . . . .	18
	Event Services Files . . . . .	19
	Event Services Flowchart . . . . .	20
	Accessing Event Services . . . . .	21
	GUI Client. . . . .	21
	Text Client. . . . .	26
<b>Chapter 2</b>	<b>Standard Event Operations . . . . .</b>	<b>29</b>
	Input Events . . . . .	30
	Input Fields . . . . .	31
	Input Event Processing . . . . .	36
	Output Events . . . . .	37
	Output Fields . . . . .	38

Event Registration . . . . .	40
Reviewing Event Registration . . . . .	40
Event Registration Fields . . . . .	41
Commonly-Used Events . . . . .	47
Service Management Events . . . . .	47
Incident Management Events . . . . .	47
Inventory Management Events . . . . .	47
Change Management Events . . . . .	48
Request Management Events . . . . .	48
Service Level Management Events . . . . .	49
Standard Events . . . . .	49
CTSCPY (2) . . . . .	49
CTSCPY (1) . . . . .	50
CTSIMP (2) . . . . .	50
CTSIMP (1) . . . . .	51
CTSIMP2 . . . . .	52
CTSRQCLS (2) . . . . .	53
CTSRQCLS (1) . . . . .	54
CTSRQOPN (2) . . . . .	54
CTSRQOPN (3) . . . . .	55
CTSRQOPN (1) . . . . .	56
CTSRQUPD (2) . . . . .	57
CTSRQUPD (1) . . . . .	58
CTSTKCLS (2) . . . . .	58
CTSTKCLS (1) . . . . .	59
CTSTKOPN (2) . . . . .	60
CTSTKOPN (3) . . . . .	61
CTSTKOPN (1) . . . . .	62
CTSTKUPD (2) . . . . .	62
CTSTKUPD (1) . . . . .	63
ERPHR (1) . . . . .	64
ERPHR (2) . . . . .	64
ERPSTATES (1) . . . . .	65
ERPSTATES (2) . . . . .	65
ICMApplication . . . . .	66

ICMcomputer . . . . .	66
ICMdevice. . . . .	67
ICMdisplaydevice. . . . .	68
ICMexample . . . . .	68
ICMfurnishings . . . . .	68
ICMhandhelds . . . . .	68
ICMmainframe. . . . .	69
ICMnetworkcomponents . . . . .	69
ICMofficeelectronics . . . . .	69
ICMsoftwarelicense . . . . .	69
ICMstorage . . . . .	70
ICMtelecom . . . . .	70
GetResRM. . . . .	70
GetResRM. . . . .	71
GetResRML . . . . .	71
GetResRML . . . . .	71
HotNews . . . . .	72
IND . . . . .	72
NDpmc . . . . .	73
NDpmc . . . . .	73
NDpmo . . . . .	74
NDpmo . . . . .	74
PSSDELETE . . . . .	75
SALESQUOTE . . . . .	75
SAPGRT . . . . .	75
SAPGRT . . . . .	76
SAPGTE. . . . .	76
SAPHR (1). . . . .	77
SAPHR (2). . . . .	78
SAPHRMD . . . . .	78
SAPORD . . . . .	79
SAPORD . . . . .	79
SAPORDQ. . . . .	80
SAPQTE. . . . .	80
SAPQTE. . . . .	81

SAPQTEQ . . . . .	81
SAPREQ. . . . .	82
SAPREQ. . . . .	82
SAPREQO . . . . .	82
ScAcBrand. . . . .	83
ScAcCompany . . . . .	84
ScAcContacts . . . . .	85
ScAcDept . . . . .	86
ScAcDevice . . . . .	87
ScAcLocation . . . . .	88
ScAcModel . . . . .	89
ScAcModelBundle . . . . .	90
ScAcModelVendor . . . . .	91
ScAcVendor . . . . .	92
ScAcVendorBACK . . . . .	93
ScFcOrderLine . . . . .	94
ScFcOrderLine . . . . .	94
TcScCompDel . . . . .	95
TcScCompany . . . . .	95
TcScContacts . . . . .	96
TcScDept . . . . .	97
TcScDeptDel. . . . .	98
TcScDeptdel . . . . .	99
TcScLocation. . . . .	99
TcScLocation. . . . .	100
WMI . . . . .	101
WMI . . . . .	102
XIND . . . . .	103
approval. . . . .	103
approval. . . . .	104
cm3rin . . . . .	104
cm3rinac . . . . .	105
cm3rout. . . . .	105
cm3tin . . . . .	106
cm3tinac . . . . .	106

cm3tout . . . . .	107
dbadd . . . . .	107
dbdel . . . . .	108
dbupd . . . . .	109
email . . . . .	110
email . . . . .	110
epmc . . . . .	111
epmc . . . . .	112
epmo . . . . .	112
epmo . . . . .	113
epmosmu . . . . .	114
epmosmu . . . . .	115
epmu . . . . .	115
epmu . . . . .	116
esmin . . . . .	117
esmin . . . . .	117
gie . . . . .	118
icma . . . . .	119
icmd . . . . .	120
icmswa . . . . .	121
icmswd . . . . .	121
icmu . . . . .	122
mlbcm3tc . . . . .	123
mlbcm3tu . . . . .	124
mlbocmlc . . . . .	125
mlbocmlu . . . . .	126
mlblpmc . . . . .	127
mlblpmo . . . . .	128
mlblpmu . . . . .	129
opera . . . . .	130
operd . . . . .	131
operu . . . . .	132
outageend . . . . .	133
outagestart . . . . .	133
page . . . . .	133

pageclose . . . . .	134
pageresp. . . . .	135
pcsoftware. . . . .	136
pmc . . . . .	137
pmc . . . . .	138
pmo . . . . .	139
pmo . . . . .	140
pmu . . . . .	141
pmu . . . . .	142
prgma. . . . .	142
prgmd . . . . .	143
prgmu . . . . .	144
rmlin . . . . .	145
rmoappr. . . . .	145
rmoin. . . . .	146
rmqappr. . . . .	146
rmqin. . . . .	147
sapordl (1). . . . .	147
sapordl (2). . . . .	147
sapqtel (1). . . . .	148
sapqtel (2). . . . .	148
saprecl (1). . . . .	149
sapreql (1). . . . .	149
sapreql (2). . . . .	149
slaresponse . . . . .	150
smin . . . . .	150
smout. . . . .	151
submit . . . . .	151
sysbull . . . . .	152
Global Variables . . . . .	153
Generic Event Administration. . . . .	154



<b>Chapter 3</b>	<b>Mapping and Filtering . . . . .</b>	<b>157</b>
	Mapping. . . . .	158
	The Event Map Form . . . . .	158
	Using Event Maps. . . . .	165
	Global Variables . . . . .	171
	Mapping Considerations for Inventory Management . . . . .	172
	Building a New Event Map . . . . .	173
	Event Filters . . . . .	180
	Fields . . . . .	181
	Blocking. . . . .	183
<b>Chapter 4</b>	<b>ServiceCenter/Network Discovery Integration . . . . .</b>	<b>189</b>
	How Network Discovery and ServiceCenter Work Together . . . . .	190
	What Does Network Discovery Provide to ServiceCenter?. . . . .	190
	How Does Network Discovery Pass Information to ServiceCenter? . . . . .	191
	Event Services Mapping for Network Discovery Device Information . . . . .	194
	Opening and Closing Incident Tickets . . . . .	194
	Event Services Mapping for Network Discovery-Detected Problems . . . . .	195
<b>Chapter 5</b>	<b>Change Management Event Services . . . . .</b>	<b>197</b>
	Input Events . . . . .	198
	Input Event Registrations . . . . .	198
	Setting Up the External Information String . . . . .	199
	Keeping ServiceCenter In-Synch with an External System . . . . .	201
	Acknowledgments . . . . .	201
	Sending Complete Output Events . . . . .	202
	Change Event Examples . . . . .	203
	Input Examples. . . . .	203
	Output Examples . . . . .	204

<b>Chapter 6</b>	<b>Event Agent Operations . . . . .</b>	<b>205</b>
	Event Scheduling . . . . .	206
	Reviewing Scheduled Events . . . . .	206
	OS/390 (MVS)/SCAuto Agents . . . . .	209
	Maintaining Agent Status. . . . .	210
	System Startup . . . . .	210
	System Status Window . . . . .	211
	Event Agent Check . . . . .	212
	The VSAM Information Record . . . . .	214
	Reviewing the vsaminfo record . . . . .	214
	The NAPA Information Record . . . . .	216
	Reviewing the napainfo record . . . . .	216
<b>Chapter 7</b>	<b>SCemail . . . . .</b>	<b>219</b>
	Email Events . . . . .	220
	SCemail vs. SCAutoMail . . . . .	220
	Sending ServiceCenter Mail to email . . . . .	220
	Changes to Existing ServiceCenter Mail Utility. . . . .	223
	SCemail . . . . .	224
	Windows NT. . . . .	224
	Starting SCemail . . . . .	226
	Optional Parameters . . . . .	227
	Unix . . . . .	228
	OS/390 . . . . .	229
	Sending email . . . . .	231
	Using Format Control . . . . .	231
	From Incident Management . . . . .	232

<b>Chapter 8</b>	<b>Format Control Options . . . . .</b>	<b>235</b>
	Generating eventout Records . . . . .	236
	Format Control. . . . .	236
	Incident Management . . . . .	237
	Inventory and Configuration Management . . . . .	239
	Generating Page Messages . . . . .	240
	Format Control. . . . .	240
	Incident Management . . . . .	242
	Sending Fax Messages . . . . .	244
	Format Control. . . . .	244
	Creating Output Events . . . . .	245
	Format Control. . . . .	245
<b>Appendix A</b>	<b>Basic Troubleshooting . . . . .</b>	<b>247</b>
	Frequently Asked Questions . . . . .	247
	<b>Index . . . . .</b>	<b>255</b>



# About This Guide

---

Increasingly, enterprise-wide network management tools depend on automation to detect activity on the network and to execute the appropriate procedures. These network incidents are often called alarms or alerts; ServiceCenter refers to them as *events*.

This manual introduces *Event Services* and explains:

- *Standard Event Operations* on page 29
- *Mapping and Filtering* on page 157
- *ServiceCenter/Network Discovery Integration* on page 189
- *Change Management Event Services* on page 197
- *Event Agent Operations* on page 205
- *SCemail* on page 219
- *Format Control Options* on page 235
- *Basic Troubleshooting* on page 247

## Knowledge Requirements

Readers of this guide need general knowledge of the following:

- How ServiceCenter works
- How the underlying database functions, and
- How the interface passes data into ServiceCenter.

Before you begin using this guide, become familiar with topics in ServiceCenter and third-party documentation as follows:

- For a working knowledge of ServiceCenter, see the *System Administrator's Guide* and the *System Tailoring* guides.
- For an understanding of how to use Database Manager and view data in records, see the *Database Management and Administration* guide.
- To become familiar with ServiceCenter interfaces, see the *System Administrator's Guide* and the *User's Guide*; for an understanding of external interfaces, see the appropriate product documentation.

## Sample Forms and Examples

The sample forms and examples included in this guide are for illustration only, and may differ from those at your site.

## Documentation Web Site

For a complete listing of the current ServiceCenter documentation, see the Documentation pages on the Peregrine CenterPoint Web site at <http://support.peregrine.com/>.

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

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

## Contacting Peregrine Systems

For further information and assistance with ServiceCenter in general, contact Peregrine's Customer Support.

### Peregrine's CenterPoint Web Site

Current details of local support offices are available through Peregrine's CenterPoint Web site at <http://support.peregrine.com/>.

**To find Peregrine Worldwide Contact Information:**

- 1 Log on with your login User Name and Password.
- 2 Click **Go for CenterPoint**.
- 3 Select **Whom Do I Call?** in the navigation bar on the left side of the page. Peregrine worldwide information displays for all products.

## Contacting Education Services

Training services are available for the full spectrum of Peregrine Products including ServiceCenter.

Current details of our training services are available through the following main contacts or at:

<http://www.peregrine.com/education>

Address:	Peregrine Systems, Inc. Attn: Education Services 3611 Valley Centre Drive San Diego, CA 92130
Telephone:	+1 (858) 794-5009
Fax:	+1 (858) 480-3928





# 1 Introduction

## CHAPTER

### Overview

Network management consists of *events* and *procedures*. The events include failed applications, new workstation installations, broken communications equipment and inaccessible facilities. The procedures include Incident, Change, Inventory and Configuration Management.

The ServiceCenter Event Services connects ServiceCenter to external systems that detect network incidents. This allows ServiceCenter to receive events generated from outside and to send information back to these external systems. Event Services performs background processing only and cannot be used for interactive work.

ServiceCenter Event Services forms the basis for event processing within the procedures of Network Management. In concert with SCAutomate, NetView Automated Problem Applications (NAPA), SCAuto for NetView OS/390 (which replaced NAPA), Get.It! and Connect.It, Event Services responds to events detected by external systems and maintains inventory items; opens, updates, and closes incidents; and generates requests. It creates external events, e.g., email, based on procedural instructions. And it provides a generic *cut-through* interface that allows users of external systems to become ServiceCenter users from their native environment.

ServiceCenter Event Services provides standard applications to open, update and close incidents, and to add, update and delete inventory items. It also provides a standard email interface, which can be used with most email systems. Because the applications are standard, data is expected in a defined and constant format for input events, and is provided in a defined and constant format to output events.

---

**Important:** The following material is organized upon the assumption the reader is already familiar with the ServiceCenter RAD environment. Refer to the *ServiceCenter RAD Guide* for further information.

---

Event Services creates and manages the environments of, and is a requirement for the following ServiceCenter products:

- SCAutomate
- NAPA
- SCAuto for NetView OS/390

## How Event Services Works

Events entering and exiting ServiceCenter are routed differently depending upon the external system with which ServiceCenter is communicating. For some products, information is routed in one direction only. For others, events flow in both directions through *listeners*. The following table shows the routing of events through external products currently supported:

File	Description
NAPA	Inbound events only. Information is routed from <b>vsam</b> (outside of ServiceCenter). An application within ServiceCenter, called <b>vsam.read</b> , reads <b>vsam</b> and writes an <b>eventin</b> record.
SCEmail	Outbound events only. Information is routed from ServiceCenter to SCEmail by an <b>eventout</b> record.
Connect.It	Inbound and outbound events. Information is routed in both directions through a listener ( <i>scenter</i> ).
Get.It!	Inbound and outbound events. Information is routed in both directions through a listener ( <i>scenter</i> ).
SCMail SCPager SCMapi SCAuto products	Inbound and outbound events. Information is routed in both directions through a listener ( <i>scenter</i> ).

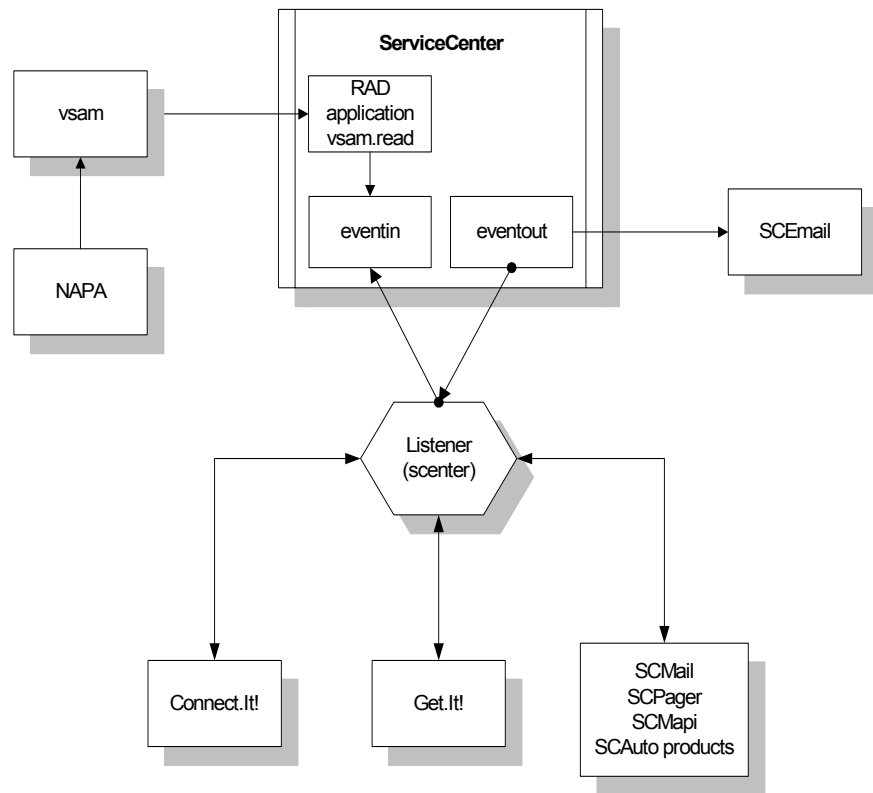


Figure 1-1: Flow of events from external sources in and out of ServiceCenter

## Components

Event Services comprises seven major application areas:

- A generic Registration/login application.
- A Review of Input records and scheduler, for events detected from external sources.
- A Review of Output records and scheduler, for routing ServiceCenter events to external sources.
- A Mapping function for fields in ServiceCenter files to positions in data sent from external sources.
- A Filter designed to eliminate events based on user-specified criteria,

- Generic Applications to process incidents and inventory, interface with paging devices, and send email.
- ERP Interface administration, allowing the connection and interaction with disparate external systems, for example, SAP and PeopleSoft.

The application components depend on external software programs to interface with the ServiceCenter input and output queues. For more information about these programs, consult the SCAuto for NetView OS/390 manual.

SCAuto for NetView OS/390 (NAPA) events are called *standard events*. These events provide and depend upon standard information in specific positions. The default mapping records provided with Event Services define standard events.

## Event Services Files

There are five event files in ServiceCenter:

File	Description
eventregister	Defines the events that exist in the system. Event registration records also specify the event maps used to process events and defines the RAD application to be used for processing.
eventin	File used to move information into ServiceCenter from an external system. If a corresponding <i>input</i> eventregister record exists, external or internal applications can write records to the eventin file.
eventout	File used to move information from ServiceCenter into an external system. A particular type of an eventout record can be written only if a corresponding <i>output</i> eventregister record exists.
eventmap	Defines how information should be parsed. Event maps define individual fields and create condition statements for eventin and eventout records. Many eventmap records can exist for each eventregistration record.
eventfilter	Prevents duplicate events. Filters block incoming events based on defined criteria to prevent external systems from creating many eventin records for the same item in a short amount of time. Filters can block events by time frame, item, or location.

# Event Services Flowchart

This flowchart depicts a macro view of ServiceCenter Event Services.

Event Services (ES) uses its own scheduler called *event*. The event scheduler is started and stopped just like any other ServiceCenter scheduler and is used to process events in background or asynchronously.

The event registration file contains all of the information ES needs to determine what to do with each event.

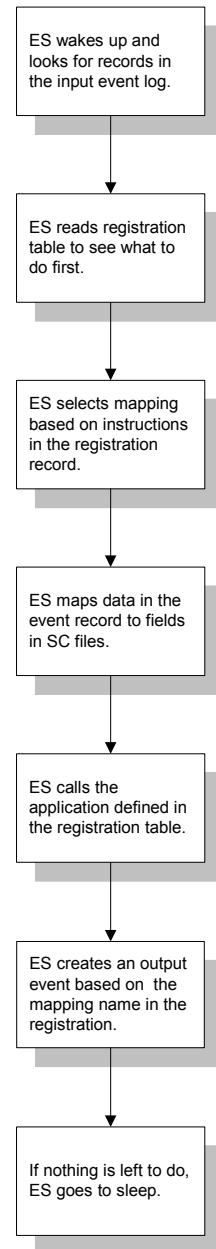
Mapping records contain instructions to move data from the eventin record to fields in ServiceCenter files.

Based on instructions in the mapping records, a data structure is built.

A multi-purpose call routine is issued to the application named in the registration record, along with any necessary variables.

When the application has completed, an output event is created and added to the queue, if instructed by the registration.

If Event Services has nothing left to do, it sleeps for an interval, then reawakens to look for more work.



# Accessing Event Services

## GUI Client

To access Event Services

- 1 Log into ServiceCenter.

You must be a ServiceCenter system administrator to work in Event Services.

- 2 Select the Utilities tab on the ServiceCenter main menu.

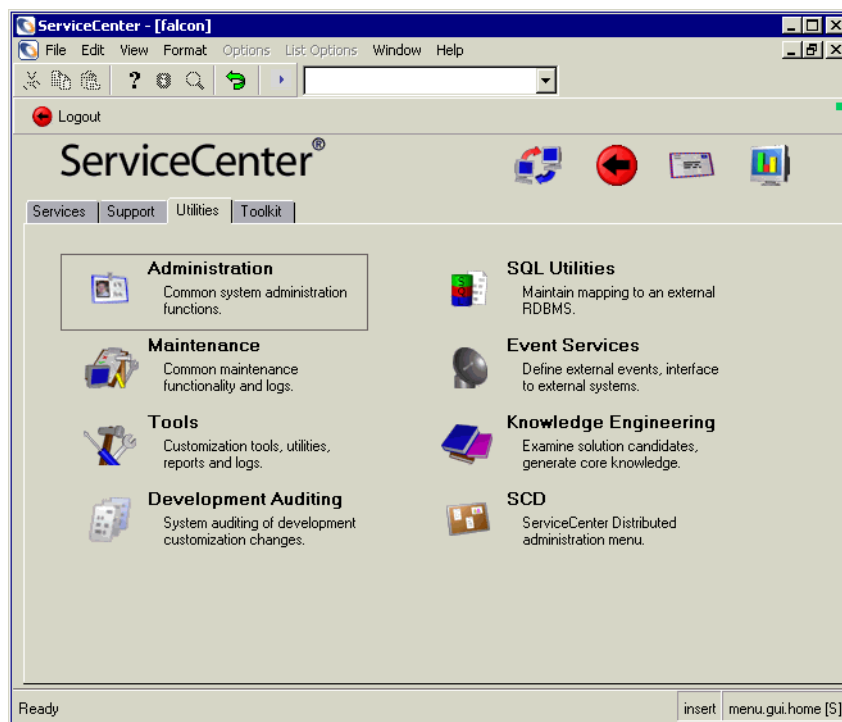
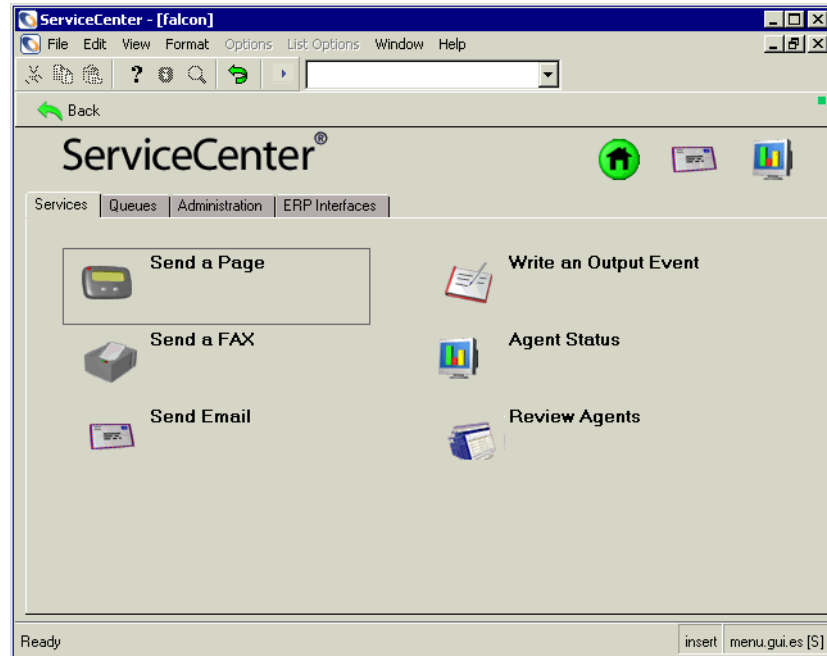


Figure 1-2: Utilities tab in the system administrator's home menu

- 3 Click Event Services.

The Event Services menu is displayed. This menu controls all of the applications, parameters and filters used by SCAuto for NetView OS/390 and ServiceCenter Automate (SCAuto).



**Figure 1-3: Event Services Main Menu**

The GUI menu consists of four tabs:

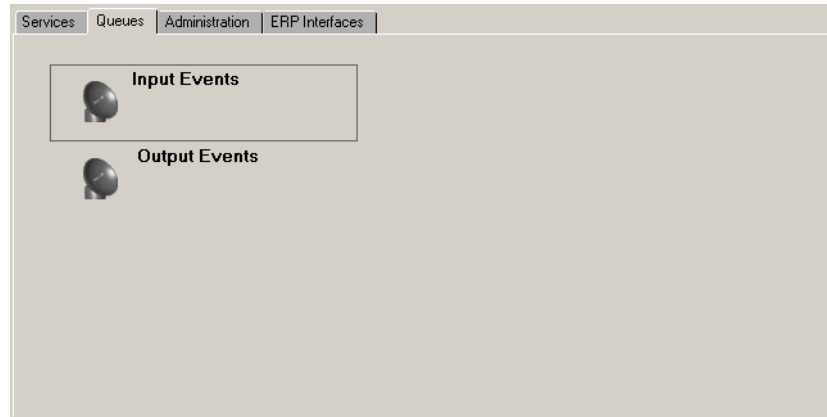
- Services
- Queues
- Administration
- ERP Interfaces.



## Services Tab

Button	Action
Send a Page	Initiates a paging event and opens the page transmission form, <b>pager.info.g</b> .
Send a Fax	Initiates the fax event and opens the fax transmission form, <b>send.fax.g</b> .
Send Email	Initiates the email event and opens the email transmission form, <b>send.email.g</b> .
Write an Output Event	Initiates event create script, prompting you for selection of type of external event: Incident, Inventory or Generic (message).
Agent Status	Displays a status list of all SCAutomate agents, including last expiration and idle time, and provides Start and Stop controls for each agent.
Review Agents	Opens the Event Scheduler screen, displaying details for scheduled system events.

## Queues Tab



### Button

### Action

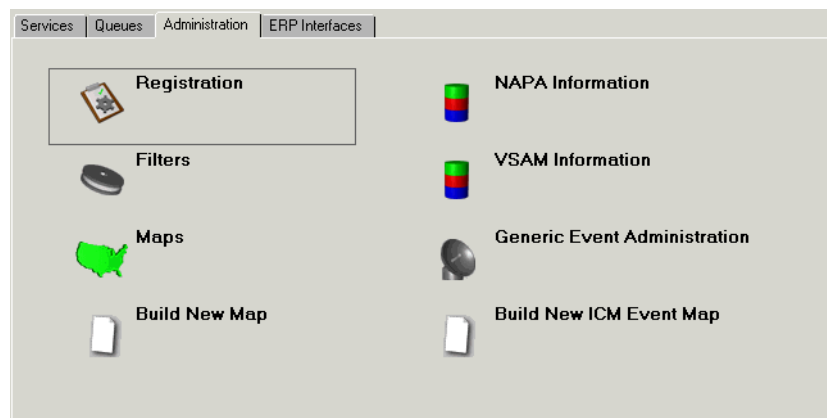
Input Events

Opens the eventin file for review. This file contains all events awaiting action by ServiceCenter and those that have been processed but not deleted.

Output Events

Opens the eventout file for review. This file contains all ServiceCenter events awaiting action by an external application and those that have been processed but not deleted.

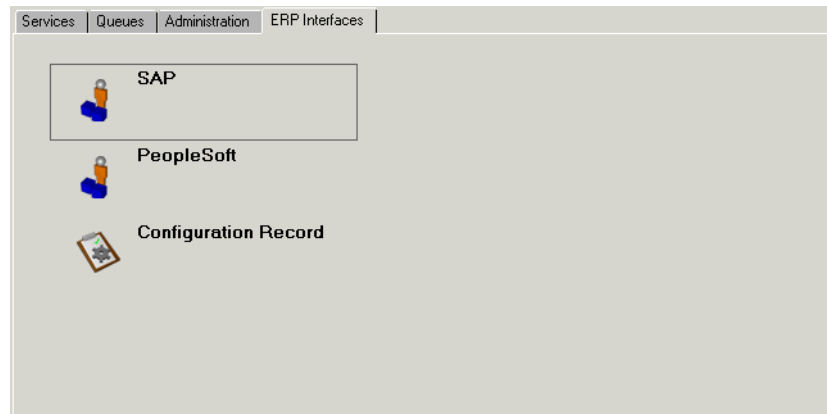
## Administration Tab



Button	Action
Registration	Accesses Event Services registration records. Each registration record provides the information ServiceCenter requires to process and event type.
Filters	Allows maintenance of event filters. Although general in scope so that filters can be used for any purpose, the primary focus is on incident filtering.
Maps	Allows maintenance of existing event maps. Event maps define the relationship between data passed into and out of ServiceCenter in flat, delimited form, and fields in ServiceCenter files.
Build New Map	Accesses an application which helps to quickly define a new event map for a ServiceCenter file.
NAPA Information	Allows maintenance of the NAPA schedulers.
VSAM Information	Allows maintenance of the <b>vsaminfo</b> file.
Generic Event Administration	Opens the Generic Events Menu of controls, which include editing of GOE configuration records, mass exporting of records, and exporting of database dictionary structures.
Build New ICM Event Map	Begins the new inventory script, beginning by prompting you for the ICM file upon which to build the event map.

## ERP Interfaces Tab

Button	Action
SAP	Opens the SAP R/3 Interface menu for establishing an exchange of SAP HR contact and support records.
PeopleSoft	Opens the PeopleSoft Interface menu for establishing an exchange of PeopleSoft contact and support records.
Configuration Record	Accesses the SAP HR configuration file, where the HR and Materials Management (MM) interfaces are specified, and default server names are identified.



## Text Client

Access the main Event Services menus with the following commands:

- Administration Menu: **am**
- Event Services Menu: **es**
- Event Services Admin Menu: **esadmin**

The text version of ServiceCenter splits the GUI options discussed in the previous section into two menus:

## Event Services

- Review Event Input Log
- Review Event Output Log
- Administration Menu
- Send a Page
- Send a FAX
- Send an Email Message
- Write an Output Event
- Event Services Admin Menu
- ERP Interfaces

## Event Services Administration

- Maintain Event Register
- Maintain Event Mapping
- Event Services Menu
- Maintain Event Filters
- Build New Event Map
- Maintain VSAM Information
- Maintain SCAuto for NetView OS/390 (NAPA) Information
- Maintain Agent Status
- Review Agents
- Build New ICM Event Map



# 2 Standard Event Operations

## CHAPTER

This chapter addresses the primary operations of the Event Services module of ServiceCenter. Events take many forms and occur at various times throughout the operation of the system.

Events are, for the purposes of this chapter, divided into the following sections:

- *Input Events* on page 30
- *Output Events* on page 37
- *Event Registration* on page 40
- *Standard Events* on page 49
- *Generic Event Administration* on page 154

# Input Events

The input event log file is called `eventin`. It contains a record for every event detected but not filtered by SCAuto external applications. The record must contain the event code, a unique system ID and a time stamp. Data is passed to ServiceCenter in a character string using a delimiter character to separate fields.

## To review input events:

- 1 Access **Event Services**. For instructions, refer to *Accessing Event Services* on page 21.
- 2 Select the **Queues** tab.
- 3 Click **Input Events**.  
The `event.in.g` form is displayed.
- 4 Click **Search** to display a QBE list of all input events.



- 5 Double-click on an event to display the record.

The screenshot shows a Windows-style application window titled "ServiceCenter - [Search eventin Records]". The window has a menu bar with "File", "Edit", "View", "Format", "Options", "List Options", "Window", and "Help". Below the menu bar is a toolbar with icons for Back, Add, Search, Find, and Fill. The main content area is titled "Event Services Input Queue" and contains several sections of input fields:

- Event Code:** Fields for "email" (value: jemail), "Status" (value: mailed), and "System Sequence" (value: 36547ef1edf1c0).
- Time Stamps:** Fields for "First Expiration" (empty) and "Time Processed" (value: 04/23/03 10:08:20).
- User Information:** Fields for "User Name" (empty), "Password" (empty), and "User Sequence" (empty).
- Incident Information:** Fields for "Network Name" (empty), "Cause Code" (empty), and "Incident ID" (empty).
- Filter Information:** Fields for "Count" (empty) and "Next Expiration" (empty).
- System Option:** Field for "System Option" (empty).
- Field Separation Character:** Field with value: ^.
- External Information String:** Field with value: problem^SCAuto Mail^Unable to deliver mail^The following mail could not be delivered to 'hibian'.

At the bottom of the window, there is a status bar with "Ready" on the left and "insert event.in.g [S]" on the right.

Figure 2-1: Input Queue form

## Input Fields

The event fields found in the Input Events form and their corresponding properties are listed below. The encoded field names, as recorded in the eventin file, are included for reference only.

## Header

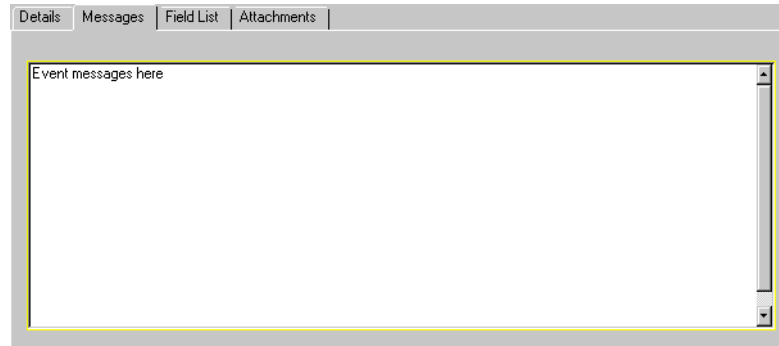
Field	Description																						
Event Code ( <i>evtype</i> )	The registration name for the event (mandatory).																						
Status ( <i>evstatus</i> )	<p>The result of the action performed by the Event Manager. If events are not deleted after processing, ServiceCenter automatically assigns one of the following statuses to each:</p> <table border="1"> <thead> <tr> <th>Status</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>added</td> <td>An inventory item has been added to the database; the device's name is in the <b>Network Name</b> field.</td> </tr> <tr> <td>closed</td> <td>An incident has been closed; the Incident Number is in the <b>Incident ID</b> field.</td> </tr> <tr> <td>deleted</td> <td>An inventory item has been marked for deletion in the database; the device's name is in the <b>Network Name</b> field.</td> </tr> <tr> <td>error</td> <td>An error occurred while processing the event. This status is assigned by</td> </tr> <tr> <td>locked</td> <td>The record to be updated or deleted was locked.</td> </tr> <tr> <td>filtered</td> <td>An incident was filtered, and is waiting for the filter condition to be satisfied.</td> </tr> <tr> <td>mailed</td> <td>Electronic mail has been sent.</td> </tr> <tr> <td>opened</td> <td>An incident has been opened; the Incident Number is in the <b>Incident ID</b> field.</td> </tr> <tr> <td>processed</td> <td>A software inventory item or change has been successfully processed.</td> </tr> <tr> <td>updated</td> <td>An incident has been updated; the Incident Number is in the <b>Incident ID</b> field.</td> </tr> </tbody> </table>	Status	Description	added	An inventory item has been added to the database; the device's name is in the <b>Network Name</b> field.	closed	An incident has been closed; the Incident Number is in the <b>Incident ID</b> field.	deleted	An inventory item has been marked for deletion in the database; the device's name is in the <b>Network Name</b> field.	error	An error occurred while processing the event. This status is assigned by	locked	The record to be updated or deleted was locked.	filtered	An incident was filtered, and is waiting for the filter condition to be satisfied.	mailed	Electronic mail has been sent.	opened	An incident has been opened; the Incident Number is in the <b>Incident ID</b> field.	processed	A software inventory item or change has been successfully processed.	updated	An incident has been updated; the Incident Number is in the <b>Incident ID</b> field.
Status	Description																						
added	An inventory item has been added to the database; the device's name is in the <b>Network Name</b> field.																						
closed	An incident has been closed; the Incident Number is in the <b>Incident ID</b> field.																						
deleted	An inventory item has been marked for deletion in the database; the device's name is in the <b>Network Name</b> field.																						
error	An error occurred while processing the event. This status is assigned by																						
locked	The record to be updated or deleted was locked.																						
filtered	An incident was filtered, and is waiting for the filter condition to be satisfied.																						
mailed	Electronic mail has been sent.																						
opened	An incident has been opened; the Incident Number is in the <b>Incident ID</b> field.																						
processed	A software inventory item or change has been successfully processed.																						
updated	An incident has been updated; the Incident Number is in the <b>Incident ID</b> field.																						
System Sequence ( <i>evsysseq</i> )	System-assigned sequence number, for event tracking (system provided).																						
First Expiration ( <i>evtime</i> )	The time the event occurred (mandatory).																						
Time Processed ( <i>evtimestamp</i> )	The system time translation of the actual time the event was processed by ServiceCenter.																						

## Details Tab

Field	Description
User Name( <i>evuser</i> )	The event user name; if passed, it is used as the operator name (optional).
Password ( <i>evpswd</i> )	The event user's password (optional).
User Sequence ( <i>evusrseq</i> )	User-assigned sequence number, used to trace an event through the ServiceCenter system (for example, an external reference number; optional).
Network Name ( <i>evnetnm</i> )	Used in filtering, the unique network name of a device (system defined by Event Services).
Cause Code ( <i>evcode</i> )	Used in filtering, an event code sent to Event Manager (system defined by Event Services).
Incident ID ( <i>evid</i> )	Problem character ID; used in filtering (system defined by Event Services).
Count ( <i>evcount</i> )	Used in filtering, the number of events for a particular transaction (system defined by Event Services).
Next Expiration ( <i>evexpire</i> )	Used in filtering, the time when an incident should be opened (system assigned by Event Services).
System Option ( <i>evsysopt</i> )	Code to identify system options (optional).
Field Separation Character ( <i>evsepchar</i> )	Character used to separate fields in the <i>evfields</i> field (substitutes ^ if null).

Field	Description
External Information String ( <i>evfields</i> )	<p>Data describing the event, with fields separated by the <i>evsepar</i> character; specific positions in the <i>evfields</i> field are reserved for application dependent data.</p> <p>For example:</p> <pre>falcon^max@peregrine.com^falcon;su sie:root^Re:meeting this afternoon^Tuesday, 23 October 2001 16:41:07</pre> <p>An email will be sent to <i>falcon</i> by <i>max@peregrine</i>, with carbon copies to <i>falcon</i>, <i>susie</i> and <i>root</i>. The subject is <i>Meeting this afternoon</i>, and the text follows the subject.</p> <p><b>Note:</b> The first line of text always includes the date and time the message was sent. Each of the data fields is separated by a separation character, or delimiter, that is defined in the registration file. If no delimiter is defined, ^ is used by default.</p>

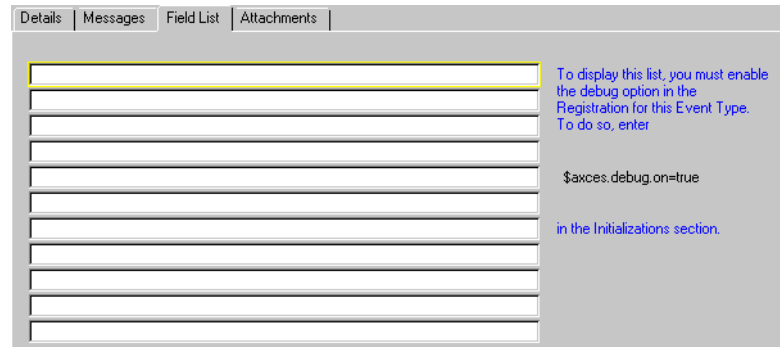
## Messages Tab



Field	Description
Messages ( <i>evmsg</i> )	Any messages generated during event processing.

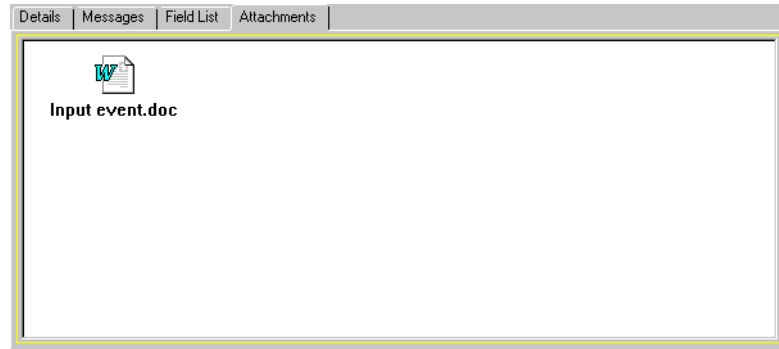
## Field List Tab

Field	Description
Field List ( <i>evlist</i> )	Array, built by the Event Manager, of the fields in the <i>evfield</i> field; available to <code>eventmap</code> as <code>\$axces.fields</code> .

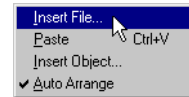


**Note:** The *evlist* field is cleaned up in the application after use. If you need to view it for debugging or trace purposes, you must set `$axces.debug=true` in your event registration initialization expressions. Should you do so, the maximum size of the *evfields* data is 16,000 bytes. Remember that email messages are often quite large, so use this feature with discretion.

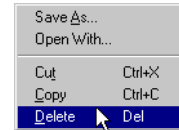
## Attachments Tab



The Attachments tab is an OLE container that allows you to insert various objects related to the Input record. To insert files as Word a document, Excel spreadsheet, or bitmap image, right-click the mouse and select the appropriate command from the pop-up menu.



To perform maintenance tasks on an object in the tab, select the object and right click the mouse. Select the appropriate command from the pop-up menu.



## Input Event Processing

All records in the eventin file are added by an external application, such as SCAuto/SDK or SCAuto for NetView OS/390, and manipulated according to its program.

For example, SCAuto supports an event called *email*. Electronic mail can be received from external sources and passed to ServiceCenter mail. The sources for electronic mail can be external email systems, alert monitors or other programs that can send messages. The external SCAuto application packages the data in a standard format and adds it to the eventin file. The format is defined in eventmap records.

**Note:** Records in the eventin file that have been processed will not contain a First Expiration value. (upper right field).

Normally, events are deleted after they have been processed unless they have been filtered or an exception has occurred during processing. The delete flag is controlled by a condition set in the `eventregister` file.

If an error occurs due to Format Control processing, event processing is terminated for that event and the specific error message is written to the eventin's Messages and to ServiceCenter `msglog` file.

**Note:** Once you have installed and tested SCAuto you should either set all delete flags in the registration records to true or use the ServiceCenter **purge/archive** routines to schedule cleanup of the file on a regular basis. Refer to the *System Administration Guide* for detailed information on the **Purge/Archive** Utility.

## Output Events

The output event log is called *eventout*. It may contain a record for each event processed by Event Services applications and instructions to be used by external software (for example, pager numbers to notify service technicians). Data is passed to external applications in a character string using a delimiter character to separate fields.

### To review output events:

- 1 Access the **Event Services** menu (see *Accessing Event Services* on page 21)
- 2 Select the Queues tab.
- 3 Click **Output Events**.  
The Output Queue form is displayed
- 4 Click Search to display a QBE list of current output events.

## 5 Double-click an event to display the record.

Figure 2-2: Output Event Queue form

## Output Fields

The encoded file names, as referenced by the eventout file, have been included for reference only.

Field	Description
Event Code ( <i>evtype</i> )	Registration name for the event (required)
Status ( <i>evstatus</i> )	Result of the action performed by the Event Manager; may be opened, updated, closed, added, deleted, filtered or error.
Event Time ( <i>evtime</i> )	Time the event occurred.
Expiration Time ( <i>evexpire</i> )	Expiration time for an event. The time when the eventout record will be processed by the VSAM scheduler; if the field is NULL, the record has been processed.



Field	Description
User Name ( <i>evuser</i> )	Event user name (optional).
Password ( <i>evpswd</i> )	Event user's password (optional).
User Sequence ( <i>evusrseq</i> )	User-defined sequence number for event tracking.
System Sequence ( <i>evsysseq</i> )	System-assigned sequence number, for event tracking (system provided); this number is used when external software is initiated to restart the eventout monitoring pointer.
System Option ( <i>evsysopt</i> )	Code to identify system options (optional).
Incident ID ( <i>evvid</i> )	Problem character ID (incident number)
Field Separator Character ( <i>evsepchar</i> )	Character used to separate fields in the <i>evfields</i> field (substitutes ^ if null).
External Information String ( <i>evfields</i> )	Data describing the event, with fields separated by the <i>evsepchar</i> character. In Figure 2-2 on page 38, ServiceCenter user <i>falcon</i> is sending electronic mail to external user <i>ruth.peters@peregrine.com</i> via email.

Records in the *eventout* file which have been processed do not contain a date of expiration. Normally, events are deleted from the *eventout* file after processing unless an error has occurred. A flag in the external IPAS or SCAuto software can be manipulated to cause record deletion after read; however, since multiple SCAuto processes can read the same record, it is not always feasible to delete on read.

**Note:** Use the ServiceCenter *purge/archive* routines to schedule cleanup of the *eventout* file on a regular basis. Refer to the *System Administration Guide* for detailed information on the *Purge/Archive* Utility.

Manipulation of *eventout* records is done according to the program interfaced.

# Event Registration

All events are registered in the *eventregister* file. The eventregister file includes a unique event code as well as a sequence number, so a single event can execute a series of applications. In addition, it contains initialization statements, mapping information and instructions for calling the ServiceCenter application.

## Reviewing Event Registration

To review event registration:

- 1 Access the Event Services menu. (See *Accessing Event Services* on page 21.)
- 2 Select the Administration tab.
- 3 Click **Registration**.

The Event Registration log is displayed.

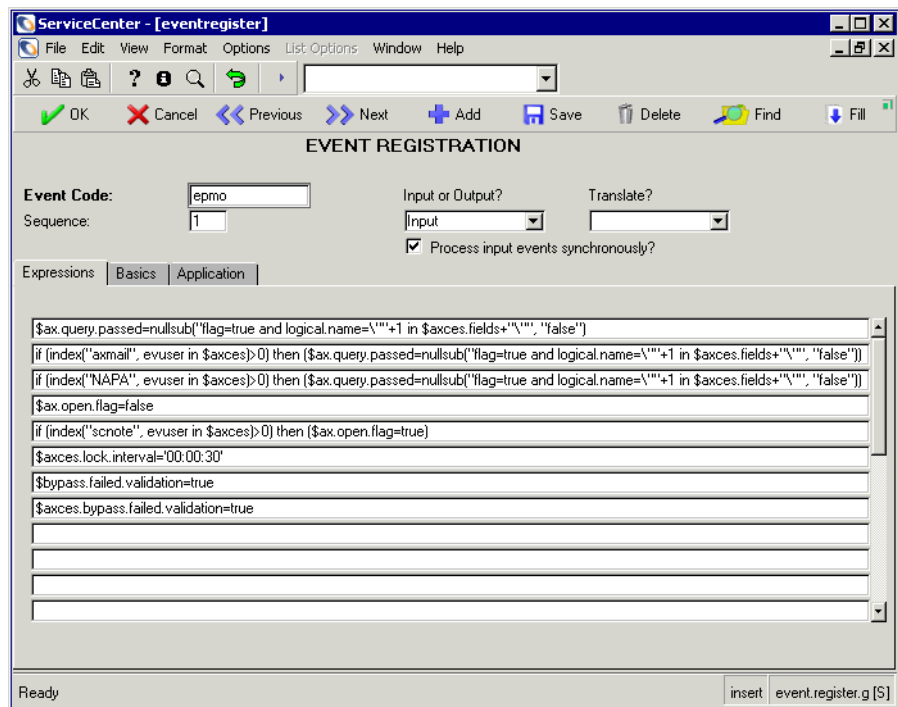


Figure 2-3: Event Registration Log

## Event Registration Fields

The encoded file names, as recorded in the `eventregister` file, are included for reference only.

### Header

Field	Description
Event Code ( <i>evtype</i> )	Unique code that identifies this registration.
Sequence ( <i>evseq</i> )	Number used to order the sequence of RAD applications to be executed for a single device type.
Input or Output ( <i>evftype</i> )	Flag to identify whether this registration is for an input or an output transaction; only input or output is acceptable.
Translate ( <i>evtranslate</i> )	Indicates whether to translate to upper ( <b>uc</b> ) or lower ( <b>lc</b> ) case; default is no translation.
Process input events synchronously? ( <i>synch.process</i> )	When selected (true), prompts the system to process the event as soon as the record is added to the database, rather than waiting for the event background scheduler to wake up and process all events in the <code>eventin</code> queue.

### Expressions Tab

Field	Description
Expressions ( <i>evinit</i> )	Array of statements that are executed at run time to initialize variables or initiate action based upon the contents of the data passed in the <code>eventin</code> ( <i>\$axces</i> ) and the <code>eventregister</code> ( <i>\$axces.register</i> ) records, and/or on global variables available at run time; the global variable <i>\$axces.fields</i> is used to represent an array of the fields passed in the <i>evfield</i> field of the <code>eventin</code> record.

## Basics Tab

Expressions	Basics	Application
Event Map Name:	<input type="text" value="e problem open"/>	Map Type <input type="text" value="Variable Length"/>
Format Name:	<input type="text"/>	<i>(optional: for output ONLY)</i>
Use Current Data?	<input type="text"/>	
Delete Condition:	<input type="text" value="false"/>	

Field	Description
Event Map Name ( <i>evmap</i> )	Name of the event map to be used.
Map Type ( <i>evmaptype</i> )	Determines the length of the map. This field is used for incoming events only. If <b>Variable Length</b> is selected, the Event Map Name is not used, and all the incoming data has no fixed length. If <b>Fixed Length</b> is selected, the Event Map Name is used, and the length is determined by the mapping definition(s).
Format Name ( <i>evformat</i> )	Used only for output events, the name of the format used to display the record.

**Field****Description**Use Current Data? (*evnullsub*)

A condition which, if true, will always substitute the current value in the target data when the external event passes a null value. For example, if an *icmu* event does not pass a value for **vendor** and the inventory item being updated has *Peregrine* in the *vendor* field, the result of mapping will leave *Peregrine* as the vendor. The event map allows specification of *evnullsub* on a field-by-field basis and will override this default when set in an individual map record.

Delete Condition (*evdelete*)

A condition whose result determines whether or not an *eventin* record will be deleted after it has been successfully processed.

**Application Tab**

Expressions Basics Application

Application Name:

Execute Condition:

Description	Parameter Names	Parameter Values
eventin record	record	\$axces
eventmap name	prompt	evmap in \$axces.register
problem file name	string1	probsummary
action to perform	text	close
probsummary query	query	\$ax.query.passed
write eventout?	boolean1	nullsub(evstatus in \$axces, "close")~#"error"

Application to Call on Error Condition:

Field	Description
Application Name ( <i>evappl</i> )	Name of the RAD application to execute.
Execute Condition ( <i>evcondition</i> )	A condition that, if <i>true</i> , allows the RAD application to be executed.
Description ( <i>comments</i> )	Array used to describe the elements in the Parameter Names and Parameter Values fields
Parameter Names ( <i>names</i> )	Array of parameter field names that are passed to the RAD application; these names must exist in the <b>application</b> file.
Parameter Values ( <i>values</i> )	Array of variables or literals that correspond to the list of parameter names passed in the <i>names</i> field; the data types must match.
Application to Call on Error Condition ( <i>evgoto</i> )	Name of a RAD application to call after execution of the primary application if the primary application fails due to an error condition; parameters may not be passed as local variables.

Registration is necessary for all input events that are processed by external applications. In Figure 2-4 on page 45, opening an incident is registered as *pmo*.

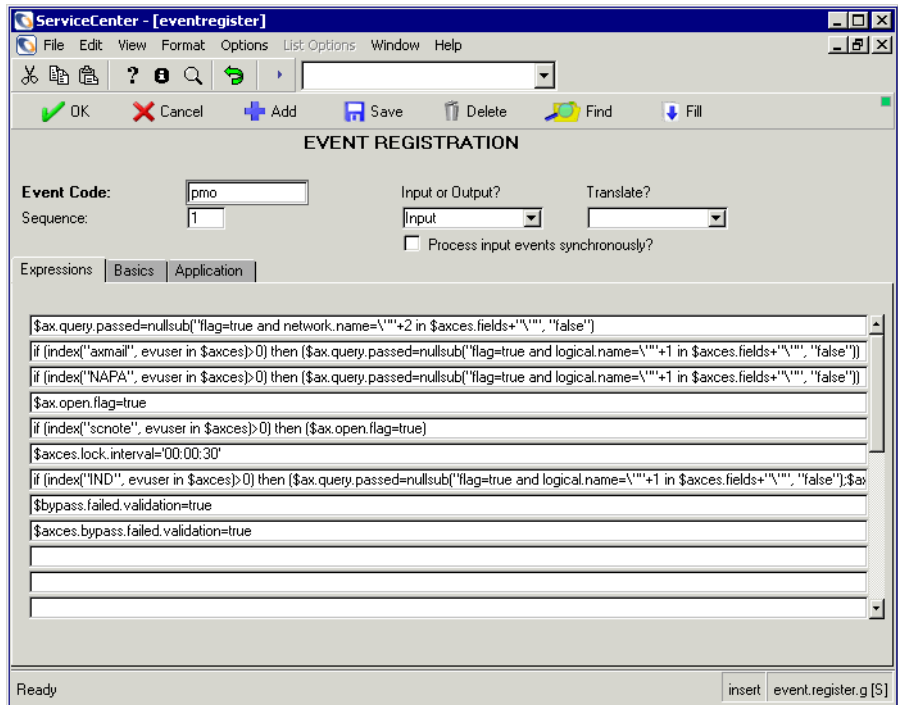


Figure 2-4: PMO Event Registration

When a *pmo* event occurs, an application called *axces.apm* is called if the condition evaluates to *true*. The parameters are passed by name and value, just as they are in the operator record. The Event Map Name identifies to the application which map should be used.

The expression statements in the previous figure are used to set up different queries depending upon the source of data. IPAS events depend on *network.name*, so the query will use *network.name* to select open incidents for update. The SCAuto mail incident event uses *logical.name*.

Expressions | Basics | Application

Application Name:

Execute Condition:

Description	Parameter Names	Parameter Values
eventin record	record	\$axces
eventmap name	prompt	evmap in \$axces.register
problem file name	string1	probsummary
action to perform	text	open
probsummary query	query	\$ax.query.passed
write eventout?	boolean1	nullsub(evstatus in \$axces,"")~#"error"
always open?	cond.input	\$ax.open.flag

Application to Call on Error Condition:

**Figure 2-5: Queries and Parameters through Event Registration**

This registration record instructs Event Services to select a record from the *probsummary* file (based on the query in *\$ax.query.passed*), then map data from the *eventin* record (*\$axces*), based on the *incident open* (evmap in *\$axces.register*) map record, then *open* an incident.

In most standard Event Services input applications, the first two parameters passed are the event record and the name of the event map. An exception in standard ServiceCenter SCAuto applications is email, which passes the mail record and the delimiter character.



## Commonly-Used Events

The following tables identify some of the more commonly-used events within each ServiceCenter application. Use the table as a quick reference for each event's function.

### Service Management Events

Event	Description
smin	Service Management incoming service request or help issue.
smout	Service Management output event.

### Incident Management Events

Event	Description
pmo	Opens an incident ticket.
pmu	Updates an incident ticket.
pmc	Closes an incident ticket.

### Inventory Management Events

Event	Description
icma	Adds an inventory item to the device file or updates the item if it already exists in the file.
icmu	Updates an inventory item.
icmd	Marks an inventory item for deletion.
prgma	Adds a software inventory item.
prgmu	Updates an inventory item.
prgmd	Deletes a software item.

## Change Management Events

Event	Description
cm3rin	Used for all incoming change events.
cm3rout	Created when a cm3message is activated. It represents a generic output message from a change phase.
cm3rinac	Used for acknowledging success in processing an incoming cm3rin event.
cm3tin	Used for incoming change events that communicate a generic message to a change task.
cm3tout	Created when a cm3message is activated. It represents a generic output message from a change task.
cm3tinac	Used for acknowledging success in processing an incoming cm3tin event.

## Request Management Events

Event	Description
rmoin	Request from an external application to open an order ticket in Request Management.
rmoappr	Request from an external application to enter an approval for an existing order ticket from one of the order's required approval group, or an approval user.
rmlin	Request from an external application to enter a new line item in an existing order ticket in Request Management.
rmqin	Request from an external application to enter a new quote in an order ticket in Request Management.
rmqappr	Request from an external application to enter an approval for a quote in an existing order ticket from one of the quote's required approval group or an approval user.

## Service Level Management Events

Event	Description
outagestart	Request from an external application to begin an outage against a device with an SLA.
outageend	Request from an external application to end an outage against a device with an SLA.
slaresponse	Request from an external application to enter a response time metric against a device with an SLA.

## Standard Events

ServiceCenter event registration currently supports events enabling integration with ERP, SAP, and other external system interfaces.

### CTSCPY (2)

This is an inbound event from SAP. It uses eventmap *cm3tctsc*.

It processes the inbound acknowledgment of SAP CTS Copy messages.

#### Application Called

axces.cm3

Parameters	Description
string1	Indicates which file of Change Management we should work with. In this case, <i>cm3r</i> is to be used.
record	Indicates what file variable is being passed in to the application. Since this is being invoked from the normal event services background operation, <i>\$axces</i> is the variable we want to use.
prompt	What map should be used to interpret the incoming message? By using <i>evmap</i> in <i>\$axces.register</i> , we actually tell the system to look at another field in the registration where it will find the name of the eventmap.
text	What action should be performed against the selected record in the Change Management system? The record should be updated to reflect any new information. The action should be <i>update</i> .

Parameters	Description
boolean1	Should outbound events be created? <i>false</i>
query	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event. The query used here is " <i>erp.parent.unique.id=\ "+str(2 in evlist in \$axces)+" and erp.development.sid=\ "+str(4 in evlist in \$axces)+" and erp.sid=\ "+str(5 in evlist in \$axces)+"</i> "
description	If the event scheduler is unable to obtain a lock for this change record, it will reschedule the event for a time in the future. This field allows us to identify what that interval should be. <i>'00:10:00'</i> is used to indicate 10 minutes.

## CTSCPY (1)

This is an outbound event to SAP. It uses eventmap *cm3tctsc*.

It is a generated message to SAP to copy a CTS Transport from one system to another.

### Application Called

axces.write

## CTSIMP (2)

This is an inbound event from SAP. Uses eventmap *cm3tctsi*.

It processes the inbound acknowledge message from SAP regarding Import of Transport.

### Application Called

axces.cm3

Parameters	Description
string1	Indicates which file of Change Management we should work with. In this case, <i>cm3t</i> is to be used.
record	Indicates what file variable is being passed in to the application. Since this is being invoked from the normal event services background operation, <i>\$axces</i> is the variable we want to use.

Parameters	Description
prompt	What map should be used to interpret the incoming message? By using <i>evmap</i> in <i>\$axces.register</i> , we actually tell the system to look at another field in the registration where it will find the name of the eventmap.
text	What action should be performed against the selected record in the Change Management system? Once the Import has been performed, the Task should be closed. The action for this is <i>close</i> .
boolean1	Should outbound events be created? <i>false</i>
query	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event. The query used here is <code>"erp.parent.unique.id=\ "+str(1 in evlist in \$axces)+"\ " and erp.sid=\ "+str(3 in evlist in \$axces)+"\ " and erp.client=\ "+str(4 in evlist in \$axces)+"\ "</code>
description	If the event scheduler is unable to obtain a lock for this change record, it will reschedule the event for a time in the future. This field allows us to identify what that interval should be. <code>'00:10:00'</code> is used to indicate 10 minutes.

## CTSIMP (1)

This is an outbound event to SAP. It uses eventmap *cm3tctsi*.

It builds a message to request a specific SAP Instance perform an Import of a given Transport.

### Application Called

*axces.write*

Parameters	Description
prompt	Indicates what data should be placed in the <i>evusrseq</i> field on the <i>eventout</i> record. This field is used by the ERP Gateway to determine what system it should be directed to. In this case, we use <i>erp.gateway.id</i> in <i>\$L.change</i>

## CTSIMP2

This event handles scheduling of outbound Import events to SAP. It uses eventmap *cm3tctsi*.

### Application Called

axces.cm3.cts.write

Parameters	Description
record	Indicates what file variable is being passed in to the application. Since this is being invoked from <i>cm3.message.pro</i> , <i>\$L.change</i> is the variable we want to use.
name	Indicates the System ID of the system which we want to send the message to and for which the schedule time must be calculated. In this case, <i>erp.sid</i> in <i>\$L.change</i> is the proper value.
prompt	Indicates the Client of the system which we want to send the message to and for which the schedule time must be calculated. In this case, <i>erp.client</i> in <i>\$L.change</i> is the proper value.
time1	What is the target time for this event? By using <i>tod()</i> here, we indicate to the system that we want the message to go at the next acceptable time.
query	Once the next “acceptable time” is found for this SID/client combination, a new schedule record will be generated for a message of this sort. <i>sap cts import scheduled</i> is used for rescheduled import messages.
boolean1	If this is true, the rescheduling portion of code will be skipped and all messages will happen as soon as possible. <i>erp.override.reschedule</i> in <i>\$L.change</i>

**Note:** Out of the box, CTSIMP2 is called whenever approval is received for a SAP Instance Import task. The routine *axces.cm3.cts.write* will determine an acceptable time to send a message to the target system. With the acceptable time calculated, a schedule record is generated to send the CTSIMP message at the calculated time.

## CTSRQCLS (2)

This is an inbound event from SAP. It uses eventmap *cm3rcts*.

It is a received message from SAP acknowledging transport release.

### Application Called

axces.cm3

Parameters	Description
string	Indicates which Change Management file should be worked with—in this case, <i>cm3r</i> .
record	Indicates what file variable is being passed in to the application. Since this is being invoked from the normal event services background operation, <i>\$axces</i> is the variable we want to use.
prompt	What map should be used to interpret the incoming message? By using <i>evmap</i> in <i>\$axces.register</i> , we actually tell the system to look at another field in the registration where it will find the name of a the eventmap.
text	What action should be performed against the selected record in Change Management? In this case, the ServiceCenter system should update the Change record with any information received from SAP; the appropriate value is <i>update</i> .
boolean1	Should outbound events be created? <i>false</i>
query	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event. The query used here is " <i>header,number="+str(1 in evlist in \$axces)+" and header,last=true"</i>
description	If the event scheduler is unable to obtain a lock for this change record, it will reschedule the event for a time in the future. This field allows us to identify what that interval should be. <i>'00:10:00'</i> is used to indicate 10 minutes.

## CTSRQCLS (1)

This is an outbound event to SAP. It uses eventmap *cm3rcts*.

It sends a message to a SAP instance instructing it to release a transport.

### Application Called

axces.write

Parameter	Description
prompt	Indicates what data should be placed in the <i>evusrseq</i> field on the <i>eventout</i> record. This field is used by the ERP Gateway to determine what system it should be directed to. In this case, we use <i>erp.development.gateway.id</i> in <i>\$L.change</i>

## CTSRQOPN (2)

This is an input event from SAP. It uses eventmap *cm3rcts*.

It is a received message from SAP acknowledging Transport Request creation. It is used to close the first phase of the Change, as well as update fields with data returned from SAP.

### Application Called

axces.cm3

Parameters	Description
string1	Indicates which file of Change Management we should work with. In this case, <i>cm3r</i> is to be used.
record	Indicates what file variable is being passed in to the application. Since this is being invoked from the normal event services background operation, <i>\$axces</i> is the variable we want to use.
prompt	What map should be used to interpret the incoming message? By using <i>evmap</i> in <i>\$axces.register</i> , we actually tell the system to look at another field in the registration where it will find the name of a the eventmap.



Parameters	Description
text	What action should be performed against the selected record in the Change Management system? Since this registration will only be invoked when it is an acknowledgment message of a ServiceCenter originated Request Open, the proper action is "close" [this phase and advance to the next.]
boolean1	Should outbound events be created? <i>false</i>
description	If the event scheduler is unable to obtain a lock for this change record, it will reschedule the event for a time in the future. This field allows us to identify what that interval should be. '00:10:00' is used to indicate 10 minutes.
query	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event. The query used here is "header,number="+str(1 in evlist in \$axces)+" and header,last=true"

## CTSRQOPN (3)

This is an input event from SAP. It uses eventmap *cm3rctso*.

It is a received message from SAP sent when a Transport Request is opened on the SAP side without first being opened within ServiceCenter. It causes a Change to be opened within ServiceCenter with data received from SAP.

### Application Called

*axces.cm3*

Parameter	Description
string1	Indicates which file of Change Management we should work with. In this case, <i>cm3r</i> is to be used.
record	Indicates what file variable is being passed in to the application. Since this is being invoked from the normal event services background operation, <i>\$axces</i> is the variable we want to use.
prompt	What map should be used to interpret the incoming message? By using <i>evmap</i> in <i>\$axces.register</i> , we actually tell the system to look at another field in the registration where it will find the name of a the eventmap.

Parameter	Description
text	What action should be performed against the selected record in the Change Management system? Since this registration will only be invoked when a Transport was opened in SAP and must be opened in ServiceCenter, the appropriate value is <i>open</i> .
boolean1	Should outbound events be created? <i>false</i>
description	If the event scheduler is unable to obtain a lock for this change record, it will reschedule the event for a time in the future. This field allows us to identify what that interval should be. '00:10:00' is used to indicate 10 minutes.
query	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event. The query used here is "header,number="+str(1 in evlist in \$axces)+" and header,last=true"

## CTSRQOPN (1)

This is an outbound system event sent to SAP. It uses the *cm3rcts* eventmap.

It sends a message to a SAP instance instructing it to open a SAP Transport Request with certain ServiceCenter supplied data.

### Application Called

axces.write

Parameters	Description
prompt	Indicates what data should be placed in the <i>evusrseq</i> field on the <i>eventout</i> record. This field is used by the ERP Gateway to determine what system it should be directed to. In this case, we use <i>erp.development.gateway.id</i> in <i>\$L.change</i>

## CTSRQUPD (2)

This is an inbound event from SAP. It uses eventmap *cm3rcts*.

It is a received message from SAP sent when a Transport Request has been updated on the SAP side. It is either an acknowledgment of a ServiceCenter originated request or a notification of a SAP originated action.

### Application Called

axces.cm3

Parameter	Description
string1	Indicates which file of Change Management we should work with. In this case, <i>cm3r</i> is to be used.
record	Indicates what file variable is being passed in to the application. Since this is being invoked from the normal event services background operation, <i>\$axces</i> is the variable we want to use.
prompt	What map should be used to interpret the incoming message? By using <i>evmap</i> in <i>\$axces.register</i> , we actually tell the system to look at another field in the registration where it will find the name of a the eventmap.
text	What action should be performed against the selected record in Change Management? Since this registration is used to convey update information, the appropriate value is <i>update</i> .
boolean1	Should outbound events be created? <i>false</i>
query	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event. The query used here is " <i>header,number="+str(1 in evlist in \$axces)+" and header,last=true"</i>
description	If the event scheduler is unable to obtain a lock for this change record, it will reschedule the event for a time in the future. This field allows us to identify what that interval should be. <i>'00:10:00'</i> is used to indicate 10 minutes.

## CTSRQUPD (1)

This is an outbound event to SAP. It uses eventmap *cm3rcts*.

It sends a message to a SAP Instance indicating that a Transport Request should have certain data elements updated.

### Application Called

axces.write

Parameters	Description
prompt	Indicates what data should be placed in the <i>evusrseq</i> field on the <i>eventout</i> record. This field is used by the ERP Gateway to determine what system it should be directed to. In this case, we use <i>erp.development.gateway.id</i> in <i>\$.change</i>

## CTSTKCLS (2)

This is an inbound event from SAP. It uses eventmap *cm3tcts*.

It is a received message from SAP when a Transport Task has been closed on the SAP side. It is either an acknowledgment of a ServiceCenter-originated request or as notification on a SAP-originated action.

### Application Called

axces.cm3

Parameters	Description
string1	Indicates which file of Change Management we should work with. In this case, <i>cm3t</i> is to be used.
record	Indicates what file variable is being passed in to the application. Since this is being invoked from the normal event services background operation, <i>\$axces</i> is the variable we want to use.
prompt	What map should be used to interpret the incoming message? By using <i>evmap</i> in <i>\$axces.register</i> , we actually tell the system to look at another field in the registration where it will find the name of a the eventmap.

Parameters	Description
text	What action should be performed against the selected record in the Change Management system? When a Task is closed, the system must perform actions. The code to be used here is <i>close</i> .
boolean1	Should outbound events be created? <i>false</i>
query	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event. The query used here is " <i>header,number="+str(10 in evlist in \$axces)+" and header,last=true"</i>
description	If the event scheduler is unable to obtain a lock for this change record, it will reschedule the event for a time in the future. This field allows us to identify what that interval should be. <i>'00:10:00'</i> is used to indicate 10 minutes.

## CTSTKCLS (1)

This is an outbound event to SAP. It uses eventmap **cm3tcts**.

It sends a message to SAP indicating that a Transport Task should be closed within SAP.

### Application Called

axces.write

Parameters	Description
prompt	Indicates what data should be placed in the <i>evusrseq</i> field on the <i>eventout</i> record. This field is used by the ERP Gateway to determine what system it should be directed to. In this case, we use <i>erp.development.gateway.id</i> in <i>\$.change</i>

## CTSTKOPN (2)

This is an inbound event from SAP. It uses eventmap `cm3tcts`.

It is an acknowledgment message from SAP indicating that a Transport Task has been closed on the SAP side.

### Application Called

`axces.cm3`

Parameters	Description
<code>string1</code>	Indicates which file of Change Management we should work with. In this case, <code>cm3t</code> is to be used.
<code>record</code>	Indicates what file variable is being passed in to the application. Since this is being invoked from the normal event services background operation, <code>\$axces</code> is the variable we want to use.
<code>prompt</code>	What map should be used to interpret the incoming message? By using <code>evmap</code> in <code>\$axces.register</code> , we actually tell the system to look at another field in the registration where it will find the name of a the eventmap.
<code>text</code>	What action should be performed against the selected record in the Change Management system? This registration will only be invoked as an acknowledgment of ServiceCenter-originated Task Opens. This registration should post any SAP-provided information to the Task via the <code>update</code> action.
<code>boolean1</code>	Should outbound events be created? <i>false</i>
<code>query</code>	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event. The query used here is <code>"header,number="+str(10 in evlist in \$axces)+" and header,last=true"</code>
<code>description</code>	If the event scheduler is unable to obtain a lock for this change record, it will reschedule the event for a time in the future. This field allows us to identify what that interval should be. <code>'00:10:00'</code> is used to indicate 10 minutes.

## CTSTKOPN (3)

This is an inbound event from SAP. It uses eventmap `cm3tctso`.

This message indicates that a Transport Task has been opened on the SAP side. It opens a Task within ServiceCenter's Change Management utilizing data supplied from the SAP system.

### Application Called

`axces.cm3`

Parameters	Description
<code>string1</code>	Indicates which file of Change Management we should work with. In this case, <code>cm3t</code> is to be used.
<code>record</code>	Indicates what file variable is being passed in to the application. Since this is being invoked from the normal event services background operation, <code>\$axces</code> is the variable we want to use.
<code>prompt</code>	What map should be used to interpret the incoming message? By using <code>evmap</code> in <code>\$axces.register</code> , we actually tell the system to look at another field in the registration where it will find the name of a the eventmap.
<code>text</code>	What action should be performed against the selected record in the Change Management system? In this case, a Transport Task has been opened first on the SAP side. ServiceCenter must be updated to reflect this information, so a new task must be opened. The action for this is <code>open</code> .
<code>boolean1</code>	Should outbound events be created? <code>false</code>
<code>query</code>	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event. The query used here is <code>"header,number="+str(10 in evlist in \$axces)+" and header,last=true"</code>
<code>description</code>	'00:10:0 If the event scheduler is unable to obtain a lock for this change record, it will reschedule the event for a time in the future. This field allows us to identify what that interval should be.'00:10:00' is used to indicate 10 minutes.0'

## CTSTKOPN (1)

This is an outbound event to SAP. It uses eventmap `cm3tcts`.

It sends a message to SAP indicating that a Transport Task should be opened within SAP.

### Application Called

`axces.write`

Parameters	Description
<code>prompt</code>	Indicates what data should be placed in the <code>evusrseq</code> field on the <code>eventout</code> record. This field is used by the ERP Gateway to determine what system it should be directed to. In this case, we use <code>erp.development.gateway.id</code> in <code>\$.L.change</code>

## CTSTKUPD (2)

This is an inbound event from SAP. It uses eventmap `cm3tcts`.

It is a message received from SAP when a Transport Task has been updated on the SAP side. It can either be an acknowledgment of a ServiceCenter-originated update or notification of a SAP-originated update.

### Application Called

`axces.cm3`

Parameters	Description
<code>string1</code>	Indicates which file of Change Management we should work with. In this case, <code>cm3t</code> is to be used.
<code>record</code>	Indicates what file variable is being passed in to the application. Since this is being invoked from the normal event services background operation, <code>\$axces</code> is the variable we want to use.
<code>prompt</code>	What map should be used to interpret the incoming message? By using <code>evmap</code> in <code>\$axces.register</code> , we actually tell the system to look at another field in the registration where it will find the name of a the eventmap.



Parameters	Description
text	What action should be performed against the selected record in the Change Management system? Since this registration is used to convey update information, the appropriate value is <i>update</i> .
boolean1	Should outbound events be created? <i>false</i>
query	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event. The query used here is " <i>header,number="+str(10 in evlist in \$axces)+" and header,last=true"</i>
description	If the event scheduler is unable to obtain a lock for this change record, it will reschedule the event for a time in the future. This field allows us to identify what that interval should be. <i>'00:10:00'</i> is used to indicate 10 minutes.

## CTSTKUPD (1)

This is an outbound event to SAP. It uses eventmap *cm3tcts*.

It is sent from ServiceCenter to SAP to indicate that a Transport Task should be updated on the SAP side to match changes on the ServiceCenter side.

### Application Called

axces.write

Parameters	Description
prompt	Indicates what data should be placed in the <i>evusrseq</i> field on the <i>eventout</i> record. This field is used by the ERP Gateway to determine what system it should be directed to. In this case, we use <i>erp.development.gateway.id</i> in <i>\$L.change</i>

## ERPHR (1)

This input event establishes contact with the ERP system. It uses the *contactserp* event map.

### Application Called

axces.database

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If an variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name using in event registration. In this case, <i>contactserp</i> is used.
string1	Identifies the file in ServiceCenter where to record data for branching or changing tasks. In this case, <i>contacts</i> is used.
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> , <i>close</i> , <i>approve</i> , <i>disapprove</i> or <i>unapprove</i> —in this case, <i>add</i> .
query	Provides conditional query. In this case, <i>contact.name=1</i> in <i>\$axces.fields</i> is used.
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion, using the same name as the input event with <i>ac</i> appended. In this case, the value is <i>true</i> .
cond.input	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether or not to process additional records; in this case, the value is <i>false</i> .
name	Identifies the Format Control record used—in this case, <i>operator.scauto</i> .

## ERPHR (2)

This is the output version of this event. It uses the event map *contactserp*.

### Application Called

axces.write

## ERPSTATES (1)

This is an input event that determines the state of the ERP system. It uses the *stateerp* event map.

### Application Called

axces.database

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If an variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name using in event registration—in this case, <i>stateerp</i> .
string1	Identifies the file in ServiceCenter where to record data for branching or changing tasks—in this case, <i>state</i> .
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> , <i>close</i> , <i>approve</i> , <i>disapprove</i> or <i>unapprove</i> —in this case, <i>add</i> .
query	Provides conditional query. In this case <i>state.code=1</i> in <i>\$axces.fields</i> is used.
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion, using the same name as the input event with <i>ac</i> appended. In this case, the value is <i>true</i> .
cond.input	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether or not to process additional records. In this case, the value is <i>false</i> .
name	Identifies the Format Control record used—in this case, <i>operator.scauto</i> .

## ERPSTATES (2)

This is the output version of this event. It uses the *stateerp* event map

### Application Called

axces.write

## ICMapplication

ServiceCenter inventory regulation event when a device of this type is added to the system.

### Application Called

axces.database

Parameter	Value	Description
record	\$axces	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
prompt	icm application	This field indicates the Map Name used in event registration—in this case, <i>ICM application</i> .
string1	device	Identifies the ServiceCenter file in which data is recorded—in this case, <i>device</i> .
text	add	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>add</i> .
query	logical.name=1 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	false	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. In this case, the value is <i>false</i> .
name	ICMapplication	Specifies the Format Control record to use—in this case, <i>ICMapplication</i> .

## ICMcomputer

Same settings as ICMapplication, except for the following settings:

Parameter	Value
prompt	icm computer
string1	joincomputer
name	ICMcomputer

## ICMdevice

Events of this type are used when you add data records to the device file. These events use the *icm device* event maps.

### Application Called

icm.process.event

Parameter	Value	Description
record	\$axces	This field writes the unique record identifier (for example, problem number) to the eventin record. If a variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
prompt	icm device	This field indicates the Map Name using in event registration—in this case, <i>icm device</i> .
string1	device	Identifies the file in ServiceCenter in which to record data for branching or changing tasks—in this case, <i>device</i> .
text	add	Provides the specific action to take; may be <i>add</i> , <i>update</i> , <i>close</i> , <i>approve</i> , <i>disapprove</i> or <i>unapprove</i> —in this case, <i>add</i> .
query	logical.name=1 in \$axces.fields	Provides conditional query, in this case <i>logical.name=1 in \$axces.fields</i> .
boolean1	false	This logical field flag ( <i>true/false</i> ) indicates whether or not eventout record should be written upon transaction completion, using the same name as the input event with <i>ac</i> appended. The value in the out-of-box system is <i>false</i> .
name	ICMdevice	Identifies the event registration name—in this case, <i>ICMdevice</i> .

## ICMdisplaydevice

Same settings as ICMapplication, except for the following settings:

Parameter	Value
prompt	icm displaydevice
string1	joindisplaydevice
name	ICMdisplaydevice

## ICMexample

Same settings as ICMapplication, except for the following settings:

Parameter	Value
prompt	icm example
string1	joinexample
name	ICMexample

## ICMfurnishings

Same settings as ICMapplication, except for the following settings:

Parameter	Value
prompt	icm furnishings
string1	joinfurnishings
name	ICMfurnishings

## ICMhandhelds

Same settings as ICMapplication, except for the following settings:

Parameter	Value
prompt	icm handhelds
string1	joinhandhelds
name	ICMhandhelds

## ICMmainframe

Same settings as ICMapplication, except for the following settings:

Parameter	Value
prompt	icm mainframe
string1	joinmainframe
name	ICMmainframe

## ICMnetworkcomponents

Same settings as ICMapplication, except for the following settings:

Parameter	Value
prompt	icm networkcomponents
string1	joinnetworkcomponents
name	ICMnetworkcomponents

## ICMofficeelectronics

Same settings as ICMapplication, except for the following settings:

Parameter	Value
prompt	icm officeelectronics
string1	joinofficeelectronics
name	ICMofficeelectronics

## ICMsoftwarelicense

Same settings as ICMapplication, except for the following settings:

Parameter	Value
prompt	icm softwarelicense
string1	joinsoftwarelicense
name	ICMsoftwarelicense

## ICMstorage

Same settings as ICMapplication, except for the following settings:

Parameter	Value
prompt	icm storage
string1	joinstorage
name	ICMstorage

## ICMtelecom

Same settings as ICMapplication, except for the following settings:

Parameter	Value
prompt	icm telecom
string1	jointelecom
name	ICMtelecom

## GetResRM

This is an input event that provides access to Request Management.

### Application Called

axces.rm

Parameter	Description
record	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
text	Provides the specific action to take, may be open, update, or close—3 in <i>evlist</i> in <i>\$axces</i> .
prompt	This field indicates the Map Name used in event registration— <i>evmap</i> in <i>\$axces.register</i> .
query	The query string to select a pre-existing record to update—in this case, not set.



Parameter	Description
string1	Identifies the file (ocmq, ocmo, ocml) in ServiceCenter where data is recorded—in this case, <i>ocmq</i> .
boolean1	The logical field flag (true/false) indicates whether <i>eventout</i> record should be written upon transaction completion. In this case, the value is <i>true</i> .

## GetResRM

An output event for Request Management that calls *axces.write*.

## GetResRML

An input event that provides access to Request Management.

### Application Called

GetResRML

Parameters	Description
record	This field identifies the <i>eventin</i> record to be passed—in this case, the value of <i>\$axces</i> .
text	Provides the specific action to take, may be open, update, or close—3 in <i>evlist</i> in <i>\$axces</i> .
prompt	This field indicates the Map Name used in event registration— <i>evmap</i> in <i>\$axces.register</i> .
query	The query string to select a pre-existing record to update—in this case, not set.
string1	Identifies the file (ocmq, ocmo, ocml) in ServiceCenter where data is recorded—in this case, <i>ocml</i> .
boolean1	The logical field flag (true/false) indicates whether <i>eventout</i> record should be written upon transaction completion. In this case, the value is <i>true</i> .

## GetResRML

An output event for Request Management and calls *axces.write*.

## HotNews

An output event.

## IND

An input event that adds or updates an inventory item(s) to the device file.

### Application Called

scauto.inventory

Parameter	Value	Description
record	\$axces	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
prompt	evmap in \$axces.register	This field indicates the Map Name using in event registration—in this case, evmap in <i>\$axces.register</i> .
string1	device	Identifies the file in ServiceCenter in which data is recorded—in this case, <i>device</i> .
text	add	Provides the specific action to take; may be <i>add</i> , <i>update</i> , <i>close</i> , <i>approve</i> , <i>disapprove</i> or <i>unapprove</i> —in this case, <i>add</i> .
query	logical.name=7 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	true	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. In this case, the value is <i>true</i> .
name	icma	Identifies the event registration name—in this case, <i>icma</i> (the legacy name).

## NDpmc

This is an inbound event coming from Network Discovery to close automatically-generated incident tickets.

### Application Called

axces.apm

Parameter	Value
record	\$axces
prompt	evmap in \$axces.register
string1	probsummary
text	close
query	\$ax.query.passed
boolean1	nullsub(evstatus in \$axces, "close")~#"error"

## NDpmc

This is an outbound event that returns an incident ticket number when Network Discovery closes an incident.

### Application Called

axces.write

Parameter	Value
record	\$axces
name	pmc
string1	^
query	evuser in \$axces
prompt	nullsub(evusrseq in \$axces, evsysseq in \$axces)

## NDpmo

This is an inbound event coming from Network Discovery to open, update, or re-open an incident ticket.

### Application Called

axces.apm

Parameter	Value
record	\$axces
prompt	evmap in \$axces.register
string1	probsummary
text	open
query	\$ax.query.passed
boolean1	nullsub(evstatus in \$axces,"")~#"error"
cond.input	\$ax.open.flag

## NDpmo

This is an outbound event that returns an incident ticket number when Network Discovery opens, updates, or re-opens an incident ticket.

### Application Called

axces.write

Parameter	Value
record	\$axces
name	pmo
string1	^
query	evuser in \$axces
prompt	nullsub(evusrseq in \$axces, evsysseq in \$axces)

## PSSDELETE

An input event that deletes selected records from a ServiceCenter file.

### Application Called

pss.delete

Parameter	Value	Description
record	\$axces	This field identifies the event in record to be passed—in this case, the value of <i>\$axces</i> .
prompt	evmap in \$axces.register	This field indicates the Map Name used in event registration—in this case, evmap in <i>\$axces.register</i> .
string1	\$L.name	Identifies the file in ServiceCenter from which data will be deleted—in this case, <i>\$L.name</i> .

## SALESQUOTE

An input event that moves an event in record to the event out file and changes the evtype.

### Application Called

axces.move.intoout

Parameter	Value	Description
record	\$axces	This field identifies the event in record to be passed—in this case, the value of <i>\$axces</i> .
name	SALESORDERPARSE	Identifies the event out type—in this case, SALESORDERPARSE.

## SAPGRT

Goods receipt output event. It calls no application but submits receipt notification to SAP system for processing.

## SAPGRT

This is an input event.

### Application Called

axces.rm

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If an variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> , <i>close</i> , <i>approve</i> , <i>disapprove</i> or <i>unapprove</i> —in this case, <i>update</i> .
prompt	This field indicates the Map Name using in event registration—in this case, <i>evmap in \$axces.registrater</i> .
query	Provides conditional query, in this case <i>number=21 in evlist in \$axces</i>
string1	Identifies the file in ServiceCenter in which to record data for branching or changing tasks—in this case, <i>ocml</i> .
name	Identifies the operator (security profile) in ServiceCenter that should be used and what userID should be stamped on records processed via this event—in this case, <i>falcon</i> .

## SAPGTE

### Application Called

axces.rm

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If an variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> , <i>close</i> , <i>approve</i> , <i>disapprove</i> or <i>unapprove</i> —in this case, <i>update</i> .

Parameters	Description
prompt	This field indicates the Map Name using in event registration—in this case, <i>evmap in \$axces.registrater</i> .
query	Provides conditional query, in this case <i>number=21 in evlist in \$axces</i>
string1	Identifies the file in ServiceCenter in which to record data for branching or changing tasks—in this case, <i>ocml</i> .
name	Identifies the operator (security profile) in ServiceCenter that should be used and what userID should be stamped on records processed via this event—in this case, <i>falcon</i> .

## SAPHR (1)

Input event processing edits to contact file originating in SC, routed through SAP, and returned to SC.

### Application Called

*axces.database*

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If an variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name using in event registration—in this case, <i>contactssap</i> .
string1	Identifies the file in ServiceCenter where to record data for branching or changing tasks—in this case, <i>contacts</i> .
text	Provides the specific action to take, may be <i>add, update, close, approve, disapprove</i> or <i>unapprove</i> —in this case, <i>add</i> .
query	Provides conditional query, in this <i>contact.name=2 in \$axces.fields</i>
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion, using the same name as the input event with <i>ac</i> appended. In this case, the value is <i>true</i> .

Parameters	Description
cond.input	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether or not to process additional records. In this case, the value is <i>false</i> .
name	Identifies the Format Control record—in this case, <i>operator.scauto</i> .

## SAPHR (2)

Same settings as SAPHR except output event routing contact file changes to SAP and calls the `axces.write` application.

## SAPHRMD

Input event processing SAP-originating contacts file changes.

### Application Called

`axces.database`

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the <code>eventin</code> record. If an variable like <code>\$axces</code> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name using in event registration—in this case, <i>contactssap</i> .
string1	Identifies the file in ServiceCenter where to record data for branching or changing tasks—in this case, <code>contacts</code> .
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> , <i>close</i> , <i>approve</i> , <i>disapprove</i> or <i>unapprove</i> —in this case, <i>add</i> .
query	Provides conditional query, in this <i>contact.name=2</i> in <i>\$axces.fields</i>
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether <code>eventout</code> record should be written upon transaction completion, using the same name as the input event with <i>ac</i> appended. In this case, the value is <i>true</i> .



Parameters	Description
cond.input	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether or not to process additional records. In this case, the value is <i>false</i> .
name	Identifies the Format Control record—in this case, <i>operator.scauto</i> .

## SAPORD

Sales order outbound event to SAP. Calls no application, but routes order information to SAP for processing.

## SAPORD

Sales order inbound event from SAP, used to break event into appropriate constituent parts.

### Application Called

axces.sap.hybrid.evin

Parameters	Description
record	This hybrid event breaks the incoming event into multiple events and calls the appropriate routines for <i>header</i> and <i>detail</i> events. It expects <b>one</b> <i>header</i> and any number of properly formed <i>detail</i> events
prompt	Identifies the eventregister which should be used to interpret the <i>detail</i> part of the event message.
name	Identifies the eventregister that should be used to interpret the <i>header</i> piece of the event stream.

## SAPORDQ

*Header* component of SAPORD event.

### Application Called

axces.rm

Parameters	Description
record	Header component created by the SAPORD event.
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> , <i>close</i> , <i>approve</i> , <i>disapprove</i> or <i>unapprove</i> —in this case, <i>update</i> .
prompt	This field indicates the Map Name used in event registration—in this case, <i>evmap</i> in <i>\$axcess.register</i> .
query	Provides conditional query, in this case <i>number=9</i> in <i>evlist</i> in <i>\$axces</i>
string1	Identifies the file in ServiceCenter where to record data for branching or changing tasks—in this case, <i>ocmq</i> .
name	Identifies the operator (security profile) in ServiceCenter that should be used and what userID should be stamped on records processed via this event—in this case, <i>falcon</i> .

## SAPQTE

Outbound sales quote event from SC, calls no application.

## SAPQTE

Inbound sales quote event from SAP, used to break event into constituent parts.

### Application Called

axces.sap.hybrid.evin

Parameters	Description
record	Hybrid definition used to identify <i>header</i> (SAPQTEQ) and <i>detail</i> (sapqtel) event registrations.
prompt	Identifies the eventregister which should be used to interpret the <i>detail</i> part of the event message.
name	Identifies the eventregister that should be used to interpret the <i>header</i> piece of the event stream.

## SAPQTEQ

*Header* component of inbound SAPQTE sales quote.

### Application Called

axces.rm

Parameters	Description
record	Header component created by the SAPQTE event.
prompt	Identifies the Map Name used in event registration—in this case, <i>evmap</i> in <i>\$axces.register</i> .
string1	Identifies the file in ServiceCenter where data is recorded for this event—in this case, <i>ocmq</i> .
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> , <i>close</i> , <i>approve</i> , <i>disapprove</i> or <i>unapprove</i> —in this case, <i>update</i> .
query	Provides conditional query, in this case <i>number=10</i> in <i>evlist</i> in <i>\$axces</i> .
name	Identifies the operator (security profile) in ServiceCenter that should be used and what userID should be stamped on records processed via this event—in this case, <i>falcon</i> .

## SAPREQ

Outbound purchase requisitions from SC, calls no application.

## SAPREQ

Inbound purchase requisition from SAP.

### Application Called

axces.sap.hybrid.evin

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If an variable like \$axces is used, it needs to be set up using Format Control to create the intended result.
name	Identifies the event registration name—in this case, <i>SAPREQO</i> .
prompt	This field indicates the Map Name using in event registration—in this case, <i>sapreql</i> .

## SAPREQO

Component of inbound SAPREQ from SAP.

### Application Called

axces.rm

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If an variable like \$axces is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name using in event registration—in this case, <i>evmap</i> in <i>\$axces.register</i> .
string1	Identifies the file in ServiceCenter where to record data for branching or changing tasks—in this case, <i>ocmo</i> .
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> , <i>close</i> , <i>approve</i> , <i>disapprove</i> or <i>unapprove</i> —in this case, <i>update</i> .

Parameters	Description
query	Provides conditional query, in this case <i>number=5 in evlist in \$axces</i>
name	Identifies the operator (security profile) in ServiceCenter that should be used and what userID should be stamped on records processed via this event.

## ScAcBrand

An input event that allows ServiceCenter and AssetCenter to integrate data from the ServiceCenter vendor file to the corresponding AssetCenter file.

### Application Called

axces.database

Parameter	Value	Description
record	\$axces	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
prompt	ScAcBrand	This field indicates the Map Name used in event registration—in this case, <i>ScAcBrand</i> .
string1	vendor	Identifies the ServiceCenter file in which data is recorded—in this case, <i>vendor</i> .
text	add	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>add</i> .
query	vendor = 1 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	false	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. In this case, the value is <i>false</i> .
cond.input	true	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether to process additional records. In this case, the value is <i>true</i> .
name	vendor	Specifies the Format Control record to use—in this case, <i>vendor</i> .

## ScAcCompany

An input event that allows ServiceCenter and AssetCenter to integrate data from the ServiceCenter company file to the corresponding AssetCenter file.

### Application Called

axces.database

Parameter	Value	Description
record	\$axces	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
prompt	ScAcCompany	This field indicates the Map Name used in event registration—in this case, <i>ScAcCompany</i> .
string1	company	Identifies the ServiceCenter file in which data is recorded—in this case, <i>company</i> .
text	add	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>add</i> .
query	customer.id = 1 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	false	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. In this case, the value is <i>false</i> .
cond.input	true	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether to process additional records. In this case, the value is <i>true</i> .
name	company	Specifies the Format Control record to use—in this case, <i>company</i> .

## ScAcContacts

An input event that allows ServiceCenter and AssetCenter to integrate data from the ServiceCenter contacts file to the corresponding AssetCenter file.

### Application Called

axces.database

Parameter	Value	Description
record	\$axces	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
prompt	ScAcContacts	This field indicates the Map Name used in event registration—in this case, <i>ScAcContacts</i> .
string1	contacts	Identifies the ServiceCenter file in which data is recorded—in this case, <i>contacts</i> .
text	add	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>add</i> .
query	contact.name = 1 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	false	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. In this case, the value is <i>false</i> .
cond.input	true	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether to process additional records. In this case, the value is <i>true</i> .
name	contacts	Specifies the Format Control record to use—in this case, <i>contacts</i> .

## ScAcDept

An input event that allows ServiceCenter and AssetCenter to integrate data from the ServiceCenter department file to the corresponding AssetCenter file.

### Application Called

axces.database

Parameter	Value	Description
record	\$axces	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
prompt	ScAcDept	This field indicates the Map Name used in event registration—in this case, <i>ScAcDept</i> .
string1	dept	Identifies the ServiceCenter file in which data is recorded—in this case, <i>dept</i> .
text	add	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>add</i> .
query	dept.id = 2 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	false	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. In this case, the value is <i>false</i> .
cond.input	true	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether to process additional records. In this case, the value is <i>true</i> .
name	dept	Specifies the Format Control record to use—in this case, <i>dept</i> .



## ScAcDevice

An input event that allows ServiceCenter and AssetCenter to integrate data from the ServiceCenter device file to the corresponding AssetCenter file.

### Application Called

scauto.inventory

Parameter	Value	Description
record	\$axces	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
prompt	evmap in \$axces.register	This field indicates the Map Name used in event registration—in this case, <i>evmap in \$axces.register</i> .
string1	device	Identifies the ServiceCenter file in which data is recorded—in this case, <i>device</i> .
text	add	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>add</i> .
query	logical.name =1 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	true	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. In this case, the value is <i>true</i> .
name	icma	Specifies the Format Control record to use—in this case, <i>icma</i> .

## ScAcLocation

An input event that allows ServiceCenter and AssetCenter to integrate data from the ServiceCenter location file to the corresponding AssetCenter file.

### Application Called

axces.database

Parameter	Value	Description
record	\$axces	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
prompt	ScAcLocation	This field indicates the Map Name used in event registration—in this case, <i>ScAcLocation</i> .
string1	location	Identifies the ServiceCenter file in which data is recorded—in this case, <i>location</i> .
text	add	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>add</i> .
query	location = 2 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	false	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. In this case, the value is <i>false</i> .
cond.input	true	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether to process additional records. In this case, the value is <i>true</i> .
name	location	Specifies the Format Control record to use— <i>location</i> in this case.

## ScAcModel

An input event that allows ServiceCenter and AssetCenter to integrate data from the ServiceCenter model file to the corresponding AssetCenter file.

### Application Called

axces.database

Parameter	Value	Description
record	\$axces	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
prompt	ScAcModel	This field indicates the Map Name used in event registration—in this case, <i>ScAcModel</i> .
string1	model	Identifies the ServiceCenter file in which data is recorded—in this case, <i>model</i> .
text	add	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>add</i> .
query	part.no = 1 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	val("false", 4)	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. In this case, the value is <i>val("false", 4)</i> .
cond.input	true	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether to process additional records. In this case, the value is <i>true</i> .
name	model	Specifies the Format Control record to use— <i>model</i> in this case.

## ScAcModelBundle

An input event that allows ServiceCenter and AssetCenter to integrate data from the ServiceCenter model file to the corresponding AssetCenter file.

### Application Called

axces.database

Parameter	Value	Description
record	\$axces	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
prompt	ScAcModelBundle	This field indicates the Map Name used in event registration—in this case, <i>ScAcModelBundle</i> .
string1	model	Identifies the ServiceCenter file in which data is recorded—in this case, <i>model</i> .
text	add	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>add</i> .
query	part.no = 1 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	val("false", 4)	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. In this case, the value is <i>val("false", 4)</i> .
cond.input	true	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether to process additional records. In this case, the value is <i>true</i> .
name	model	Specifies the Format Control record to use— <i>model</i> in this case.

## ScAcModelVendor

An input event that allows ServiceCenter and AssetCenter to integrate data from the ServiceCenter modelvendor file to the corresponding AssetCenter file.

### Application Called

axces.database

Parameter	Value	Description
record	<code>\$axces</code>	This field identifies the eventin record to be passed—in this case, the value of <code>\$axces</code> .
prompt	ScAcModelVendor	This field indicates the Map Name used in event registration—in this case, <code>ScAcModelVendor</code> .
string1	modelvendor	Identifies the ServiceCenter file in which data is recorded—in this case, <code>modelvendor</code> .
text	add	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>add</i> .
query	part.no = 1 in <code>\$axces.fields</code> and vendor = 2 in <code>\$axces.fields</code>	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	false	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. In this case, the value is <i>false</i> .
cond.input	true	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether to process additional records. In this case, the value is <i>true</i> .
name	modelvendor	Specifies the Format Control record to use—in this case, <code>modelvendor</code> .

## ScAcVendor

An input event that allows ServiceCenter and AssetCenter to integrate data from the ServiceCenter vendor file to the corresponding AssetCenter file.

### Application Called

axces.database

Parameter	Value	Description
record	\$axces	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
prompt	ScAcVendor	This field indicates the Map Name used in event registration—in this case, <i>ScAcVendor</i> .
string1	vendor	Identifies the ServiceCenter file in which data is recorded—in this case, <i>vendor</i> .
text	add	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>add</i> .
query	vendor = 2 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	false	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. In this case, the value is <i>false</i> .
cond.input	true	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether to process additional records. In this case, the value is <i>true</i> .
name	vendor	Specifies the Format Control record to use—in this case, <i>vendor</i> .

## ScAcVendorBACK

An input event that allows ServiceCenter and AssetCenter to integrate data from the ServiceCenter vendor file to the corresponding AssetCenter file.

### Application Called

axces.database

Parameter	Value	Description
record	\$axces	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
prompt	ScAcVendor	This field indicates the Map Name used in event registration—in this case, <i>ScAcVendor</i> .
string1	vendor	Identifies the ServiceCenter file in which data is recorded—in this case, <i>vendor</i> .
text	add	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>add</i> .
query	vendor=9 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	true	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. In this case, the value is <i>true</i> .
cond.input	false	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether to process additional records. In this case, the value is <i>false</i> .

## ScFcOrderLine

An input event that allows ServiceCenter and FacilityCenter to integrate data from the ServiceCenter omcl file to the corresponding FacilityCenter file.

### Application Called

axces.rm

Parameter	Value	Description
record	\$axces	This field identifies the event in record to be passed—in this case, the value of <i>\$axces</i> .
prompt	evmap in \$axces.register	This field indicates the Map Name used in event registration—in this case, <i>evmap in \$axces.register</i> .
query	foreign.id=2 in evlist in \$axces	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
string1	ocml	Identifies the ServiceCenter file in which data is recorded—in this case, <i>ocml</i> .
boolean1	true	This logical field flag ( <i>true/false</i> ) indicates whether event out record should be written upon transaction completion. In this case, the value is <i>true</i> .

## ScFcOrderLine

An output event for ServiceCenter/FacilityCenter integration that calls the *axcel.write* application.



## TcScCompDel

An input event that allows ServiceCenter and TeleCenter to integrate data from the ServiceCenter company file to the corresponding TeleCenter file.

### Application Called

axces.database

Parameter	Value	Description
record	\$axces	This field identifies the event in record to be passed—in this case, the value of <i>\$axces</i> .
prompt	TcScCompDel	This field indicates the Map Name used in event registration—in this case, <i>evmap in TcScCompDel</i> .
string1	company	Identifies the ServiceCenter file in which data is recorded—in this case, <i>company</i> .
text	delete	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>delete</i> .
query	customer.id=1 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
condition,1	true	This logical field flag ( <i>true/false</i> ) indicates whether the selected record should be deleted. In this case, the value is <i>true</i> .

## TcScCompany

An input event that allows ServiceCenter and TeleCenter to integrate data from the ServiceCenter company file to the corresponding TeleCenter file.

### Application Called

axces.database

Parameter	Value	Description
record	\$axces	This field identifies the event in record to be passed—in this case, the value of <i>\$axces</i> .
prompt	TcScCompany	This field indicates the Map Name used in event registration—in this case, <i>TcScCompany</i> .

Parameter	Value	Description
string1	company	Identifies the ServiceCenter file in which data is recorded—in this case, <i>company</i> .
text	add	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>add</i> .
query	customer.id= 1 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	false	This logical field flag ( <i>true/false</i> ) indicates whether <i>eventout</i> record should be written upon transaction completion. In this case, the value is <i>false</i> .
cond.input	true	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether to process additional records. In this case, the value is <i>true</i> .
name	company	Specifies the Format Control record to use—in this case, <i>company</i> .

## TcScContacts

An input event that allows ServiceCenter and TeleCenter to integrate data from the ServiceCenter contacts file to the corresponding TeleCenter file.

### Application Called

axces.database

Parameter	Value	Description
record	\$axces	This field identifies the <i>eventin</i> record to be passed—in this case, the value of <i>\$axces</i> .
prompt	TcScContacts	This field indicates the Map Name used in event registration—in this case, <i>TcScContacts</i> .
string1	contacts	Identifies the ServiceCenter file in which data is recorded—in this case, <i>contacts</i> .
text	add	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>add</i> .

Parameter	Value	Description
query	contact.name =1 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	false	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. In this case, the value is <i>false</i> .
cond.input	true	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether to process additional records. In this case, the value is <i>true</i> .
name	contacts	Specifies the Format Control record to use—in this case, <i>contacts</i> .

## TcScDept

An input event that allows ServiceCenter and TeleCenter to integrate data from the ServiceCenter department file to the corresponding TeleCenter file.

### Application Called

axces.database

Parameter	Value	Description
record	\$axces	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
prompt	TcScDept	This field indicates the Map Name used in event registration—in this case, <i>TcScDept</i> .
string1	dept	Identifies the ServiceCenter file in which data is recorded—in this case, <i>dept</i> .
text	add	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>add</i> .
query	dept.id =2 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	false	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. In this case, the value is <i>false</i> .

Parameter	Value	Description
cond.input	true	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether to process additional records. In this case, the value is <i>true</i> .
name	dept	Specifies the Format Control record to use—in this case, <i>dept</i> .

## TcScDeptDel

An input event that allows ServiceCenter and TeleCenter to integrate data from the ServiceCenter department file to the corresponding TeleCenter file.

### Application Called

axces.database

Parameter	Value	Description
record	\$axces	This field identifies the event in record to be passed—in this case, the value of <i>\$axces</i> .
prompt	TcScDeptDel	This field indicates the Map Name used in event registration—in this case, <i>TcScDeptDel</i> .
string1	dept	Identifies the ServiceCenter file in which data is recorded—in this case, <i>dept</i> .
text	delete	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>delete</i> .
query	dept=1 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
condition,1	true	This logical field flag ( <i>true/false</i> ) indicates whether the selected record should be deleted. In this case, the value is <i>true</i> .

## TcScDeptdel

An input event that allows ServiceCenter and TeleCenter to integrate data from the ServiceCenter department file to the corresponding TeleCenter file.

### Application Called

axces.database

Parameter	Value	Description
record	\$axces	This field identifies the event in record to be passed—in this case, the value of <i>\$axces</i> .
prompt	deptdel	This field indicates the Map Name used in event registration—in this case, <i>deptdel</i> .
string1	dept	Identifies the ServiceCenter file in which data is recorded—in this case, <i>dept</i> .
text	delete	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>delete</i> .
query	dept=1 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
condition,1	true	This logical field flag ( <i>true/false</i> ) indicates whether the selected record should be deleted. In this case, the value is <i>true</i> .

## TcScLocation

An input event that allows ServiceCenter and TeleCenter to integrate data from the ServiceCenter location file to the corresponding TeleCenter file.

### Application Called

axces.database

Parameter	Value	Description
record	\$axces	This field identifies the event in record to be passed—in this case, the value of <i>\$axces</i> .
prompt	TcScLocDel	This field indicates the Map Name used in event registration—in this case, <i>TcScLocDel</i> .
string1	location	Identifies the ServiceCenter file in which data is recorded—in this case, <i>location</i> .

Parameter	Value	Description
text	delete	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>delete</i> .
query	location.full.name=1 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
condition,1	true	This logical field flag ( <i>true/false</i> ) indicates whether the selected record should be deleted. In this case, the value is <i>true</i> .

## TcScLocation

An input event that allows ServiceCenter and TeleCenter to integrate data from the ServiceCenter location file to the corresponding TeleCenter file.

### Application Called

axces.database

Parameter	Value	Description
record	\$axces	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
prompt	TcScLocation	This field indicates the Map Name used in event registration—in this case, <i>TcScLocation</i> .
string1	location	Identifies the ServiceCenter file in which data is recorded—in this case, <i>location</i> .
text	add	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>add</i> .
query	location = 2 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	false	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. In this case, the value is <i>false</i> .

Parameter	Value	Description
cond.input	true	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether to process additional records. In this case, the value is <i>true</i> .
name	location	Specifies the Format Control record to use—in this case, <i>location</i> .

## WMI

This is an output event.

### Application Called

axces.write

Parameters	Value	Description
record	\$axces.	This field writes the unique record identifier (for example, problem number) to the event in record. If a variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
name	WMI	Specifies the Format Control record to use—in this case, <i>WMI</i> .
string1	^	Identifies the text delimiter to use—in this case, <i>^</i> .
query	evuser in \$axces	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
prompt	nullsub(evurseq in \$axces, evsysseq in \$axces)	This field indicates the Map Name used in event registration—in this case, <i>nullsub(evurseq in \$axces, evsysseq in \$axces)</i> .

## WMI

This is an input event.

### Application Called

wmi.inventory.check

Parameter	Value	Description
record	\$axces	This field identifies the event in record to be passed—in this case, the value of <i>\$axces</i> .
prompt	evmap in \$axces.register	This field indicates the Map Name used in event registration—in this case, <i>evmap in \$axces.register</i> .
string1	device	Identifies the ServiceCenter file in which data is recorded—in this case, <i>device</i> .
text	add	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>add</i> .
query	\$L.temp.query	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	true	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. In this case, the value is <i>true</i> .
name	icma	Identifies the event registration name— <i>icma</i> in this case.
cond.input	val("true", 4)	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether to process additional records. In this case, the value is <i>val("true", 4)</i> .



## XIND

This is an input event.

### Application Called

scauto.inventory

Parameter	Value	Description
record	\$axces	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
prompt	evmap in \$axces.register	This field indicates the Map Name used in event registration—in this case, <i>evmap in \$axces.register</i> .
string1	device	Identifies the ServiceCenter file in which data is recorded—in this case, <i>device</i> .
text	add	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>add</i> .
query	logical.name=7 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	true	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. In this case, the value is <i>true</i> .
name	icma	Identifies the event registration name—in this case, <i>icma</i> (the legacy name).

## approval

An output event that sends approvals for Request Management and Change Management.

### Application Called

axces.write

## approval

An input event that processes approvals for Request Management and Change Management.

### Application Called

es.approval

Parameter	Value	Description
record	\$axces	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
text	ApprovalLog	Identifies the ServiceCenter file in which data is recorded—in this case, <i>ApprovalLog</i> .
name	evmap in \$axces.register	Indicates the Map Name used in event registration—in this case, <i>evmap in \$axces.register</i> .
cond.input	false	This logical field ( <i>true/false</i> ) specifies whether eventout record should be written upon transaction completion. In this case, the value is <i>false</i> .

## cm3rin

This is used for all incoming change events.

### Application Called

axces.cm3

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If an variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name using in event registration—in this case, <i>evmap</i> .
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> , <i>close</i> , <i>approve</i> , <i>disapprove</i> or <i>unapprove</i> .—3 in <i>evlist</i> in <i>\$axces</i> is used in this case.

Parameters	Description
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion, using the same name as the input event with <i>ac</i> appended. The value is <i>true</i> in this case.
string1	Identifies the area of CM3 to work in and where to record data for branching or changing tasks (cm3r or cm3t)— <i>cm3r</i> in this case.
query	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
description	If the event scheduler is unable to obtain a lock for this change record, it will reschedule the event for a time in the future. This field allows us to identify what that interval should be. '00:10:00' is used to indicate 10 minutes.

## cm3rinac

This is sent if the write event out is set to *true*. Calls the following application:

axces.write

**Note:** This returns failed events so the calling application is notified an error has occurred.

## cm3rout

This is created when a cm3 message fires and *cm3rout* is entered into the **axces.out** field. Calls the following application:

axces.write

## cm3tin

This is used for all incoming change tasks.

### Application Called

axces.cm3

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If an variable like \$axces is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name using in event registration— <i>evmap</i> in \$axces.register in this case.
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> , <i>close</i> , <i>approve</i> , <i>disapprove</i> or <i>unapprove</i> — <i>3</i> in <i>evlist</i> in \$axces is used in this case.
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion, using the same name as the input event with <i>ac</i> appended. The value is <i>true</i> in this case.
string1	Identifies the area of CM3 to work in and where to record data for branching or changing tasks (cm3r or cm3t)— <i>cm3t</i> in this case.
query	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
description	If the event scheduler is unable to obtain a lock for this change record, it will reschedule the event for a time in the future. This field allows us to identify what that interval should be. '00:10:00' is used to indicate 10 minutes.

## cm3tinac

This is sent if the write event out is set to *true*. Calls the following application:

axces.write

**Note:** This returns failed events so the calling application is notified an error has occurred.

## cm3tout

This is created when a cm3 message fires and *cm3tout* is entered into the *axces.out* field. Calls the following application:

```
axces.write
```

## dbadd

Adds an item to a specified ServiceCenter file if filter criteria are satisfied; updates if item already exists.

### Application Called

```
axces.database
```

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If an variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name using in event registration—in this case, <i>scauto test</i> .
string1	Identifies the ServiceCenter file in which to record data—in this case, <i>scautotest</i> .
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> , <i>close</i> or <i>delete</i> —in this case, <i>add</i> .
query	The query string to select the specific item is located in this field, for example, <i>field.1=1</i> in <i>\$axces.fields</i> .
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. In this case, the value is <i>true</i> .
cond.input	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether or not to process additional records. The value is <i>true</i> in this case.
name	Specifies the Format Control record to use— <i>scautotest</i> in this case.

## dbdel

Deletes an item from a specified ServiceCenter file if filter criteria are satisfied.

### Application Called

axces.database

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If an variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name using in event registration— <i>scauto test</i> in this case.
string1	Identifies the file in ServiceCenter where to record data— <i>scautotest</i> in this case.
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> , <i>close</i> or <i>delete</i> — <i>delete</i> is used in this case.
query	The query string to select the specific item is located in this field, for example, <i>field.1=1</i> in <i>\$axces.fields</i>
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. The value is <i>true</i> in this case.
cond.input	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether or not to process additional records. The value is <i>true</i> in this case.
condition,1	This logical field flag ( <i>true/false</i> ) indicates whether the selected record(s) should be removed from the database, (rather than updated with a deleted status in the <i>estatus</i> field)— <i>false</i> in this case
name	Specifies the Format Control record to use— <i>scautotest</i> in this case.

## dbupd

Updates an item in a specified ServiceCenter file if filter criteria are satisfied.

### Application Called

axces.database

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If an variable like \$axces is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name using in event registration;— <i>scauto test</i> in this case.
string1	Identifies the file in ServiceCenter where to record data— <i>scautotest</i> in this case.
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> or <i>delete</i> — <i>update</i> is used in this case.
query	The query string to select the specific item is located in this field, for example, <i>field.1=1</i> in <i>\$axces.fields</i>
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. The value is <i>true</i> in this case.
cond.input	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether or not to process additional records. The value is <i>true</i> in this case.
name	Specifies the Format Control record to use— <i>scautotest</i> in this case.

## email

This is an output event and the standard interface to convert ServiceCenter mail to standard email format.

### Application Called

axces.email

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If an variable like <i>\$mail</i> is used, it needs to be setup using Format Control to create the intended result.
text	Indicates the delimiter character used—^ in this case.

## email

This is an input event and the standard interface to receive external email and convert to ServiceCenter mail.

### Application Called

axces.email.receive

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If an variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.



## epmc

This is an input event that uses the **e problem close** map to initiate the problem close process associated with the Get.It! interface.

### Application Called

axces.apm

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the <code>eventin</code> record. If an variable like <code>\$axces</code> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name using in event registration— <code>evmap</code> in <code>\$axces.register</code> in this case.
string1	Identifies the file in ServiceCenter where to record data— <code>probsummary</code> in this case.
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> or <i>delete</i> — <i>close</i> is used in this case.
query	The query string to select the specific item is located in this field, for example, <code>\$ax.query.passed</code>
boolean1	This logical field flag ( <i>true/false/conditional statement</i> ) indicates whether <code>eventout</code> record should be written upon transaction completion. The value is the conditional statement <code>nullsub(evstatus in \$axces, "close")~#"error"</code> in this case.

## epmc

This is an output event that uses the **e problem close** map to write out that a problem has been closed in association with the Get.It! interface.

### Application Called

axces.write

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the <code>eventin</code> record. If an variable like <code>\$axces</code> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name using in event registration— <code>nullsub(evuserseq in \$axces, evsysseq in \$axces)</code> in this case.
string1	Identifies the file in ServiceCenter where to record data— <code>^</code> in this case.
query	The query string to select the specific item is located in this field, for example, <code>evuser in \$axces</code>
name	Specifies the Format Control record to use— <code>pmc</code> in this case.

## epmo

This is an input event that uses the **e problem open** map to initiate the problem open process associated with the Get.It! interface.

### Application Called

axces.apm

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the <code>eventin</code> record. If an variable like <code>\$axces</code> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name using in event registration— <code>evmap in \$axces.register</code> in this case.

Parameters	Description
string1	Identifies the file in ServiceCenter where to record data; <b>probsummary</b> in this case.
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> or <i>delete</i> — <i>open</i> is used in this case.
query	The query string to select the specific item is located in this field, for example, <i>\$ax.query.passed</i>
boolean1	This logical field flag ( <i>true/false/conditional statement</i> ) indicates whether eventout record should be written upon transaction completion. The value is the conditional statement <i>nullsub(evstatus in \$axces,"")~#"error"</i> in this case.
cond.input	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether or not to process additional records; the value is the conditional statement <i>\$ax.open.flag</i> in this case.

## epmo

This is an output event that uses the **e problem open** map to write out that a problem has been opened in association with the Get.It! interface.

### Application Called

axces.write

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If an variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name using in event registration— <i>nullsub(evuserseq in \$axces, evsysseq in \$axces)</i> in this case.
string1	Identifies the file in ServiceCenter where to record data— $\wedge$ in this case.
query	The query string to select the specific item is located in this field, for example, <i>evuser in \$axces</i>
name	Specifies the Format Control record to use— <i>pmo</i> in this case.

## epmosmu

An input event that opens a problem from a call.

### Application Called

axces.apm.epmosu

Parameter	Value	Description
record	\$axces	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
prompt	evmap in \$axces.register	This field indicates the Map Name used in event registration—in this case, <i>evmap in \$axces.register</i> .
string1	probsummary	Identifies the ServiceCenter file in which data is recorded—in this case, <b>probsummary</b> .
text	open	Provides the specific action to take, may be <i>open</i> , <i>reopen</i> , or <i>close</i> —in this case, <i>open</i> .
query	\$ax.query.passed	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	nullsub(evstatus in \$axces,"")~#"error"	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion.
cond.input	\$ax.open.flag	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether to process additional records.

## epmosmu

An output event that writes out after an incident ticket is opened from a call.

### Application Called

axces.write

Parameter	Value	Description
record	\$axces	This field writes the unique record identifier (for example, problem number) to the eventin record. If a variable like \$axces is used, it needs to be set up using Format Control to create the intended result.
name	pmo	Identifies the Format Control record used—in this case, <i>pmo</i> .
string1	^	Identifies the text delimiter to use—in this case, <i>^</i> .
query	evuser in \$axces	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
prompt	nullsub(evusrseq in \$axces, evsysseq in \$axces)	This field indicates the Map Name used in event registration—in this case, <i>nullsub(evusrseq in \$axces, evsysseq in \$axces)</i> .

## epmu

This is an input event that uses the **e problem update** map to initiate the problem update process associated with the Get.It! interface.

### Application Called

axces.apm

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If an variable like \$axces is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name using in event registration— <i>evmap in \$axces.register</i> in this case.

Parameters	Description
string1	Identifies the file in ServiceCenter where to record data— <i>probsummary</i> in this case.
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> or <i>delete</i> — <i>update</i> is used in this case.
query	The query string to select the specific item is located in this field, for example, <i>\$ax.query.passed</i>
boolean1	This logical field flag ( <i>true/false/conditional statement</i> ) indicates whether eventout record should be written upon transaction completion. The value is the conditional statement <i>nullsub(evstatus in \$axces, "update")~#"error"</i> in this case.

## epmu

This is an output event that uses the **e** problem update map to write out that a problem has been updated in association with the Get.It! interface.

### Application Called

*axces.write*

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the <i>eventin</i> record. If an variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name using in event registration— <i>nullsub(evuserseq in \$axces, evsysseq in \$axces)</i> in this case.
string1	Identifies the file in ServiceCenter where to record data— <i>^</i> in this case.
query	The query string to select the specific item is located in this field, for example, <i>evuser in \$axces</i>
name	Specifies the Format Control record to use— <i>pmu</i> in this case.

## esmin

An input event that opens a call in Service Management.

### Application Called

axces.sm

Parameter	Value	Description
record	\$axces	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
prompt	evmap in \$axces.register	This field indicates the Map Name used in event registration—in this case, <i>evmap in \$axces.register</i> .
string1	incidents	Identifies the ServiceCenter file in which data is recorded—in this case, <i>incidents</i> .
query	\$ax.query.passed	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	true	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion—in this case, <i>true</i> .
text	esmin	Designates the output event type—in this case, <i>esmin</i> .

## esmin

An output event that writes out once a call is opened in Service Management.

### Application Called

axces.write

Parameter	Value	Description
record	\$axces	This field writes the unique record identifier (for example, problem number) to the eventin record. If a variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
name	smout	Identifies the Format Control record used—in this case, <i>smout</i> .

Parameter	Value	Description
string1	^	Identifies the text delimiter to use—in this case, ^.
query	evuser in \$axces	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
prompt	nullsub(evusrseq in \$axces, evsysseq in \$axces)	This field indicates the Map Name used in event registration—in this case, <i>nullsub(evusrseq in \$axces, evsysseq in \$axces)</i> .

## gie

Generic Input Event (GIE), used with AssetCenter and ServiceCenter. Calls no application.



## icma

Adds an inventory item to the device file if filter criteria are satisfied; updates if device already exists.

### Application Called

scauto.inventory

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If a variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name used in event registration— <i>evmap</i> in <i>\$axces.register</i> in this case.
string1	Identifies the file in ServiceCenter inventory where data is recorded— <i>device</i> in this case.
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> or <i>delete</i> — <i>add</i> is used in this case.
query	The query string to select specific items or files is placed in this field; for example, to select the device using network name — <i>network.name=5</i> in <i>\$axces.fields</i>
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. The value is <i>true</i> in this case.
name	Event registration name— <i>icma</i> in this case

## icmd

Marks an inventory item for deletion if filter criteria are satisfied by placing *inactive* in the status field.

### Application Called

scauto.inventory

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the <code>eventin</code> record. If a variable like <code>\$axces</code> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name used in event registration— <code>evmap</code> in <code>\$axces.register</code> in this case.
string1	Identifies the file in ServiceCenter inventory where data is recorded— <code>device</code> in this case.
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> or <i>delete</i> — <i>delete</i> is used in this case.
query	The query string to select specific items or files is placed in this field, for example, to select the device using network name — <code>nullsub("network.name=\""+1 in \$axces.fields+"\" or logical.name=\""+1 in \$axces.fields+"\", "false")</code> in this case
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether <code>eventout</code> record should be written upon transaction completion. The value is <i>true</i> in this case.
name	Event registration name— <i>icmd</i> in this case

## icmswa

Adds an inventory item discovered by ServerView or StationView to the device file if filter criteria are satisfied; updates if device already exists.

### Application Called

axces.pcfiles

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the <code>eventin</code> record. If a variable like <code>\$axces</code> is used, it needs to be set up using Format Control to create the intended result.
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> or <i>delete</i> — <i>add</i> is used in this case.
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether or not <code>eventout</code> record should be written upon transaction completion. The default value is <i>false</i> .
name	Specifies the Format Control record to use— <i>pc.files</i> in this case.

## icmswd

Marks a software inventory item discovered by StationView or ServerView for deletion in the *pcfiles* file if filter criteria are satisfied.

### Application Called

axces.pcfiles

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the <code>eventin</code> record. If a variable like <code>\$axces</code> is used, it needs to be set up using Format Control to create the intended result.
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> or <i>delete</i> — <i>update</i> is used in this case.

Parameters	Description
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. The default value is <i>false</i> .
name	Specifies the Format Control record to use— <i>pc.files</i> in this case.

## icmu

Updates an inventory item if filter criteria are satisfied.

### Application Called

scauto.inventory

Parameter	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If a variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name used in event registration— <i>evmap</i> in <i>\$axces.register</i> in this case.
string1	Identifies the file in ServiceCenter inventory where data is recorded— <i>device</i> in this case.
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> or <i>delete</i> — <i>update</i> is used in this case.
query	The query string to select specific items or files is placed in this field, for example, to select the device using network name — <i>network.name=5</i> in <i>\$axces.fields</i>
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. The value is <i>true</i> in this case.
name	Event registration name— <i>icmu</i> in this case

## mlbcm3tc

An input event that allows Peregrine Mobile to access Change Management.

### Application Called

axces.cm3

Parameter	Value	Description
string1	cm3t	Identifies the ServiceCenter file in which data is recorded—in this case, <code>cm3t</code> .
record	<code>\$axces</code>	This field identifies the eventin record to be passed—in this case, the value of <code>\$axces</code> .
prompt	evmap in <code>\$axces.register</code>	This field indicates the Map Name used in event registration—in this case, <code>evmap</code> in <code>\$axces.register</code> .
text	3 in evlist in <code>\$axces</code>	Provides the specific action to take, may be <i>add</i> , <i>update</i> , <i>close</i> , or <i>reopen</i> —in this case, <i>3</i> in <i>evlist</i> in <code>\$axces</code> .
boolean1	true	This logical field flag ( <i>true/false</i> ) indicates whether <code>eventout</code> record should be written upon transaction completion. In this case, the value is <i>true</i> .
query		Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
description	'00:10:00'	If the event scheduler is unable to obtain a lock for this change record, it will reschedule the event in the time interval you specify in this field—in this case, '00:10:00' (in 10 minutes).

## mlbcm3tu

An input event that allows Peregrine Mobile to access Change Management.

### Application Called

axces.cm3

Parameter	Value	Description
string1	cm3t	Identifies the ServiceCenter file in which data is recorded—in this case, <code>cm3t</code> .
record	<code>\$axces</code>	This field identifies the eventin record to be passed—in this case, the value of <code>\$axces</code> .
prompt	evmap in <code>\$axces.register</code>	This field indicates the Map Name used in event registration—in this case, <i>evmap</i> in <code>\$axces.register</code> .
text	3 in evlist in <code>\$axces</code>	Provides the specific action to take, may be <i>add</i> , <i>update</i> , <i>close</i> , or <i>reopen</i> —in this case, <i>3</i> in <i>evlist</i> in <code>\$axces</code> .
boolean1	true	This logical field flag ( <i>true/false</i> ) indicates whether <code>eventout</code> record should be written upon transaction completion. In this case, the value is <i>true</i> .
query		Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
description	'00:10:00'	If the event scheduler is unable to obtain a lock for this change record, it will reschedule the event in the time interval you specify in this field—in this case, '00:10:00' (in 10 minutes).

## mlbocmlc

This is an input event that provides access to Request Management.

### Application Called

axces.rm

Parameter	Value	Description
record	\$axces	This field identifies the event in record to be passed—in this case, the value of <i>\$axces</i> .
text	3 in evlist in \$axces	Provides the specific action to take, may be <i>open</i> , <i>update</i> , or <i>close</i> —in this case, <i>3 in evlist in \$axces</i> .
prompt	evmap in \$axces.register	This field indicates the Map Name used in event registration—in this case, <i>evmap in \$axces.register</i> .
query	number=1 in evlist in \$axces	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
string1	ocml	Identifies the ServiceCenter file in which data is recorded—in this case, <i>ocml</i> .

## mlbocmlu

This is an input event that allows Peregrine Mobile to access Request Management.

### Application Called

axces.rm

Parameter	Value	Description
record	\$axces	This field identifies the event in record to be passed—in this case, the value of <i>\$axces</i> .
text	3 in evlist in \$axces	Provides the specific action to take, may be <i>open</i> , <i>update</i> , or <i>close</i> —in this case, <i>3 in evlist in \$axces</i> .
prompt	evmap in \$axces.register	This field indicates the Map Name used in event registration—in this case, <i>evmap in \$axces.register</i> .
query	number=1 in evlist in \$axces	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
string1	ocml	Identifies the ServiceCenter file in which data is recorded—in this case, <i>ocml</i> .



## mblpmc

An input event that allows Peregrine Mobile to close a problem.

### Application Called

axces.apm

Parameter	Value	Description
record	\$axces	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
prompt	evmap in \$axces.register	This field indicates the Map Name used in event registration—in this case, <i>evmap in \$axces.register</i> .
string1	probsummary	Identifies the ServiceCenter file in which data is recorded—in this case, <b>probsummary</b> .
text	close	Provides the specific action to take, may be <i>open</i> , <i>update</i> , <i>reopen</i> , or <i>close</i> —in this case, <i>close</i> .
query	\$ax.query.passed	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	nullsub(evstatus in \$axces, "close")~#"error"	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion.

## mblpmo

An input event that allows Peregrine Mobile to open a problem.

### Application Called

axces.apm

Parameter	Value	Description
record	\$axces	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
prompt	evmap in \$axces.register	This field indicates the Map Name used in event registration—in this case, <i>evmap in \$axces.register</i> .
string1	probsummary	Identifies the ServiceCenter file in which data is recorded—in this case, <b>probsummary</b> .
text	open	Provides the specific action to take, may be <i>open</i> , <i>update</i> , <i>reopen</i> , or <i>close</i> —in this case, <i>open</i> .
query	\$ax.query.passed	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	nullsub(evstatus in \$axces,"")~#"error"	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion.
cond.input	\$ax.open.flag	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether to process additional records.

## mblpmu

An input event that allows Peregrine Mobile to update a problem.

### Application Called

axces.apm

Parameter	Value	Description
record	<code>\$axces</code>	This field identifies the eventin record to be passed—in this case, the value of <code>\$axces</code> .
prompt	<code>evmap in \$axces.register</code>	This field indicates the Map Name used in event registration—in this case, <code>evmap in \$axces.register</code> .
string1	<code>probsummary</code>	Identifies the ServiceCenter file in which data is recorded—in this case, <code>probsummary</code> .
text	<code>update</code>	Provides the specific action to take, may be <code>open</code> , <code>update</code> , <code>reopen</code> , or <code>close</code> —in this case, <code>update</code> .
query	<code>\$ax.query.passed</code>	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	<code>nullsub(evstatus in \$axces,"update")~#"error"</code>	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion.

## opera

Adds a new user to ServiceCenter if filter criteria are satisfied; updates if user already exists.

### Application Called

axces.database

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If a variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name used in event registration— <i>operator</i> in this case.
string1	Identifies the file in ServiceCenter inventory where data is recorded— <i>operator</i> in this case.
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> or <i>delete</i> — <i>add</i> is used in this case.
query	The query string to select specific items or files is placed in this field, for example, to select the operator— <i>name=1</i> in <i>\$axces.fields</i>
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. The value is <i>true</i> in this case.
cond.input	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether or not to process additional records. The value is <i>false</i> in this case.
name	Specifies the Format Control record to use— <i>operator.scauto</i> in this case.

**Important:** The default query selects for the operator and adds a new user with the minimum privileges to access ServiceCenter: No access to Problem, Change, Inventory, Request or Financial Management.

**Note:** Most organizations establish a template operator record for each class of users (for example, Incident Management) and modify their select query (indicated below) to the name defined for the template operator record.

For example, an operator record named *standarduser* can be set up with Execute Capabilities of Incident Management, Inventory Management, Change Request and Change Task, and OCML, OCMQ and OCMO. This allows non-administrative access to Incident, Inventory, Change and Request Management respectively. The query parameter would be changed from `name=1 in $axces.fields` to `name="standarduser"`

## operd

Deletes a user from ServiceCenter, if filter criteria are satisfied.

### Application Called

axces.database

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If a variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name used in event registration— <i>operator</i> in this case.
string1	Identifies the file in ServiceCenter inventory where data is recorded— <i>operator</i> in this case.
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> or <i>delete</i> — <i>delete</i> is used in this case.
query	The query string to select specific items or files is placed in this field, for example, to select the operator— <i>name=1 in \$axces.fields</i> .
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. The value is <i>true</i> in this case.
cond.input	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether or not to process additional records. The value is <i>false</i> in this case.

Parameters	Description
condition,1	This logical field flag ( <i>true/false</i> ) indicates whether the selected record(s) should be removed from the database, (rather than updated with a deleted status in the <i>estatus</i> field). <i>true</i> is used in this case
name	Specifies the Format Control record to use— <i>operator.scauto</i> in this case.

## operu

Updates an item in a specified ServiceCenter file if filter criteria are satisfied.

### Application Called

axces.database

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If a variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name used in event registration— <i>operator</i> in this case.
string1	Identifies the file in ServiceCenter inventory where data is recorded— <i>operator</i> in this case.
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> or <i>delete</i> — <i>update</i> is used in this case.
query	The query string to select specific items or files is placed in this field, for example, to select the operator— <i>name=1</i> in <i>\$axces.fields</i> .
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. The value is <i>true</i> in this case.
cond.input	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether or not to process additional records. The value is <i>true</i> in this case.
name	Specifies the Format Control record to use— <i>operator.scauto</i> in this case.

## outageend

### Application Called

axces.outageend

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If a variable like \$axces is used, it needs to be set up using Format Control to create the intended result.

## outagestart

### Application Called

axces.outagestart

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If a variable like \$axces is used, it needs to be set up using Format Control to create the intended result.

## page

Outbound page action that calls the axces.write application.

## pageclose

Uses condition statement (`evfiends in $axces)#"pm"`)

### Application Called

axces.apm

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the <code>eventin</code> record. If a variable like <code>\$axces</code> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name used in event registration— <code>envmap</code> in <code>\$axces.register</code> in this case.
string1	Identifies the file in ServiceCenter where data is recorded— <code>problem</code> in this case.
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> or <i>delete</i> — <i>close</i> is used in this case.
query	The query string to select specific items or files is placed in this field, for example, to select the operator — <code>"number=\""+substr(1 in \$axces.fields, 3, lng(1 in \$axces.fields) - 2)+"\""</code>
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether <code>eventout</code> record should be written upon transaction completion. The value is <i>false</i> in this case.



## pageresp

Updates a problem with an acknowledgment or message received as response to a page. Uses condition statement (`evfiends in $axces)#"pm"`)

### Application Called

axces.apm

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If a variable like <code>\$axces</code> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name used in event registration— <code>envmap</code> in <code>\$axces.register</code> in this case.
string1	Identifies the file in ServiceCenter where data is recorded— <code>problem</code> in this case.
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> or <i>delete</i> — <i>update</i> is used in this case.
query	The query string to select specific items or files is placed in this field, for example, to select the operator— <code>"number=\""+1 in \$axces.fields+"\\"</code>
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. The value is <i>false</i> in this case.

## pcsoftware

An input event that allows desktop inventory products to update ServiceCenter.

### Application Called

axces.database

Parameter	Value	Description
record	\$axces	This field identifies the eventin record to be passed—in this case, the value of <i>\$axces</i> .
prompt	pcsoftware	This field indicates the Map Name used in event registration—in this case, <i>pcsoftware</i> .
string1	pcsoftware	Identifies the ServiceCenter file in which data is recorded—in this case, <i>pcsoftware</i> .
text	add	Provides the specific action to take, may be <i>add</i> , <i>update</i> , or <i>delete</i> —in this case, <i>add</i> .
query	logical.name=20 in \$axces.fields and license.number=2 in \$axces.fields and application.name=1 in \$axces.fields	Indicates the query to be used to find the pre-existing record which should be updated by this inbound event.
boolean1	false	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. In this case, the value is <i>false</i> .
cond.input	true	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether to process additional records. In this case, the value is <i>true</i> .

## pmc

This is an input event that closes a problem ticket if filter criteria is met, using same path as manual close operation.

### Initialization expressions

- `cleanup($ax.query.passed)`
- `if (not null(3 in $axces.fields)) then ($ax.query.passed="number=\"+str(3 in $axces.fields)+"\") else ($ax.query.passed="flag=true and network.name=\"+2 in $axces.fields+"\")`
- `if null($ax.query.passed) then if (not null(20 in $axces.fields)) then ($ax.query.passed="flag=true and reference.no=\"+str(20 in $axces.fields)+"\")`
- `if null($ax.query.passed) then ($ax.query.passed=nullsub("flag=true and network.name=\"+2 in $axces.fields+"\", "false"))`
- `if (index("NAPA", evuser in $axces)>0) then ($ax.query.passed=nullsub("flag=true and logical.name=\"+2 in $axces.fields+"\", "false"))`
- `if (index("IND", evuser in $axces)>0) then ($ax.query.passed=nullsub("flag=true and logical.name=\"+1 in $axces.fields+"\", "false"))`
- `$bypass.failed.validation=true`
- `$axces.bypass.failed.validation=true`

### Application Called

axces.apm

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the <code>eventin</code> record. If a variable like <code>\$axces</code> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name used in event registration— <code>envmap</code> in <code>\$axces.register</code> in this case.
string1	Identifies the file in ServiceCenter where data is recorded— <code>probsummary</code> in this case.

Parameters	Description
text	Provides the specific action to take, may be <i>open</i> , <i>add</i> , <i>update</i> or <i>delete</i> — <i>close</i> is used in this case.
query	The query string used to select existing ticket for update— <i>\$ax.query.passed</i> in this case.
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether <i>eventout</i> record should be written upon transaction completion. The value in this case is: <i>nullsub(evstatus in \$axces, "close")~#"error"</i>

## pmc

This is an output event that writes out after an event is closed.

### Application Called

*axces.write*

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the <i>eventin</i> record. If a variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
name	Specifies Format Control record to use— <i>pmc</i> in this case.
string1	Text delimiter to use. The default is $\wedge$ .
prompt	This field indicates the Map Name used in event registration— <i>nullsub(evusrseq in \$axces, evsysseq in \$axces)</i> in this case
query	The query string to select existing ticket for update— <i>evuser in \$axces</i> in this case.

## pmo

An input event that opens a problem ticket, if filter criteria are satisfied, using same path as manual problem open.

### Initialization expressions

- `$ax.query.passed=nullsub("flag=true and network.name=\""+2 in $axces.fields+"\\"", "false")`
- `if (index("axmail", evuser in $axces)>0) then ($ax.query.passed=nullsub("flag=true and logical.name=\""+1 in $axces.fields+"\\"", "false"))`
- `if (index("NAPA", evuser in $axces)>0) then ($ax.query.passed=nullsub("flag=true and logical.name=\""+1 in $axces.fields+"\\"", "false"))`
- `$ax.open.flag=false`
- `if (index("scnote", evuser in $axces)>0) then ($ax.open.flag=true)`
- `$axces.lock.interval='00:00:30'`
- `if (index("IND", evuser in $axces)>0) then ($ax.query.passed=nullsub("flag=true and logical.name=\""+1 in $axces.fields+"\\"", "false");$ax.open.flag=false)`
- `$bypass.failed.validation=true`
- `$axces.bypass.failed.validation=true`

### Application Called

axces.apm

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If a variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name used in event registration— <i>envmap</i> in <i>\$axces.register</i> in this case
string1	Identifies the file in ServiceCenter where data is recorded— <i>probsummary</i> in this case.
text	Provides the specific action to take, may be <i>open</i> , <i>add</i> , <i>update</i> or <i>delete</i> — <i>open</i> is used in this case.

Parameters	Description
query	The query string to select existing ticket for update— <i>\$ax.query.passed</i> in this case.
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. The value in this case is: <i>nullsub(evstatus in \$axces, "")~#"error"</i>
cond.input	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether or not to process additional records. The value is <i>\$ax.open.flag</i> in this case.

## pmo

This is an output event that writes out after a problem ticket is opened.

### Application Called

axces.write

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If a variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
name	Identifies Format Control record to use— <i>pmo</i> in this case.
string1	Text delimiter to use. The default is ^.
query	The query string to select existing ticket for update— <i>evuser in \$axces</i> in this case
prompt	This field indicates the Map Name used in event registration— <i>nullsub(enusrseq in \$axces, evsysseq in \$axces</i> in this case.

## pmu

An input event that updates a problem ticket if the filter criteria is met, using the same path as a manual update.

### Initialization expressions

- `cleanup($ax.query.passed)`
- `if (not null(3 in $axces.fields)) then ($ax.query.passed="number=\"+str(3 in $axces.fields)+"\") else ($ax.query.passed="flag=true and network.name=\"+2 in $axces.fields+"\")`
- `if null($ax.query.passed) then if (not null(20 in $axces.fields)) then ($ax.query.passed="flag=true and reference.no=\"+str(20 in $axces.fields)+"\")`
- `if null($ax.query.passed) then ($ax.query.passed="false")`
- `$bypass.failed.validation=true`
- `$axces.bypass.failed.validation=true`

### Application Called

`axces.apm`

Parameters	Description
<code>record</code>	This field writes the unique record identifier (for example, problem number) to the eventin record. If a variable like <code>\$axces</code> is used, it needs to be set up using Format Control to create the intended result.
<code>prompt</code>	This field indicates the Map Name used in event registration— <code>envmap</code> in <code>\$axces.register</code> in this case.
<code>string1</code>	Identifies the file in ServiceCenter where data is recorded— <code>probsummary</code> in this case.
<code>text</code>	Provides the specific action to take, may be <i>open</i> , <i>add</i> , <i>update</i> or <i>delete</i> — <i>update</i> is used in this case.
<code>query</code>	The query string to select existing ticket for update— <code>\$ax.query.passed</code> in this case.
<code>boolean1</code>	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. The value in this case is: <code>nullsub(evstatus in \$axces, "update")~#"error"</code>

## pmu

This is an output event that writes out after a problem ticket is updated.

### Application Called

axces.write

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If a variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
name	Identifies Format Control record to use— <i>pmu</i> in this case.
string1	Text delimiter to use. In this case ^.
query	The query string to select existing ticket for update— <i>evuser in \$axces</i> in this case.
prompt	This field indicates the Map Name used in event registration— <i>nullsub(evusrseq in \$axces, evsysseq in \$axces)</i> in this case.

## prgma

Adds a software inventory item discovered by an external agent (other than ServerView or StationView) to the *pcfiles* file if filter criteria are satisfied; updates if item already exists.

### Application Called

axces.software

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If a variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name used in event registration;— <i>software</i> in this case.
string1	Identifies the software file in ServiceCenter inventory where data is recorded— <i>pcfiles</i> in this case.



Parameters	Description
query	The query string to select the device to use for update—in this case using network name: <i>logical.name=1 in \$axces.fields and description=9 in \$axces.fields</i>
name	Specifies the Format Control record to use— <i>pc.files</i> in this case. (optional)
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. The value is <i>false</i> in this case.

## prgmd

Deletes a software item discovered by an external agent (other than ServerView or StationView) from the **pcfiles** file if filter criteria are satisfied. Note that the default is to update a field called **estatus** with *deleted* rather than to remove the record from the database.

### Application Called

axces.software

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If a variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name used in event registration— <i>pcfiles</i> in this case.
string1	Identifies the software file in ServiceCenter inventory where data is recorded— <b>software</b> in this case.
query	The query string to select device to use for update—in this case using network name— <i>logical.name=1 in \$axces.fields and description=9 in \$axces.fields</i>
name	Specifies the Format Control record to use— <i>pc.files</i> in this case. (optional)
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. The value is <i>false</i> in this case.

Parameters	Description
cond.input	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether or not to process additional records. The value is <i>true</i> in this case.
condition,1	This logical field flag ( <i>true/false</i> ) specifies indicates that records should be removed from the database rather than marked for delete. The value in this case is <i>false</i> .

## prgmu

Updates an inventory item discovered by an external agent (other than ServerView or StationView) in the *pcfiles* file if filter criteria are satisfied.

### Application Called

axces.software

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If a variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name used in event registration— <i>pcfiles</i> in this case.
string1	Identifies the software file in ServiceCenter inventory where data is recorded— <i>software</i> in this case.
query	The query string to select device to use for update—in this case using network name: <i>logical.name=1 in \$axces.fields and description=9 in \$axces.fields</i>
name	Specifies the Format Control record to use— <i>pc.files</i> in this case (optional).
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. The value is <i>false</i> in this case.
cond.input	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether or not to process additional records. The value is <i>false</i> in this case.

## rmlin

Request Management line item input.

### Application Called

axces.rm

Parameters	Description
record	This field identifies the eventin record to be passed— <i>\$axces</i> in this case.
prompt	This field indicates the Map Name used in event registration— <i>evmap</i> in <i>\$axces.register</i> in this case.
string1	Identifies the file ( <i>ocmq</i> , <i>ocmo</i> , <i>ocml</i> ) in ServiceCenter where data is recorded— <i>ocml</i> in this case.
text	Provides the specific action to take, may be <i>open</i> , <i>update</i> , <i>close</i> or <i>approve</i> — <i>3</i> in <i>evlist</i> in <i>\$axces</i> is used in this case.
query	The query string to select a pre-existing record to update, approve or close— <i>number=1</i> in <i>evlist</i> in <i>\$axces</i> .

## rmoappr

Request Management order approval.

### Application Called

axces.rm

Parameters	Description
record	This field identifies the eventin record to be passed— <i>\$axces</i> in this case.
prompt	This field indicates the Map Name used in event registration— <i>evmap</i> in <i>\$axces.register</i> in this case.
string1	Identifies the area in ServiceCenter where data is recorded— <i>ocmo</i> in this case.
text	Provides the specific action to take, may be <i>open</i> , <i>update</i> , <i>close</i> or <i>approve</i> — <i>3</i> in <i>evlist</i> in <i>\$axces</i> is used in this case.
query	The query string to select specific items or files is placed in this field, for example, to select the operator— <i>number=1</i> in <i>evlist</i> in <i>\$axces</i>

## rmoin

Request Management order input.

### Application Called

axces.rm

Parameters	Description
record	This field identifies the eventin record to be passed— <i>\$axces</i> in this case.
prompt	This field indicates the Map Name used in event registration—in this case <i>evmap</i> in <i>\$axces.register</i> .
string1	Identifies the file in ServiceCenter where data is recorded; <i>ocmo</i> in this case.
text	Provides the specific action to take, may be <i>open</i> , <i>update</i> , <i>close</i> or <i>approve</i> — <i>3</i> in <i>evlist</i> in <i>\$axces</i> is used in this case.
query	The query string to select a pre-existing record to update, approve or close— <i>number=1</i> in <i>evlist</i> in <i>\$axces</i>

## rmqappr

Request Management quote approval.

### Application Called

axces.rm

Parameters	Description
record	This field identifies the eventin record to be passed—in this case, <i>\$axces</i> .
prompt	This field indicates the Map Name used in event registration; in this case <i>evmap</i> in <i>\$axces.register</i> .
string1	Identifies the area in ServiceCenter where data is recorded; <i>ocmq</i> in this case.
text	Provides the specific action to take, may be <i>open</i> , <i>update</i> , <i>close</i> or <i>approve</i> — <i>3</i> in <i>evlist</i> in <i>\$axces</i> is used in this case.
query	The query string to select a pre-existing record to update, approve or close— <i>\$L.approve.action</i>

## rmqin

Request Management quote input.

### Application Called

axces.rm

Parameters	Description
record	This field identifies the eventin record to be passed.
prompt	This field indicates the Map Name used in event registration; in this case <i>evmap in \$axces.register</i> .
string1	Identifies the file in ServiceCenter where data is recorded; <i>ocmq</i> in this case.
text	Provides the specific action to take, may be <i>open, update, close</i> or <i>approve</i> — <i>3 in evlist in \$axces</i> is used in this case.
query	The query string to select a pre-existing record to update, <i>approve</i> or <i>close</i> — <i>number=1 in evlist in \$axces</i>

## sapordl (1)

An output event. Calls no application.

## sapordl (2)

An input event.

### Application Called

axces.rm

Parameters	Description
record	This field identifies the eventin record to be passed.
prompt	This field indicates the Map Name used in event registration— <i>evmap in \$axces.register</i> in this case.
string1	Identifies the file in ServiceCenter inventory where data is recorded— <i>ocml</i> in this case.
text	Provides the specific action to take, may be <i>open, update, close</i> or <i>approve</i> — <i>update</i> is in this case.

Parameters	Description
query	The query string to select specific items or files is placed in this field, for example, to select the operator— <i>number=12 in evlist in \$axces</i>
name	Identifies the operator (security profile) in ServiceCenter that should be used and what userID should be stamped on records processed via this event— <i>falcon</i> in this case.

## sapqtel (1)

This output event is the outbound quote line item component of the SAPQTE event. It uses this registration to identify the eventmap to be used for message formatting. It calls no application.

## sapqtel (2)

This input event is the detail portion of the SAPQTE event.

### Application Called

axces.rm

Parameters	Description
record	This field identifies the eventin record to be passed.
prompt	This field indicates the Map Name used in event registration— <i>evmap in \$axces.register</i> in this case.
string1	Identifies the file in ServiceCenter inventory where data is recorded— <i>ocml</i> in this case.
text	Provides the specific action to take, may be <i>open</i> , <i>update</i> , <i>close</i> or <i>approve</i> — <i>update</i> is in this case.
query	The query string to select specific items or files is placed in this field, for example, to select the operator— <i>number=12 in evlist in \$axces</i>
name	Identifies the operator (security profile) in ServiceCenter should be used and what userID stamped on records processed via this event— <i>falcon</i> in this case.

## saprecl (1)

This output event is the outbound goods receipt line item component of the SAPQTE event. It uses this registration to identify the eventmap for message formatting use. It calls no application.

## sapreql (1)

This output event is the outbound request line item component of SAPREQ. It uses this registration to identify the eventmap to be used for message formatting.

## sapreql (2)

This input event is the detail portion of SAPREQ event.

### Application Called

axces.rm

Parameters	Description
record	This field identifies the eventin record to be passed.
prompt	This field indicates the Map Name used in event registration— <i>evmap</i> in <i>\$axces.register</i> in this case.
string1	Identifies the file in ServiceCenter inventory where data is recorded— <i>ocml</i> in this case.
text	Provides the specific action to take, may be <i>open</i> , <i>update</i> , <i>close</i> or <i>approve</i> — <i>update</i> is in this case.
query	The query string to select specific items or files is placed in this field, for example, to select the operator— <i>number=20</i> in <i>evlist</i> in <i>\$axces</i>
name	Identifies the operator (security profile) in ServiceCenter should be used and what userID stamped on records processed via this event— <i>falcon</i> in this case.

## slaresponse

### Application Called

axces.postresponse

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the <code>eventin</code> record. If a variable like <code>\$axces</code> is used, it needs to be set up using Format Control to create the intended result.

## smin

Service Management incoming service request or help issue.

### Initialization expressions

- `$ax.query.passed=nullsub("incident.id=\")+1 in $axces.fields+"\", "false")`
- `if (null(1 in $axces.fields) or 1 in $axces.fields="") then ($ax.query.passed="false")`

### Application Called

axces.sm

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the <code>eventin</code> record. If a variable like <code>\$axces</code> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name used in event registration—in this case <i>evmap</i> in <code>\$axces.register</code> .
string1	Identifies the file in ServiceCenter inventory where data is recorded—incidents in this case.



Parameters	Description
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. The value is <i>true</i> in this case.
query	The query string to select specific items or files is placed in this field, for example, to select the operator— <i>\$ax.query.passed</i>

## smout

This is an output event that writes out once an incoming service request or help issue has been entered into the system.

### Application Called

axces.write

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If a variable like <i>\$axces</i> is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name used in event registration— <i>nullsub(evusrseq in \$axces, evsysseq in \$axces)</i> in this case.
name	Identifies the Format Control record to use— <i>smout</i> in this case
string1	Text delimiter to use— <i>^</i> In this case
query	The query string to select specific items or files is placed in this field, for example, to select the operator— <i>evuser in \$axces</i> In this case.

## submit

Submits a job for processing. Calls the **axces.write** application.

## sysbull

Adds a new System Bulletin to ServiceCenter if filter criteria are satisfied.

### Application Called

axces.database

Parameters	Description
record	This field writes the unique record identifier (for example, problem number) to the eventin record. If a variable like \$axces is used, it needs to be set up using Format Control to create the intended result.
prompt	This field indicates the Map Name used in event registration— <i>bulletin</i> in this case.
string1	Identifies the file in ServiceCenter inventory where data is recorded— <i>bulletin</i> in this case.
text	Provides the specific action to take, may be <i>add</i> , <i>update</i> or <i>delete</i> — <i>add</i> is used in this case.
query	The query string to select specific items or files is placed in this field, for example, to select the operator— <i>date=date(val(str(1 in \$axces.fields),3))</i> in this case.
boolean1	This logical field flag ( <i>true/false</i> ) indicates whether eventout record should be written upon transaction completion. The value is <i>true</i> in this case.
cond.input	This logical field ( <i>true/false</i> ) specifies if more than one record is selected, whether or not to process additional records. The value is <i>false</i> in this case.

**Note:** The system bulletin record in use will be the one for today's date (for example, if today is New Year's Day the bulletin will be the one for 01/01/99 00:00) or the one with the default flag set to true.

**Warning:** You should not attempt to modify the application names or parameters unless you are completely familiar with RAD programming.

Event Services is designed to provide a standard interface for user-defined applications as well as those described above. Generally speaking, any RAD application that does not require user I/O can be called as an event services application. Refer to the *SCAuto for NetView OS/390* manual or the *RAD Guide* for more information.

## Global Variables

The following global variables are available for use when defining registration events:

Variable	Description
\$axces	Represents the eventin record.
\$axces.fields	Represents the evlist field in the eventin record.
\$axces.register	Represents the event registration record
\$axces.lock.interval	An interval of time (for example, '00:02:00' for two minutes) after which a retry will occur if the attempt to update a problem has been denied due to a lock.
\$axces.debug	If set to true, the evlist array in the eventin record will not be removed before attempting to update the record. If the size of the record exceeds 32Kbytes an error will be issued, the eventin record will NOT be updated and the event will be reprocessed (since the evtime field will not be removed). Use <b>this feature with discretion</b> .
\$axces.bypass.failed.validation	Used in events calling the application axces.apm. Defaults to "true". If set to true, the application will ignore any failed formatctrl validations. If set to false, the event will be set to status "error-fc".

**Note:** The two additional standard events, **page** and **fax**, are not controlled through the registration table.

- A **fax** event is simply a report that uses the FAX config record's name as its printer name. The report is written to the eventout file and the external SCAuto application directs it as required.
- A **page** event is normally called as a Format Control subroutine based upon conditions at problem open time.

# Generic Event Administration

The controls under this option allow for administration of outgoing event records into Connect.It, including the following:

- Editing eventout information generation
- Export of configuration records
- Export of Database Dictionary structures

To access these event controls:

- 1 Select the Administration tab in the Event Services menu.

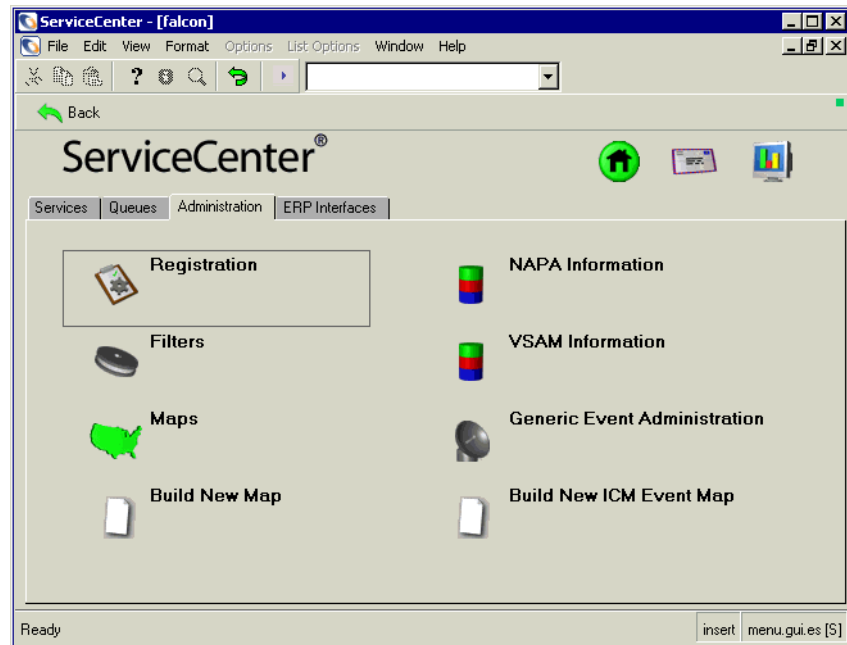
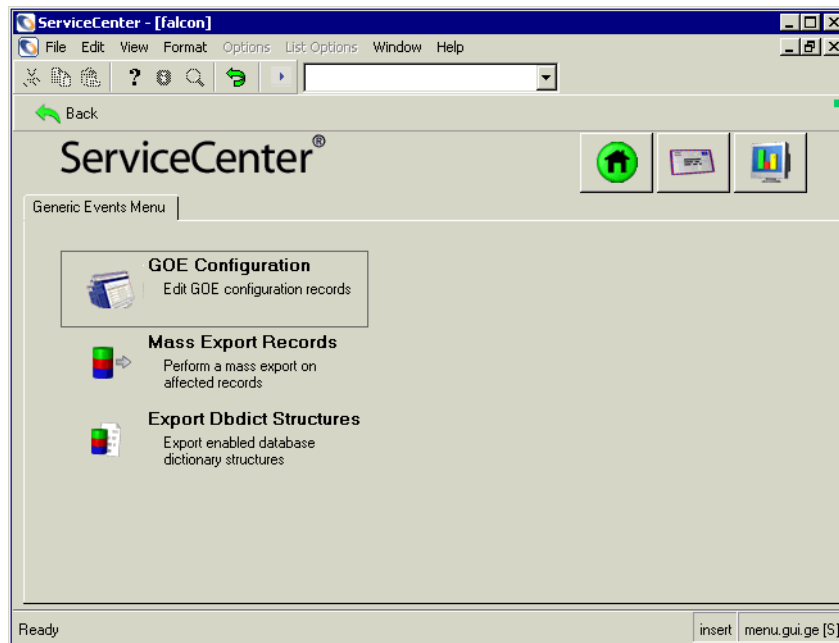


Figure 2-6: Administration tab of the Event Services menu

## 2 Click Generic Event Administration.

The Generic Event Administration menu is displayed.



**Figure 2-7: Generic Event Administration menu**

For details about the Connect.It product, see the Connect.It documentation.



# 3 Mapping and Filtering

## CHAPTER

Once events have been created coming in or out of ServiceCenter, processes need to be set into place to manage and direct the events. Event mapping and event filtering take the event and its constituent data, and direct it in specified ways to create results within other areas of the system.

This chapter divides its discussion of these process into the following sections:

- *Mapping* on page 158
- *Event Filters* on page 180

# Mapping

Event mapping information is stored in the `eventmap` file. There are two types of maps: input maps and output maps. Input maps contain instructions for moving data from the `eventin` record's **External Information String** (*evfields*) field to the target file, while output maps move information from the source file to the `eventout` record's **External Information String** field.

---

**Important:** Event Maps provided with Event Services describe *standard events*. Changing the relative position of data in the information exchanged between ServiceCenter and the external applications (for example, IPAS) may cause standard events to fail. You should create new maps for non-standard events rather than modifying existing maps.

---

## The Event Map Form

To review event maps:

- 1 Access the **Event Services** menu.



- 2 Select the Administration tab.

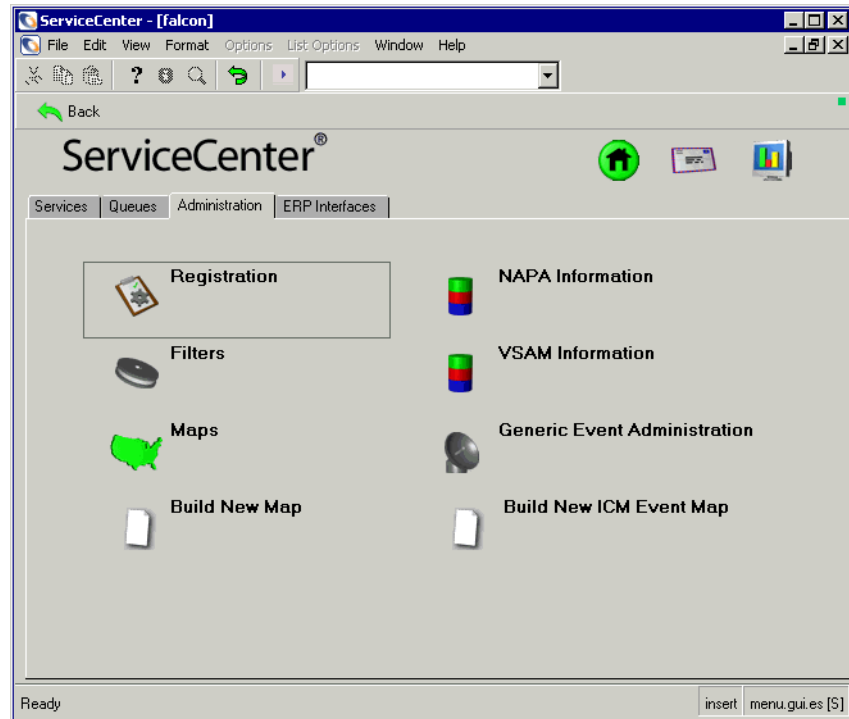


Figure 3-1: Administration tab in the Event Services menu

- 3 Click Maps.

The Event Map form is displayed.

**Figure 3-2: Event Mapping**

## Header Fields

Encoded field input names, as recorded in the eventmap file, have been included in parenthesis for reference only.

Field	Description
Map Name (evmap)	A unique name that identifies each map; combined with the evseq field and the evtype field, comprises the unique key.
Type (evtype)	A flag to identify whether this registration is for an input or an output transaction; only input or output are acceptable values.

Field	Description
Fixed or Variable (evmpty)	Either Fixed Length or Variable Length; indicates the format of data passed in eventin record; default is variable with a delimiter between fields.
Sequence (evseq)	Number indicating the sequence in which data is mapped from the eventin record to the target record; when multiple files are updated, certain dependencies may exist which would necessitate a prescribed order for field mapping; used in icm* maps.
Position (evindex)	Number corresponding to the relative position of data in the eventin record's evfields field.
Length (evlength)	If evmpty is Fixed Length, the length of the field must be provided.

## Basic Tab Fields

Field	Description
File Name (evfile)	Name of the file from (for output) or into (for input) which data will be mapped.
Query (evquery)	Query used to select a record from the named file if the file name is different from the one currently in use (i.e., the sequence number changes); allows update of multiple files with a single map.
Field Name (evfield)	Name of the field from (for output) or into (for input) which data will be mapped.
Nullsub (evnullsub)	Value in this field replaces the contents of the source field if NULL. To keep the value present in a record that is being updated, enter <i>\$axces.field</i> in the Nullsub field. You can set a global condition to keep the value present in a record that is being updated by setting the Use Current Data? condition in the event registration record to true.
Data Type (evdtype)	Data type of the field being mapped; this value is set by the Build Event Maps process and is automatically set when the event map record is being added or updated.  If evdtype is Array, you must complete the appropriate fields in the Array Information section of the form.
Translate (evxlate)	Indicates whether to translate the field value to uppercase (uc) or lowercase (lc); by default no translation is performed.
Element Type (evetype)	Data type of array elements.  If evetype is Structure, you must enter a <i>different</i> separator for the evsepchar and evsepchar.struc fields. If evetype contains a value <i>other</i> than Structure, you must enter a value for either the evsepchar or the evitmlng field.
Element Separator (evsepchar)	Separation character to use for elements in array-type fields; default is   (pipe symbol).
Element Length (evitmlng)	If not NULL, defines the length of each element in array type fields.  <b>Note:</b> This field does not apply if evetype is Structure.
Element Separator (structure) (evsepchar.struc)	Separation character to use for the subelements within the structure of an array of structures; the default is ` (grave accent).



## Expressions Tab Fields

The screenshot shows a configuration window with two tabs: 'Basics' and 'Expressions'. The 'Expressions' tab is active and contains the following sections:

- Initialization:** A section with four horizontal text input fields.
- Condition for Mapping:** A section with one horizontal text input field.
- Post-Map Instructions:** A section with five horizontal text input fields.

Figure 3-4: Expressions tab in the Event Map

Field	Description
Initialization (evinit)	Array of statements that are executed at run time to initialize variables or initiate action based upon the contents of the data passed in the <code>eventin</code> record and/or on global variables available at run time; the global variable <code>\$axces.fields</code> is used to represent an array of the fields passed in the <code>evfield</code> field of the <code>eventin</code> record.
Condition for Mapping (evmapcond)	Condition that, if true, will allow the data to be mapped.
Post-Map Instructions (evcalc)	Array of expressions that are evaluated at run time to execute processing statements after field has been mapped.

## Using Event Maps

Each record in the `eventmap` file describes a single field. Event Services uses this information to map data from external sources to ServiceCenter files, and data in ServiceCenter files to a sequence of delimited fields for export to external applications.

For example, when a ServiceCenter user sends mail, certain fields in the ServiceCenter mail file are populated. These include **user.to**, **user.from**, **user.array**, **subject** and **text**. When email is sent, the information in these fields must be mapped in a standard, defined sequence so that the SCAuto mail application can translate it to external programs. Likewise, when SCAuto receives mail from an external program and posts it to the eventin file, the Event Services application populates the appropriate fields in the ServiceCenter mail file.

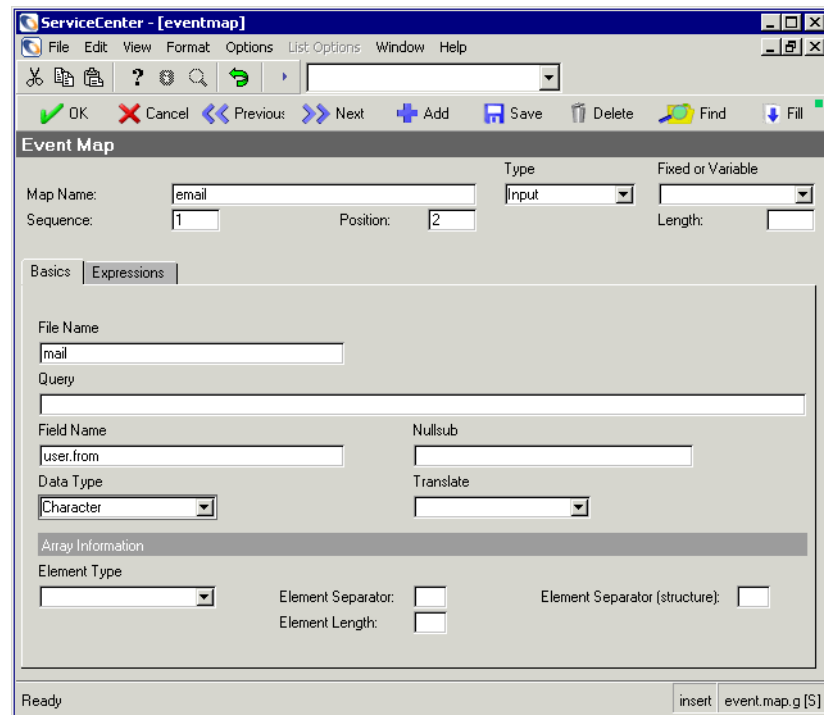
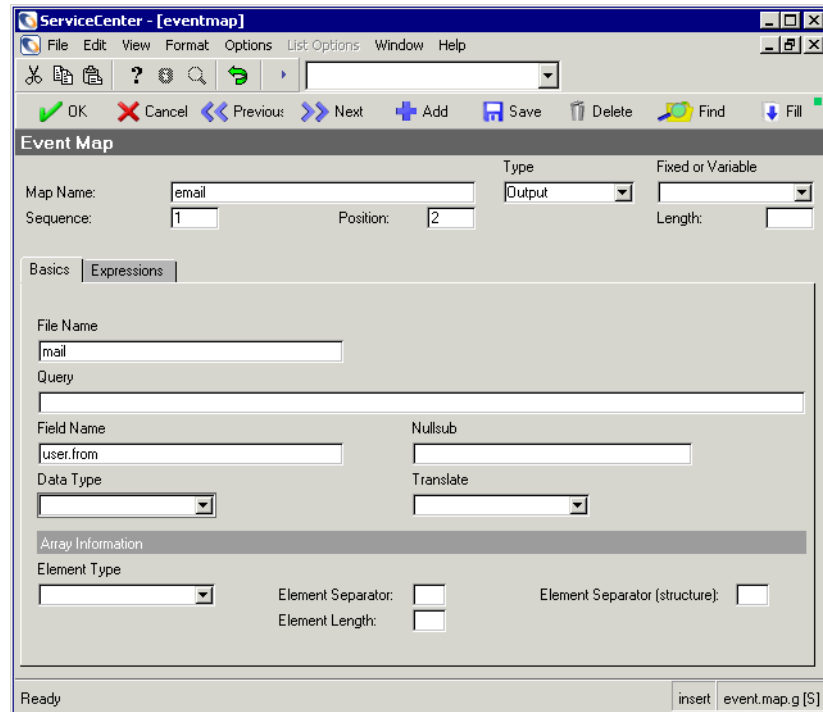


Figure 3-5: Incoming email event mapping

In the preceding record, the `user.from` field in the ServiceCenter mail file has a position of 2, and is the second field in the delimited text string written to the `eventin` record's **Field List**.

For output, the contents of the `user.from` field in the ServiceCenter mail file is placed in the second position in the **External Information String** field of the `eventout` record (see Figure 3-6 on page 166). The **Type** field is changed to *output*.



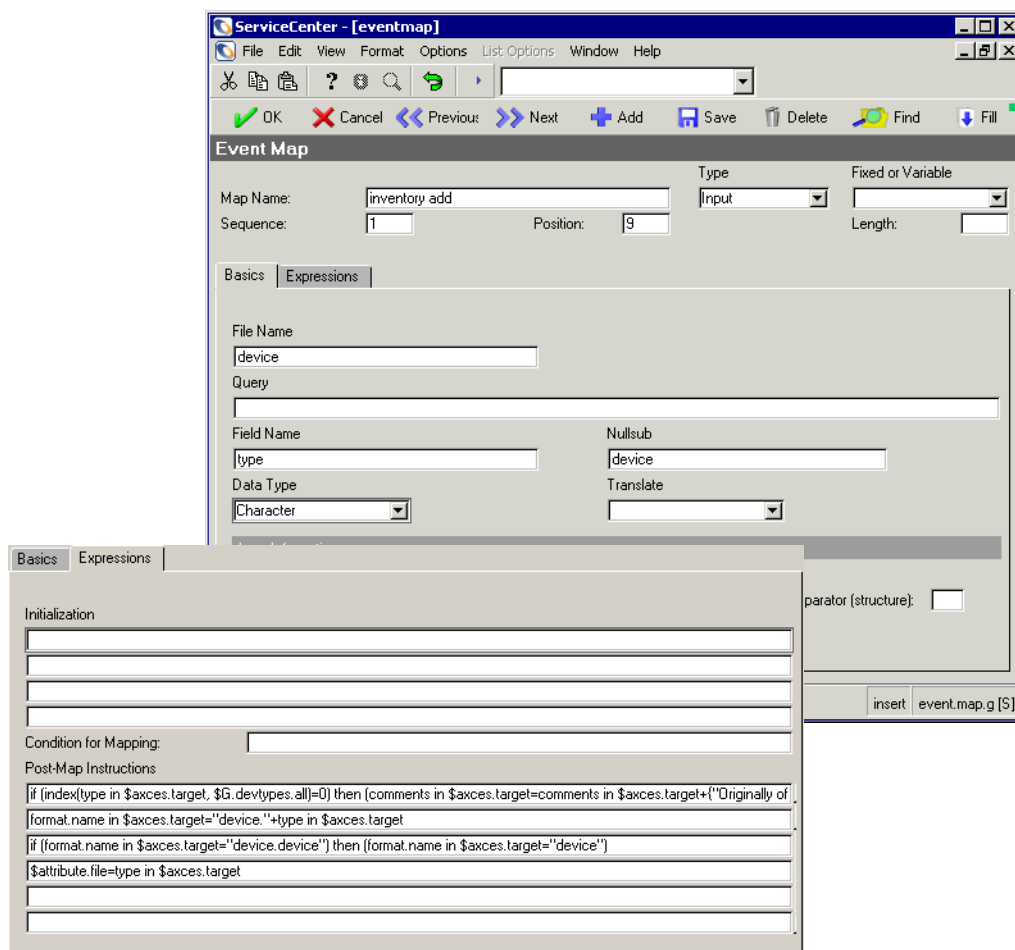
**Figure 3-6: Outgoing email event mapping**

If the mapping records for email are deleted, ServiceCenter uses the default described above and provided with the system upon installation of Event Services.

Event Services also handles mapping to multiple files. For example, SCAuto for NetView OS/390 and SCAuto may send inventory information that is stored in more than one file. The ICM applications use two files to describe each device: the `entity` file and the `attribute` file. The `entity` file is called



device; the attribute file depends upon the device type, and is identified by the **type** field in the entity file. When inventory information is gathered via discovery processes in external applications, such as OpenView and passed to ServiceCenter via SCAuto, both files may be updated.



**Figure 3-7: Multiple file event mapping**

The first step in preparing to map multiple files is to identify the attribute file. This is done using an expression (on line 4 in Figure 3-7 on page 167) in the **Post Map Instructions** to set the variable `$attribute.file` to the value in the **type** field of the device (TARGET) record.

**Note:** The Sequence is 1, and the File Name in the map record is device.

Until all fields are mapped to the device file, **Sequence** remains *1* and **File Name** remains **device**. No Query is necessary since the record has already been selected by the query passed in the Registration file.

After the last field for the initial file has been mapped, the record is added or updated and a new file is initialized based on the value of *\$attribute.file*.

**Note:** While *\$axces.target* and *\$axces.field* have special meaning within Event Services, *\$attribute.file* is an arbitrary global variable name.

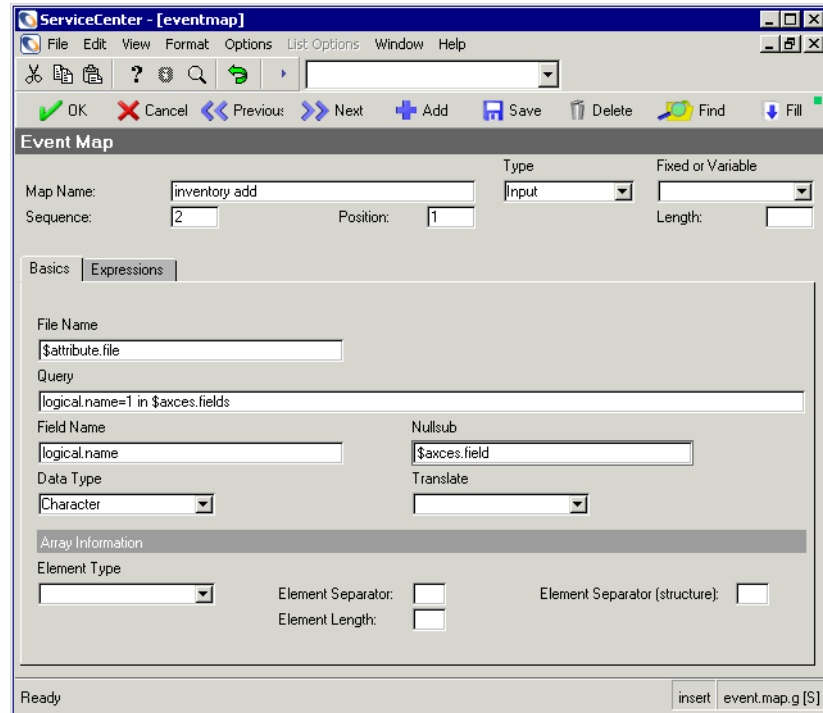
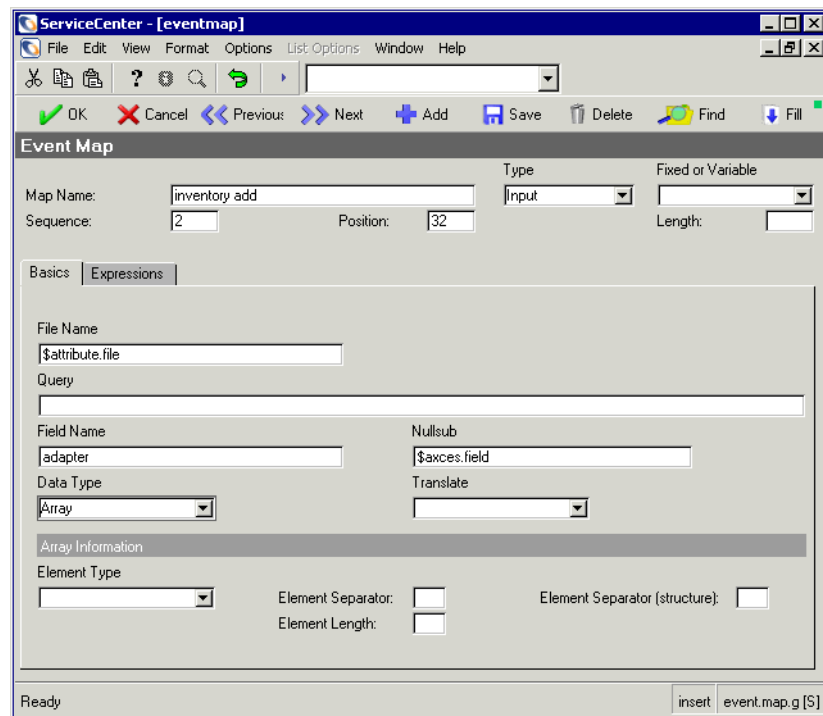


Figure 3-8: Using mapped events

When all fields have been mapped into the device file, the next map record has a **Sequence** of 2 (see Figure 3-8 on page 168), the File Name is different and a Query is supplied.

- **File Name** now contains the value assigned to the *\$attribute.file* variable.
- **Query** tells Event Services how to select the record to update from the file identified by *\$attribute.file*. The query can be either a literal statement (as illustrated) or a variable set in previous **Post Map Instruction** or **Initialization** fields.

The first mapping for the new file is *logical.name*, which is stored in **Position 1** of the *evfields* array field (which is itself represented by the *\$axces.fields* variable) in the *eventin* record.

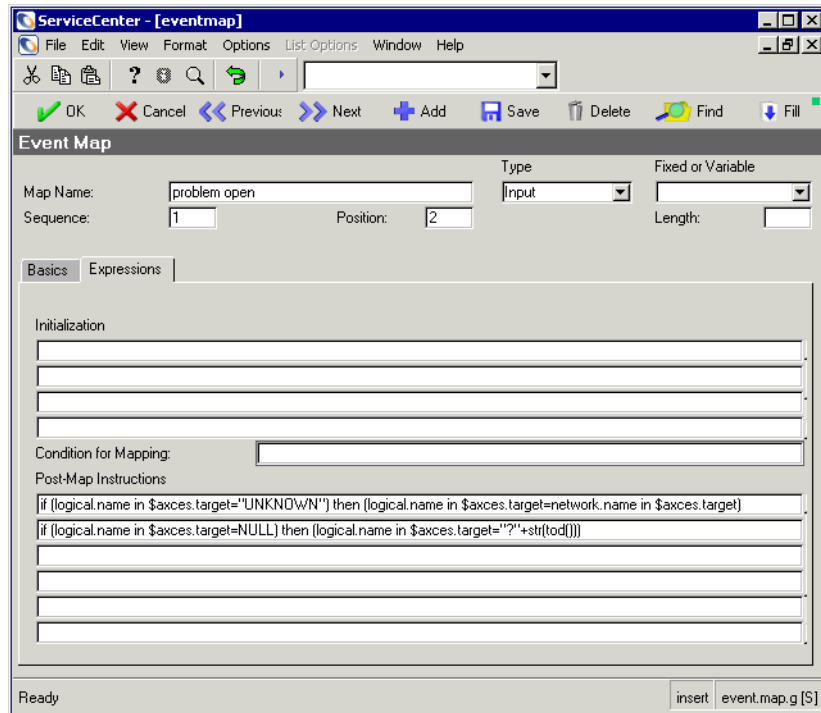


**Figure 3-9: Record placement**

Subsequent map records move data from the *eventin* record to the new file.

**Note:** When updating an existing record, Event Services substitutes the value in the original record for a null value passed from the *eventin* record.

Mapping also allows complete flexibility of data manipulation during the mapping process. Because Event Services runs as a background task, no input/output routines are available for on-line validation with user feedback, but field values may be checked and substitutions can be made based on processing statements.



**Figure 3-10: Condition statements**

In the preceding record, *logical.name* is replaced with the value in *network.name* if *logical.name* is *UNKNOWN*. The second statement sets *logical.name* to a constant if it is *NULL*.

Other common uses for expressions are to set the value of a field to the current date and time and to calculate a value based on information in the record. Data type and case conversions are handled by the Event Services applications, as long as the **Field Type** field is correctly identified and the data is to be written to the *descriptor* structure.

**Note:** You can use a single Format Control record named *login.event* to establish initial global variables (such as lists of valid operators) when the event agent is started, just as you can for users when they log into ServiceCenter.

---

**Important:** If you are writing data to a field whose name exists in more than one structure in a record, you must explicitly name the field. In other words, if you have added a field named *assignment* to the *middle* structure of your incident Database Dictionary record and you want to manipulate that field, you must identify it as *middle,assignment*. Remember that the field must exist in the target file before it can be manipulated by any instruction. Make sure the data type is correctly identified.

---

**Note:** Event Services data type conversions occur for *character*, *number*, *date/time*, *logical*, and *array* fields only.

## Global Variables

The following global variables are active when mapping event data:

Variable	Description
\$axces	Represents the eventin record.
\$axces.fields	Represents the evlist field in the eventin record.
\$axces.field	Value of a field in the target record at the time the target record is selected and before information is mapped to it from the event.
\$axces.register	Represents the event registration record.
\$axces.source	Map record.
\$axces.target	Record into which data is mapped; the record selected from the ServiceCenter database to which event information is posted.

Variable	Description
\$axces.notriml	If set to true, any blank spaces or tabs at the end of the field will not be removed.
\$axces.notrimr	If set to true, any blank spaces or tabs at the beginning of the field will not be removed.

**Note:** When email events are sent to ServiceCenter, the text field's leading and/or trailing spaces and tabs are not removed.

## Mapping Considerations for Inventory Management

While ServiceCenter provides both an entity file (**device**) and attribute files (for example, **server**), it is not necessary that both files exist to represent the characteristics of every device type. It is often the case that a device can be fully described using only the fields in the **device** file.

The map record for the **type** field (field #9 in standard events) defines how ServiceCenter selects and displays information about a device once the data has been added. The **type** field in the **device** file refers directly to the associated attribute file of each device. If there is no attribute file associated with a particular device, the **type** field should contain *device* or be empty (NULL).

Likewise, the **format.name** field in the **device** record defines the name of the form used to display the device within ServiceCenter and, by extension, the name of the join file used to temporarily store information for review and update. The **formatctrl** record for the format name stored in the **device** record should contain *device* as the file name for all device types that do not have associated attribute files.

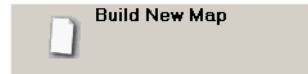
Should an unknown device type be detected by an external agent, ServiceCenter processes the event, updating the **device** file with the information provided. If no attribute file exists for that device type a Warning message is written to the event's Message list but the device is still added or updated in ServiceCenter's data repository. If event mapping indicates processing in more than one table, but the number of fields passed to the event is less than the position of the first field in the second table, there is no attempt to open the second table.

## Building a New Event Map

Both input and output event maps can be built for any file in ServiceCenter.

### To build a new map

- 1 Select the Administration tab in the Event Services menu (Figure 3-1 on page 159).
- 2 Click **Build New Map**.



The following form is displayed for building an event map.

 A screenshot of a software window titled "ServiceCenter - [\*\* enter map and file names \*\*]". The window has a standard menu bar with "File", "Edit", "View", "Format", "Options", "List Options", "Window", and "Help". Below the menu bar is a toolbar with icons for cut, copy, paste, help, search, and refresh. The main area of the window is titled "Build Event Mapping" and contains the text "Please enter the event map name and source file." Below this text are two input fields: "Enter Event Map Name:" and "Enter Source File Name:". At the bottom of the window, there are three buttons: "< Previous", "Next >", and "Cancel". The status bar at the very bottom shows "Ready" on the left and "insert build.eventmap.g [S]" on the right.

Figure 3-11: Form for building an event map

- 3 Enter the Map Name and a Source file name.

A unique name must be provided for each mapping.

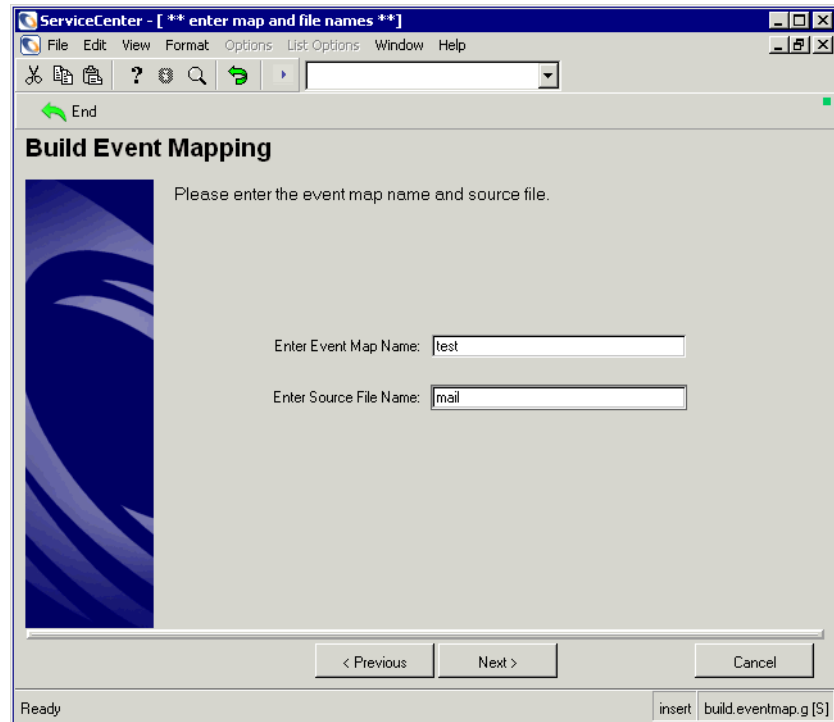


Figure 3-12: Completed map building form

#### 4 Press Enter.

If no source file name is provided, ServiceCenter displays a QBE list of files from which a selection can be made.

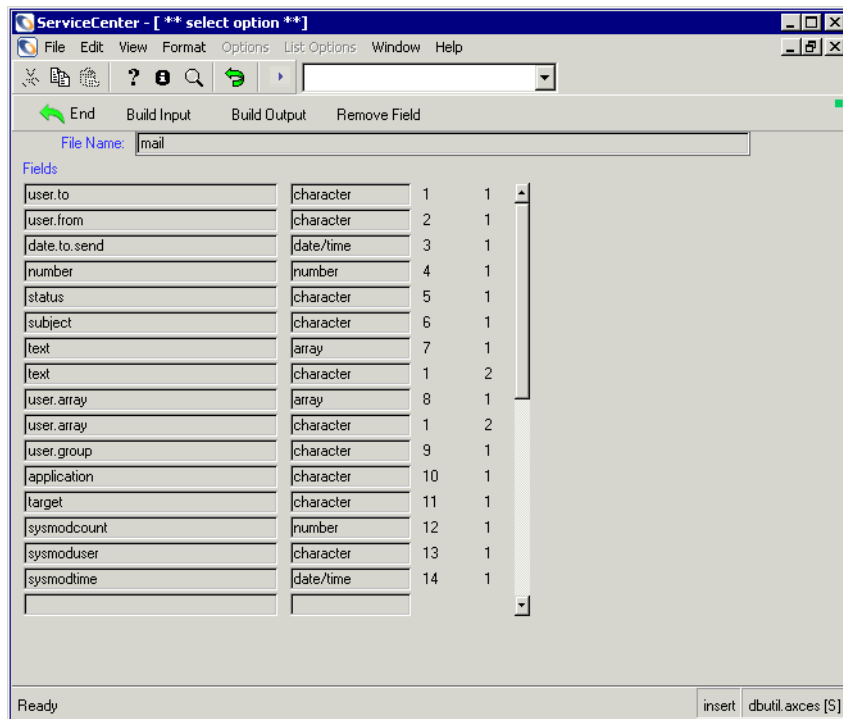
---

**Warning:** ServiceCenter issues a warning if the event map name entered already exists. In this case, building a new input map overwrites an existing input map and building a new output map overwrites an existing output map. If an input map exists and you are building an output map of the same name (or vice versa) the existing map will not be removed.

---



A list of field names and data types for the file you have selected is displayed.



**Figure 3-13: Event map fields**

Along the top of the screen, several buttons are displayed in the service tray which allow the user to manipulate and build a map record. These buttons and their functions are listed below.

Button	Property
Build Input	Builds the records used to map information from the eventin file to the selected ServiceCenter file.
Build Output	Builds the records to map information from the selected ServiceCenter file to a formatted string to be passed to SCAuto via the eventout file.
Remove Field	Deletes fields before a map is created. Place the cursor in the field you want to remove and click <b>Remove Fields</b> . Repeat this action for each field you want to remove.

**Note:** If an array field is part of your mapping, delete the second instance of the field in the list presented when building a new map, leaving only the array field.

## Rules for Building Maps

The purpose of event mapping is to relate elements in a list to fields in a record. When an external event, such as SCAutomate, or SCAuto for NetView OS/390, passes data into the ServiceCenter eventin file it does so in a field called **fields**. Each element is separated from the others with a delimiter, or separation character.

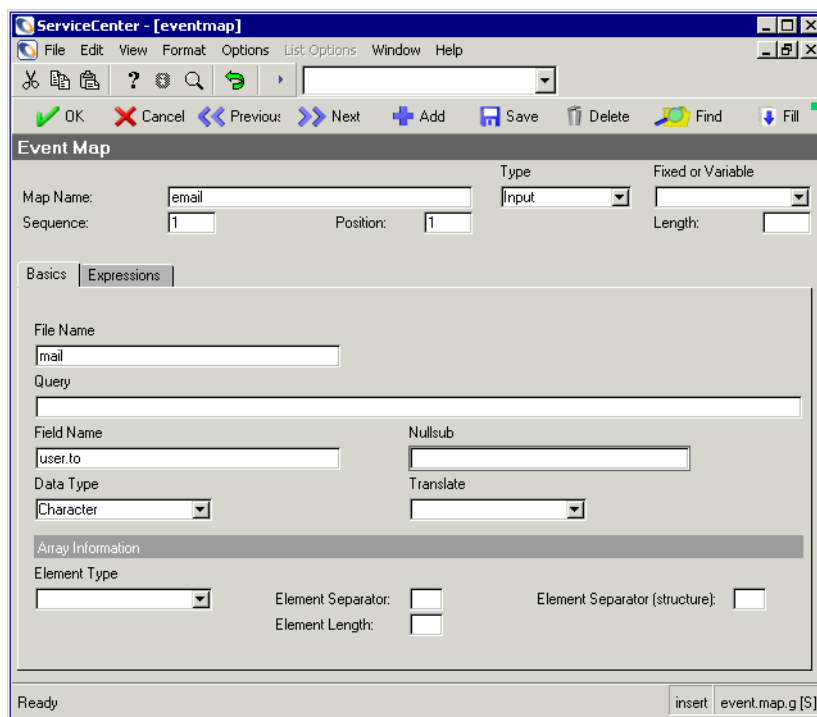
```
john@peregrine^falcon^toby;al;joe^Meeting today^Tue 12 Aug
```

In this example there are five fields, separated by the ^ character. Internally, Event Services converts this string to a list (*\$axces.fields*):

```
john@peregrine
falcon
toby;al;joe
Meeting today
Tue 12 Aug
```

The event processor assumes that fields with a type of date/time are in the time zone of the ServiceCenter system (that is, the time zone defined in the System Wide Company Record). If the event background process has its own operator record, that operator's time zone is used. For synchronous processing, the session processing the event handles the date/time in the time zone for which it is defined.

Mapping defines the link between the elements in the internal list (*evlist*) and fields in a ServiceCenter file. The first field, `john@peregrine`, is mapped to the mail file's `user.to` field.



**Figure 3-14: Array mapping**

For best results when building new maps which may utilize array fields, follow these guidelines:

- Select the first instance of any array fields (such as `user.array` in the mail file) so the proper type is built for the field.
- Only scalar and array fields can be directly mapped; all other types must be manipulated using expressions.

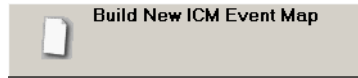
If possible, build maps first and then design external applications to use the maps.

## Building a New ICM Event Map

This option enables the generation of event registrations and maps based on the actual field names that exist for a particular device type. These do not supersede the existing ICM events. They are a different way of processing the ICM data that can be passed from Event Services. This method is used mainly for SMS related data. See the *SCAuto for SMS Guide* for more information.

### To create a new ICM event map and registration

- 1 Access the Event Services Administration tab (Figure 3-1 on page 159).
- 2 Select the **Build new ICM Event Map** button.



An introduction to the Asset Management Event Maps process is displayed.

- 3 Click Next.

The following form is displayed.

**Figure 3-15: Form for building an event map**

- 4 Click on the arrow button and select a device type from the drop-down list.
- 5 Enter a name for the Event Registration, or leave the field blank to use the default naming convention:

ICMdevice<type>.

- 6 Click **Next**.

The following confirmation message is presented: *This wizard has now created the Device Event Registration and Map.*

- 7 Click **Finish**.

## Event Filters

Event filtering information is stored in the *eventfilter* file. This file instructs SCAuto and SCAuto for NetView OS/390 when to block incoming events. If an event is not blocked, filters can also be used to prevent opening incident tickets based upon recurrence intervals and counts, and on incident intervals.

To review event filters:

- ▶ Click **Filters** in the Administration tab of the Event Services menu.

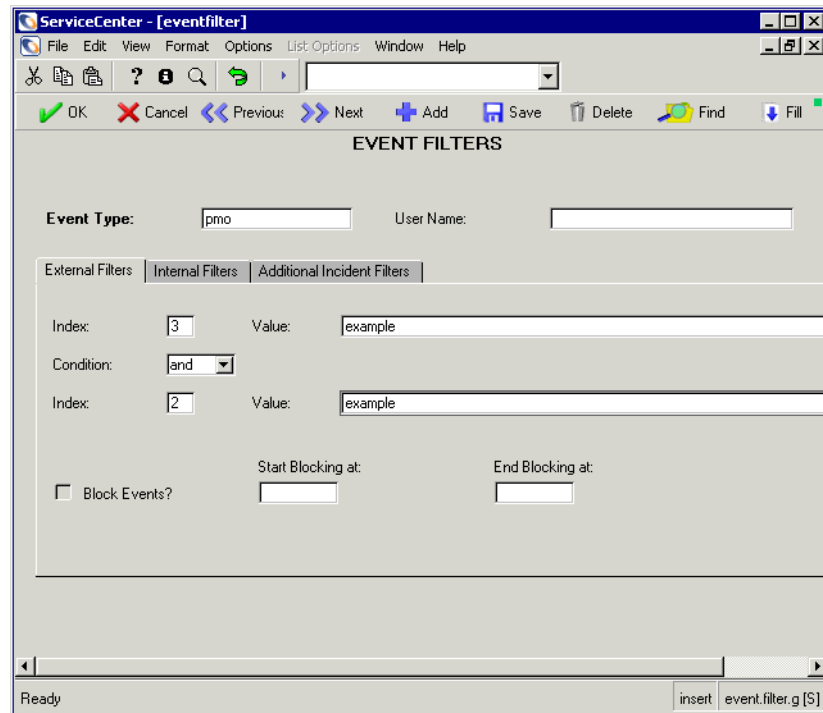


Figure 3-16: Event filters

## Fields

The encoded field names as recorded in the `eventfilter` file are included for reference only.

### Header

Field	Description
Event Type( <i>evtype</i> )	Unique identifier for the event filter; must match the code in the <code>eventin</code> record.
User Name ( <i>evuser</i> )	Name of the user or process, passed from external application; this field is REQUIRED when blocking events from being written to the <code>eventin</code> file by the external scheduler.

### External Filters Tab

Field	Description
Index ( <i>evindex1</i> )	Position in the <code>eventin</code> record's <code>evfields</code> field that identifies the first mask field.
Value ( <i>evvalue1</i> )	Value which, if it appears in the position indicated by <i>evindex1</i> in the <code>eventin</code> record's <code>evfields</code> field, causes that event to be masked.
Condition ( <i>evcondition</i> )	Value of <i>and</i> or <i>or</i> that is used to concatenate the clauses built with the <code>evindex</code> and <code>evvalue</code> fields.
Index ( <i>evindex2</i> )	Position in the <code>eventin</code> record's <code>evfields</code> field that identifies the second mask field.
Value ( <i>evvalue2</i> )	Value which, if it appears in the position indicated by <i>evindex2</i> in the <code>eventin</code> record's <code>evfields</code> field, causes that event to be masked.
Block Events? ( <i>evblock</i> )	Logical field that indicates whether events should be blocked entirely; this field is required when blocking events from being written to the <code>eventin</code> table by the external scheduler. See <a href="#">Blocking</a> on page 183 for more information.
Start Blocking at ( <i>evstime</i> )	Beginning time for masking events.
End Blocking at ( <i>evetime</i> )	Ending time for masking events.

## Internal Filters Tab

### Fields

### Description

Initial Statements (*evinit*)

Array of statements that are executed at run time to initialize variables or initiate action based upon the contents of the data passed in the **eventin** record and/or on global variables available at run time; the global variable *\$axces.fields* is used to represent an array of the fields passed in the **evfield** field of the **eventin** record.

Block Conditions (*evblockcond*)

List of conditions which, if any are true at run time, will block the event and cause the registered application to exit normally; the status in the **eventin** record will be *filtered*.

## Additional Incident Filters Tab



Fields	Description
Network Name ( <i>evnetnm</i> )	Unique network identifier for the device; the external application masks all events; should contain <i>SCAuto</i> for the master filter used for all internal blocking action.
Event Interval ( <i>interval</i> )	Amount of time an event must be active before an incident is opened in ServiceCenter; effective only when <i>evblock</i> is false.
Cause Code ( <i>evcode</i> )	Code, usually sent by the external agent, that identifies the fault.
Recurrence Count ( <i>recurrence.count</i> )	If completed, the number of times an event must be received for a particular <i>evnetnm</i> or <i>evcode</i> before an incident is opened in ServiceCenter; effective only when <i>evblock</i> is false.
Recurrence Interval ( <i>recurrence.interval</i> )	If completed, the amount of time (for example, 00:05:00) in which the <i>recurrence.count</i> is in effect; effective only when <i>evblock</i> is false.

## Blocking

The **External Filters** tab of the filter record is used by the external SCAuto and SCAuto for NetView OS/390 applications to prevent the insertion of eventin records in the ServiceCenter database. The contents of the **User Name** field must either match that of the external process or be empty (NULL).

The **Block Events?** condition must be set to *true* to prevent records from being added to the eventin file. The **Start Blocking at** and **End Blocking at** values are optional, however they allow for a block to be placed over a specified time frame allowing a more customize-able administration.

In the following record, all incident open events are blocked from 08:00 to 17:00.

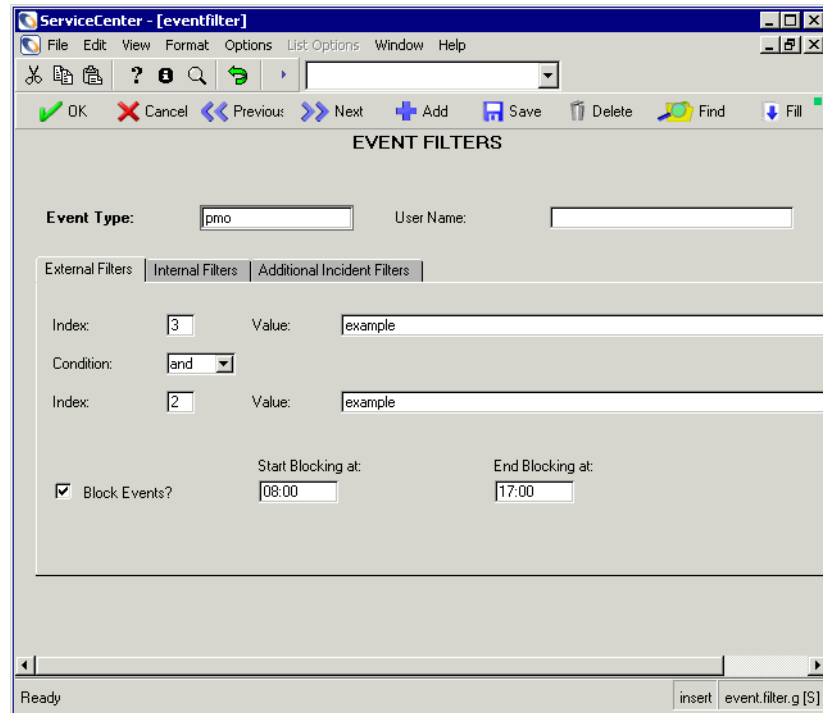


Figure 3-17: Filtering events by time

You can also prevent the insertion of events for specific network devices, domain names and error types by using the **Index**, **Value** and **Condition** fields. These can be used independently or in conjunction with the **Start Blocking at** and **End Blocking at** fields to populate other fields on the form.

- **Index** refers to the position of the data in the event message.
- **Value** refers to the actual data contained at that position.

For example, a *pmo* event may contain the following message:

```
peregrine^peregrine^^6 58916865^Node Down^^^^SNMP
Trap(IPAS)^net.hware^^^^^^^^^^^^^^
```

The ^ character separates fields in the message. The first field, which references the logical name of the device (refer to *Mapping* on page 158), contains *peregrine*. To block the insertion of all incident open events reported for the device *peregrine*, enter *pmo* in the **Event Type** field, 2 in the first **Index** field and *peregrine* in the first **Value** field.

**Note:** Only Index values of 2 or 3 are supported for incident open actions.

To block incident open events from both *peregrine* and another server named *dolphin*, enter information as previously described plus *or* in the **Condition** field, 2 in the second **Index** field and *dolphin* in the second **Value** field. If a condition has been specified (*and* or *or*), then *both* Index and *both* Value fields must be completed.

---

**Important:** To prevent insertion of records in the eventin file, the **Block** field must be *true*.

---

In the following tab, all *inventory add (icma)* events are blocked between 08:00 and 17:00 if they come from either the peregrine or dolphin server. This action might be taken to avoid unnecessary adds and updates if installation activity is scheduled to occur on the network during this time.

The screenshot shows a configuration window with three tabs: 'External Filters', 'Internal Filters', and 'Additional Incident Filters'. The 'Additional Incident Filters' tab is active. It contains two filter rules. The first rule has 'Index' set to '2' and 'Value' set to 'peregrine'. The second rule has 'Index' set to '2' and 'Value' set to 'dolphin'. The 'Condition' between the two rules is set to 'or'. Below the rules, there is a checkbox labeled 'Block Events?' which is checked. To the right of the checkbox are two time fields: 'Start Blocking at:' with the value '08:00' and 'End Blocking at:' with the value '17:00'.

**Figure 3-18: Inventory event filters**

The number of filters available for external blocking is unlimited; the external process (SCAuto or SCAuto for NetView OS/390) reads the eventfilter file to select records with the same **Event Code** and **User Name** (or **User Name=NULL**) and with **Block Events?=true** until it finds one that satisfies the criteria for the event being processed. If none is found the event is inserted in the eventin file.

Refer to SCAuto and SCAuto for NetView OS/390 documentation for further information regarding filters on specific platforms and with different external interfaces.

Once records have been added to the `eventin` file, Event Services assumes the filtering task using **Internal Filters**. Event Services first selects the filter with the same **Event Code** as that of the event being processed and with a **Network Name** of `SCAuto`. This filter should contain **all** internal blocking conditions. If an `eventin` record satisfies one of the **Block Conditions**, it will be updated to reflect a **Status** of *blocked*. The event action (for example, incident open or inventory add) will not take place.

The screenshot shows a configuration window with three tabs: 'External Filters', 'Internal Filters', and 'Additional Incident Filters'. The 'Internal Filters' tab is active. It contains two sections: 'Initial Statements' and 'Block Conditions'. The 'Block Conditions' section has a text input field containing the expression `(location in $axces.target)#!'Atlanta''`. Below this field are two empty text input fields and a vertical scrollbar on the right side.

**Figure 3-19: Event blocked by location**

With **incident open** event types (*pmo*), the **Additional incident Filters** take effect if no blocking condition exists. **This filtering mechanism is available only when opening new incidents.** Filters are selected using the following search criteria and in the order listed:

- The **Event Type** is the same as that of the event being processed and the **Network Name** is the same as the network name specified in the `eventin` record and the **Cause Code** is the same as the cause code specified in the `eventin` record.
- The **Event Type** is the same as that of the event being processed and the **Network Name** is the same as the network name specified in the `eventin` record.
- The **Event Type** is the same as that of the event being processed and the **Network Name** is `AXCES` and the **Cause Code** is the same as the cause code specified in the `eventin` record.
- The **Event Type** is the same as that of the event being processed and the **Network Name** is `AXCES`.

Using this event as an example:

```
peregrine^peregrine^^6 58916865^Node Down^^^^SNMP
Trap(IPAS)^net.hware^^^^^^^^^^^^^^
```

The queries would be:

```
evtype="pmo" and evnetnm="peregrine" and evcode="6 58916865"
evtype="pmo" and evnetnm="peregrine"
evtype="pmo" and evnetnm="AXCES" and evcode="6 58916865"
evtype="pmo" and evnetnm="AXCES"
```

You can permanently block problem open by entering a Network Name or Cause Code. This has the same effect as a **Block Condition** except that the status in the eventin record is filtered rather than blocked.

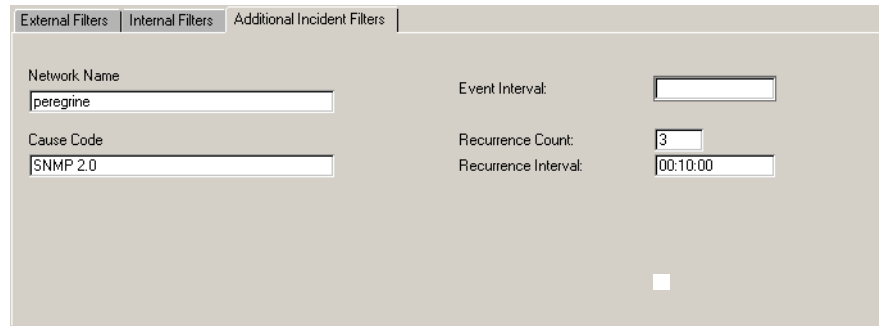
You can also use the **Event Interval**, **Recurrence Count**, and **Recurrence Interval** fields to limit problem open activity based upon frequency and duration.

The filter in the following record prevents any events from server **peregrine** with event code of **SNMP 2,0** from opening a problem unless three events are received within a ten minute interval.

External Filters	Internal Filters	Additional Incident Filters												
		<table> <tr> <td>Network Name</td> <td>Event Interval:</td> <td><input type="text" value="00:10:00"/></td> </tr> <tr> <td><input type="text" value="peregrine"/></td> <td>Recurrence Count:</td> <td><input type="text" value="3"/></td> </tr> <tr> <td>Cause Code</td> <td>Recurrence Interval:</td> <td><input type="text"/></td> </tr> <tr> <td><input type="text" value="SNMP 2.0"/></td> <td></td> <td></td> </tr> </table>	Network Name	Event Interval:	<input type="text" value="00:10:00"/>	<input type="text" value="peregrine"/>	Recurrence Count:	<input type="text" value="3"/>	Cause Code	Recurrence Interval:	<input type="text"/>	<input type="text" value="SNMP 2.0"/>		
Network Name	Event Interval:	<input type="text" value="00:10:00"/>												
<input type="text" value="peregrine"/>	Recurrence Count:	<input type="text" value="3"/>												
Cause Code	Recurrence Interval:	<input type="text"/>												
<input type="text" value="SNMP 2.0"/>														

Figure 3-20: Server-based event filters

The filter in the following record prevents any events from server **peregrine** from opening a problem unless 3 events are received and remain active for more than ten minutes.



External Filters	Internal Filters	Additional Incident Filters												
		<table><tr><td>Network Name</td><td><input type="text" value="peregrine"/></td><td>Event Interval:</td><td><input type="text"/></td></tr><tr><td>Cause Code</td><td><input type="text" value="SNMP 2.0"/></td><td>Recurrence Count:</td><td><input type="text" value="3"/></td></tr><tr><td></td><td></td><td>Recurrence Interval:</td><td><input type="text" value="00:10:00"/></td></tr></table>	Network Name	<input type="text" value="peregrine"/>	Event Interval:	<input type="text"/>	Cause Code	<input type="text" value="SNMP 2.0"/>	Recurrence Count:	<input type="text" value="3"/>			Recurrence Interval:	<input type="text" value="00:10:00"/>
Network Name	<input type="text" value="peregrine"/>	Event Interval:	<input type="text"/>											
Cause Code	<input type="text" value="SNMP 2.0"/>	Recurrence Count:	<input type="text" value="3"/>											
		Recurrence Interval:	<input type="text" value="00:10:00"/>											

**Figure 3-21: Time-based event filter**

# 4 ServiceCenter/Network Discovery Integration

CHAPTER

Peregrine Systems' Network Discovery product (formerly known as InfraTools™ Network Discovery (IND)) provides network monitoring capabilities within ServiceCenter. Depending on your Network Discovery license, Network Discovery automatically populates ServiceCenter's device records with data about the various devices on the network. Network Discovery can also send events to ServiceCenter through Event Services and automatically open problem tickets when a problem is detected on the network. You also can launch specific Network Discovery elements from ServiceCenter to quickly gather information about a device or problem.

**Note:** To launch a Network Discovery web client from ServiceCenter, you must have a Network Discovery login. See the *Network Discovery User's Guide* for information about using Network Discovery.

The chapter describes:

- *How Network Discovery and ServiceCenter Work Together* on page 190
- *Event Services Mapping for Network Discovery Device Information* on page 194
- *Opening and Closing Incident Tickets* on page 194

# How Network Discovery and ServiceCenter Work Together

ServiceCenter and Network Discovery work together to provide you with a complete network inventory and quickly notify you when a problem occurs.

Network Discovery supplies device and event information to ServiceCenter, utilizing Peregrine's Connect.It! and Event Services applications. This process is discussed later.

From ServiceCenter's Incident Management, Inventory/Configuration Management and Change Management, you can launch Network Discovery applications, via your web browser. This is also discussed later in this section.

For more information on using Network Discovery, see the *Network Discovery User's Guide*.

## What Does Network Discovery Provide to ServiceCenter?

Network Discovery is a real-time, web-based network manager. Network Discovery utilizes a network appliance to discover and monitor all SNMP-managed devices in your network. Network Discovery stores the information in a database located on the Network Discovery network appliance.

The device data gathered by Network Discovery is provided to ServiceCenter to automatically populate the device records in Inventory Management.

When Network Discovery detects that the information about a device has changed, such as its icon has changed, an event is passed to ServiceCenter to update that device record.

Network Discovery provides the following event types:

- Add device/port
- Delete device/port
- Change device/port
- Change device/port state
- Change connectivity information of device/port.



**Note:** Network Discovery provides state information for *attributes* associated with devices and ports rather than for the devices and ports themselves.

You can also use Network Discovery to find, diagnose, and solve network problems. When Network Discovery discovers a problem, an event is triggered. That event is passed to ServiceCenter, which automatically opens an incident ticket. When Network Discovery detects that the severity of the problem has changed, an update is made to the ticket. When Network Discovery detects that the problem has been fixed, another event is sent and the ticket is automatically closed.

## How Does Network Discovery Pass Information to ServiceCenter?

Peregrine's Connect.It! integration product translates the data sent from Network Discovery to ServiceCenter. Connect.It! is installed on the same machine as the ServiceCenter server. Connect.It! uses mapping scenarios to translate data from an outside source to ServiceCenter. For further information, see the *Connect.It! User Guide*.

XML data from Network Discovery is sent to Connect.It! over HTTP. Utilizing the mapping scenario, data from the XML tags is translated into the appropriate ServiceCenter event field. Event Services then provides the information to ServiceCenter.

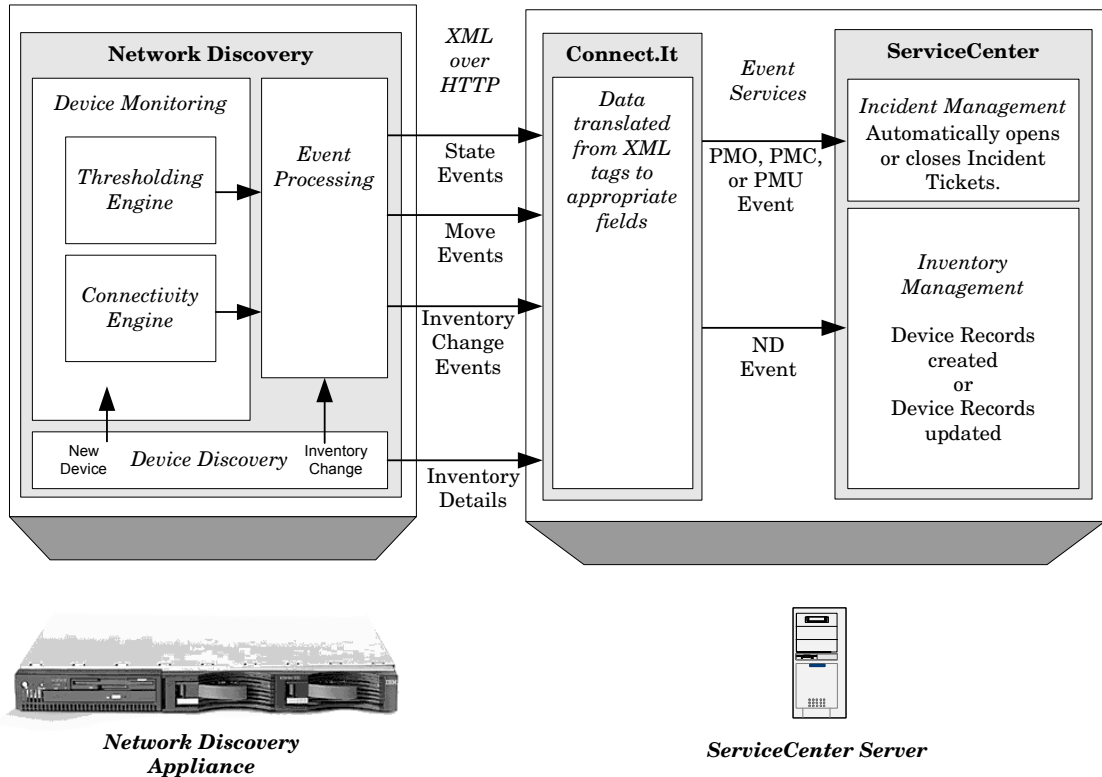


Figure 4-1: Network Discovery sends data to ServiceCenter

**Note:** Two types of events are discussed here. When Network Discovery detects a problem on the network, an event is generated. This is a message to indicate the problem was found. Connect.It! translates this information and passes it to Event Services.

Event Services uses its events to trigger actions in ServiceCenter. These events could come from either the Network Discovery event message or the device information.

## Network Discovery State Events

A Network Discovery event is the transition from one state to another on a device or port for the categories monitored by Network Discovery.

In the event database, the alarm or warning is logged when it is first detected, and then again when the condition abates. Information events are also generated when there is no alarm or warning, but a significant occurrence is detected for a device, such as when a device is added or deleted.

## Device information Goes to the Device Records

The device information from triggers a *Network Discovery* event in Event Services. This event creates and updates device records.

When Network Discovery discovers a new device, the data is passed along to ServiceCenter and a device record is created.

When Network Discovery discovers that the information about a device has changed, the information is passed to ServiceCenter, which updates the device record.

When Network Discovery discovers that a device is no longer on the network, the device is deleted from the Network Discovery database. The information is sent to ServiceCenter, which flags the device record. The record is not deleted from the ServiceCenter database.

## Event Information Goes to Incident Tickets

The events from Automated Resolution trigger either of these Event Services events. Connect.It! determines whether the event is to open or close a ticket:

- PMO (problem open) - automatically opens an incident ticket when a problem is discovered.
- PMC (problem close) - automatically closes an incident ticket when an incident is resolved.
- PMU (problem update) - automatically updates an incident ticket when a problem is modified.

# Event Services Mapping for Network Discovery Device Information

Connect.It! receives the device and port information and translates the data into event information that is sent to Event Services.

An event named *IND* has been added to the Event Services events. This event points to the event map *IND add*, which has the following format:

**To access the IND Add event map:**

- 1 Select the Administration tab in the Event Services menu.
- 2 Click Maps.
- 3 Type *IND add* in the Map Name field.
- 4 Click Search or press Enter.

A QBE list of all the *IND add* map records is displayed.

**Note:** For inventory additions and changes, the **ind.removed** field is set to *false* by the Connect.It!. For deletions, it is set to *true*.

## Opening and Closing Incident Tickets

Network Discovery can trigger ServiceCenter to close incident tickets that were automatically opened if Network Discovery detects that the problem has been corrected. Before the event reaches ServiceCenter, Connect.It! determines if a *problem open* (PMO), *problem update* (PMU), or *problem close* (PMC) event is sent to ServiceCenter by Event Services.

## Event Services Mapping for Network Discovery-Detected Problems

Network Discovery events provide information to Event Services for opening and closing incident tickets. These actions on tickets are triggered by the PMO, PMU, and PMC events, respectively.

To view the appropriate map:

- 1 Select the Administration tab in the Event Services menu.
- 2 Click **Maps**.
- 3 Type **problem open** or **problem close** in the Map Name field.
- 4 Click **Search** or press Enter.
- 5 A QBE list of all the *problem open*, *problem update*, or *problem close* map records is displayed.



# 5 Change Management Event Services

## CHAPTER

The Change Management module of ServiceCenter is fully supported by Event Services. This allows users outside of the ServiceCenter system to perform all standard functionality of Change Management from an external system, for example, SAP or PeopleSoft. The Event Services implementation is bi-directional, allowing external systems to synchronize with the ServiceCenter system.

This chapter provides the ServiceCenter system administrator with a basic understanding of the input and output events used to communicate data in and out of Change Management using Event Services. An administrator level of knowledge of Change Management and Event Services is required.

This chapter contains the following sections:

- *Input Events* on page 198
- *Keeping ServiceCenter In-Synch with an External System* on page 201
- *Change Event Examples* on page 203

## Input Events

A correctly formatted `eventin` record must be created within ServiceCenter to use an external system to produce an action within ServiceCenter's Change Management module. The `eventin` record can be formatted with an SCAutomate product.

The `eventin` record fields specific to the Change Management implementation are:

Field	Description
Event Code ( <i>evtype</i> )	Name of the corresponding Event Registration record to use for this event. This should always be <code>cm3rin</code> for changes and <code>cm3tin</code> for tasks.
User Name ( <i>evuser</i> )	User name in this field is interpreted as the operator for this event. The Change Management environment used depends on which user is entered in this field.
External Information String ( <i>evfields</i> )	Delimited data fields that correspond to a specific event mapping.

## Input Event Registrations

The following two registrations are used for input events:

Event Code	Input/Output	Event Map	Application	Description
<code>cm3rin</code>	Input	<code>cm3r</code>	<code>axces.cm3</code>	Used for Changes
<code>cm3tin</code>	Input	<code>cm3t</code>	<code>axces.cm3</code>	Used for Tasks

One of these two event codes must appear in the `eventin` record, depending on whether the event is related to a change or a task.



## Setting Up the External Information String

The External Information String, or EIS, is the **evfields** field of the **eventin** record. This field carries the specific data of the change or task into the ServiceCenter system. These fields are placed in a single string with a user-specified separation character (the default is the ^ character). The first four fields contain specific functions that determine which change/task is being processed and what action the system should take. These fields are passed in a specific order:

Sequence	Field Description
1	Change/Task number of the object to be acted upon. This field is blank when opening a change/task.
2	The foreign ID. This field is the identifier of the change/task used by the external system. This field is used if a different number is used outside of ServiceCenter.
3	Action Token indicates which logical action to take, either: open, update, close, reopen, approve, unapprove, disapprove.
4	The Change Group or Operator performing an approval action (only used for approve, unapprove, or disapprove.).

### Determining the Correct Change/task

The first two EIS fields are used to determine the unique identifier of the change or task both in ServiceCenter and in an external system (if applicable).

- The first field contains the unique number that corresponds to the number field in the cm3r or cm3t database dictionary. This field is blank if the action is an open.
- The second field of the EIS corresponds to the foreign.id field of the change or task. This field specifies the unique identifier of the change or task in the external system that is sending the request. If the ServiceCenter number is not specified, the system attempts to find the correct record by comparing the foreign.id to this field.

## Supported Actions

The third field of the EIS is used by Event Services to determine what type of action to perform on the specific change or task specified by one of the first two fields. The supported actions are:

Action	Description
approve	Approve a change/task
disapprove	Disapprove a change/task
unapprove	Unapprove a change/task
open	Create a new change/task
update	Update an existing change/task
close	Close current phase and go to the next phase if it exists
reopen	Reopen a change/task in the current phase

The third field of the EIS must contain one of these actions to correctly process the event.

## Approval Actions

When the action is an approval action (either an approve, disapprove, or unapprove), the Change Management Group or Operator Name that is performing the approval action must be specified in the fourth field of the EIS. The group or operator specified must match one of the approval groups specified in the change or task record for the approval action to complete properly.

## Data Fields

The remaining fields in the EIS contain field level data that Event Services uses to populate the change or task record being processed. If the action performed is not an open, these fields write over any existing data in the change or task. If a field in the EIS is blank, the existing data in the change or task is used.

The exact field that each piece of data corresponds to can be determined by examining the proper input event map for changes (cm3r) or tasks (cm3t).

# Keeping ServiceCenter In-Synch with an External System

When ServiceCenter is used with a separate external system, the changes and tasks must be synchronized between the two systems. Event Services supplies two methods of sending output to the external system for this task.

First, a simple acknowledgment can be sent to the external system. This acknowledgment contains enough data to map the ServiceCenter change/task number to the unique ID used in the external system, along with enough messages to determine if the input event was successful.

Alternatively, a complete output event may be sent to an external system in order to synchronize every piece of data between the two systems.

## Acknowledgments

In order to synchronize the unique numbers of each system, the `cm3rinac` and `cm3tinac` event registrations are used:

Event Code	Input/Output	Event Map	Application	Description
cm3rinac	Output	cm3ack	axces.write	Used for Changes
cm3tinac	Output	cm3ack	axces.write	Used for Tasks

Both event types use the `cm3ack` event map definition. This mapping passes the following fields in the EIS of the `eventout` record:

Sequence	Field Description
1	Change/Task number of the object being acknowledged.
2	The foreign ID. This is the identifier of the change/task used by the external system. This field is used if a different number is used outside of ServiceCenter.
3	Action Token indicating which action was performed on this object (open, update, etc.).

Sequence	Field Description
4	The status of the eventin record created by the original event. This field may be used to determine if there were any errors encountered when processing the original event.
5	An array of up to 5 messages sent during the original event (ex: Change 15 updated, Location XXX is invalid). These messages can be used to determine if a Format Control or validation error occurred during the original event.

The acknowledgment events can be turned on or off in the cm3rin or cm3tin Event Registration records by modifying the value associated with the boolean1 parameter on the application tab. When this parameter value is set to true an acknowledgment event is sent out each time an input event is processed, while a setting of false keeps the acknowledgment event from being sent.

## Sending Complete Output Events

The standard output events for Change Management are triggered by the cm3messages file. When the change scheduler processes a cm3message, the value is checked in the Event Services Reg (axces.out) field in the corresponding cm3message record. If the value matches an output event (most likely cm3rout or cm3tout), that event is processed and an eventout record is written. This gives an administrator great flexibility when deciding what types of events (opens, alerts, etc.) cause the output event to be written.

The output maps used for these events are cm3r and cm3t. These maps correspond to their related input maps with the exception of the third and fourth fields. The third field contains the name of the event that caused the event to process (for example, cm3r open or cm3t update). The fourth field is used as a place-holder to keep the data fields of the input and the output event synchronized and always contains the words not used.

# Change Event Examples

## Input Examples

### Open a Change

For example, you could open a change with the following parameters:

- The MAC category for pc001, with an external foreign ID of CM01, requested by falcon, assigned to bob.helpdesk.

The change contains a simple description while letting all other fields use default values.

The event register would have the following specific fields:

Field	Value
evtype	cm3rin
evuser	falcon

The EIS would be:

```
^CM01^open^^^MAC^^^falcon^^^bob.helpdesk^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^Move PC001 to Mike's
office.^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^pc001^^^...
```

The field positions correspond to the cm3r input event map as follows:

Position	Field Name	Value
2	foreign.id	CM01
3	actiondummy	Open
6	category	MAC
9	requested.by	falcon
13	assigned.to	bob.helpdesk
42	description	Move PC001 to Mike's office.
76	logical.name	pc001

## Output Examples

### Using cm3messages to Output Changes When Updated

Entering cm3rout in the cm3r update record triggers an output event whenever a change is updated.

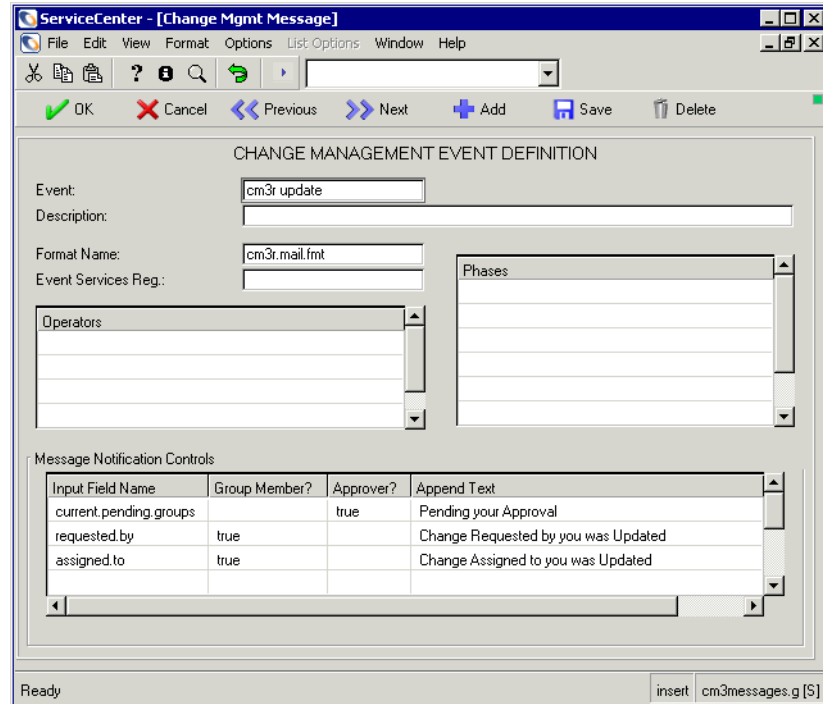


Figure 5-1: Triggering an output event when change is updated

# 6 Event Agent Operations

## CHAPTER

Automatic monitors within ServiceCenter, known as agents, can be set to collect data and create events appropriately within the system. These agents are set up using the Event Scheduler and can be activated either automatically or by user input.

Information on event agents has been divided into the following sections in this chapter:

- *Event Scheduling* on page 206
- *Maintaining Agent Status* on page 210
- *The VSAM Information Record* on page 214
- *The NAPA Information Record* on page 216

# Event Scheduling

The *schedule* file contains a record for each SCAuto agent. It contains instructions for how often the agent should read a queue, and which application should be executed if the read returns records.

## Reviewing Scheduled Events

To review SCAuto event schedules

- 1 Select the Services tab in the Event Services menu.

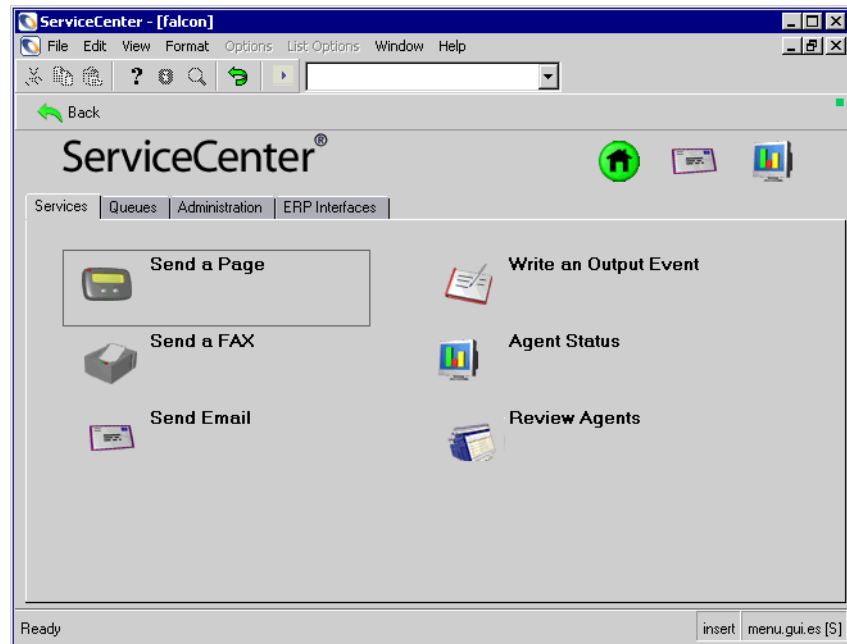


Figure 6-1: Event Services menu

- 2 Click Review Agents.  
A QBE list of event agents is displayed.
- 3 Select an agent from list.



The Event Scheduler is displayed.

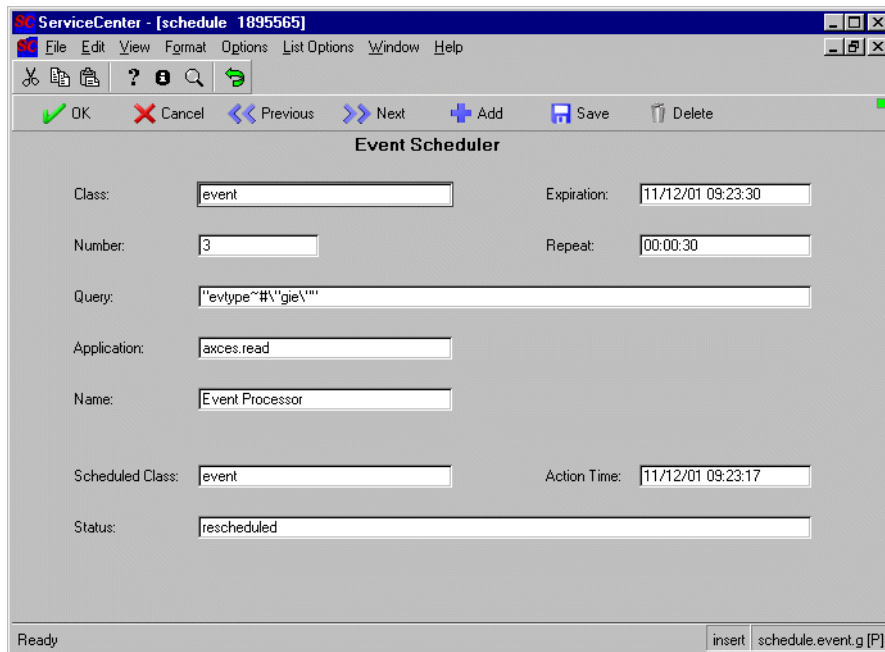


Figure 6-2: Event Scheduler

## Schedule Fields

The encoded field names as recorded in the schedule file are included for reference only.

Field	Description
Class ( <i>class</i> )	Schedule class; must match the name of the agent as defined in the <i>info</i> start-up record.
Expiration ( <i>expiration</i> )	Data and time when the agent will next be activated.
Number ( <i>number</i> )	Unique number to identify the schedule record.
Repeat ( <i>repeat</i> )	Interval defining the <i>sleep</i> time for the application.
Query ( <i>query</i> )	Optional query that can be combined with the class to allow multiple agents.
Application ( <i>application</i> )	Name of the ServiceCenter application called by the agent.

Field	Description
Name ( <i>name</i> )	Name associated with the agent. <b>Important:</b> For OS/390/SCAuto, SCAuto for NetView OS/390 and NAPA agents, the name must either be blank or match the name of the associated record in the config file. For example, if the config record that describes the input vsam file is named VSAMIN, the name in the agent record must be VSAMIN. If the name is blank, ServiceCenter uses the name of the class to select the config record. If a config record cannot be found with the name (or, if the name is blank, the class or schedule class) defined in the agent, the associated application will fail with an error.
Scheduled Class ( <i>sched.class</i> )	Class that was used when the application was last executed.
Action Time ( <i>action.time</i> )	Last time the application was executed.
Status ( <i>status</i> )	Current status of the event scheduler: <ul style="list-style-type: none"> <li>■ application running</li> <li>■ rescheduled</li> <li>■ application failed due to error</li> </ul>

When the event agent is started, the event schedule record must have a Class of **event** (or whatever you specify the event scheduler's name to be) and must have an expiration earlier than the current time. Set the expiration to the current date and time before starting the scheduler.

Since the event scheduler is a serial process, you may want to have more than one scheduler read events in the event queue. This is particularly true when inventory activity is high, preventing incident management activity.

The **Query** field can be used to further define what type of event should be selected from the eventin file.

The user-specified query entered in the schedule record is appended automatically to the default event scheduler query, `evtime<=tod()`, to form a more specific query. If the **Query** field is left blank, only the default query is applied.

**Note:** The system always places the time portion of the query in front of the user-specified query.

If you define a query for use against the eventin file, be sure it is fully-keyed for maximum performance.

---

**Important:** The agent processor will attempt to restart any applications that were killed while running (that have a status of *application running*). If you change for one of your agents, make sure there are no other agents with the same schedule class and a status of *application running*.

---

## OS/390 (MVS)/SCAuto Agents

SCAutomate allows you to read and write any number of VSAM files. For each VSAM file read or written, there must be a separate scheduler with a unique value in the **Class** and **Name** fields and a separate config record that defines the data set name (for example, *netview* for the SCAuto for NetView OS/390 agent).

All events read from a VSAM file are written to the eventin file. They must be in standard SCAutomate eventin format.

All events written to a VSAM file originate from the eventout file. They must use standard eventout format. Once the vsam.write scheduler has processed an eventout record, the **evexpire** field is set to NULL and the **Status** is updated with either *error* or *written*.

# Maintaining Agent Status

Agents can be started and stopped within ServiceCenter in several ways:

- System startup
- Status window
- Event agent check

## System Startup

To view the startup info record

- 1 Enter `info` in the command line and press Enter, or select **Utilities > Maintenance > Startup Information**.

A blank Agent Initialization Registry record is displayed

- 2 Enter `startup` in the **Type** field
- 3 Click Search or press Enter.

At system startup, all agents defined in this record are initialized.

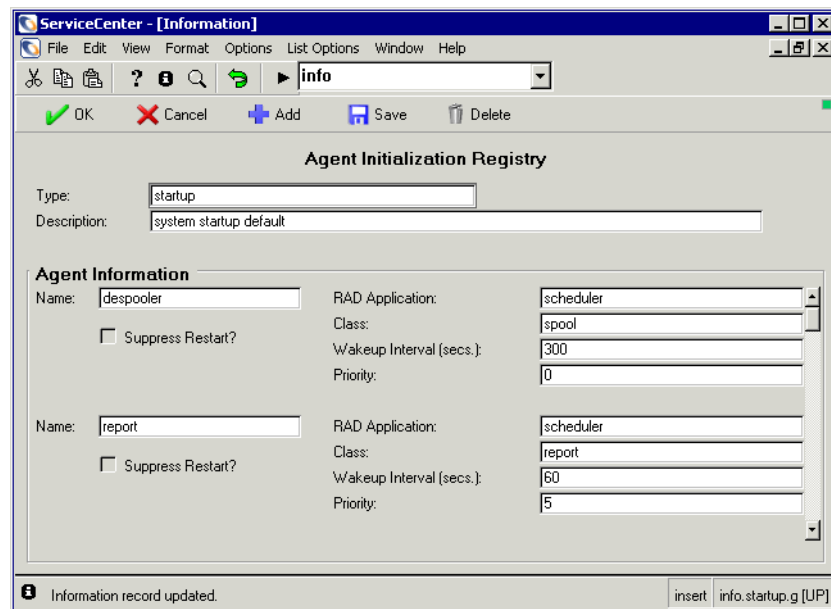


Figure 6-3: System startup info record

## System Status Window



Click the **System Status** button (on most ServiceCenter menus) or type `status` in the command line to display the system status window. From this window you can start, stop or kill individual agents by name, if you are either using an express client or are directly logged into ServiceCenter from its server.

Command	User Name	PID	Device ID	Login Time	Idle Time
	CLIENT-12870	1020	SYSTEM	11/09/01 08:35:45	01:32:33
	CLIENT-12680	1084	SYSTEM	11/09/01 08:35:46	3 02:17:13
	CLIENT-12870	1012	SYSTEM	11/09/01 08:35:46	3 02:17:13
	CLIENT-12680	1076	SYSTEM	11/09/01 08:35:46	3 02:17:13
	SCAuto Server	1228	SCAuto	11/09/01 08:35:50	3 02:17:13
	SCAuto Server	1220	SCAuto	11/09/01 08:35:50	3 02:17:13
	problem	1508	SYSTEM	11/09/01 08:36:00	00:00:07
	report	1464	SYSTEM	11/09/01 08:36:00	00:00:37
	sla	1572	SYSTEM	11/09/01 08:36:00	00:00:38
	change	1532	SYSTEM	11/09/01 08:36:00	00:00:49
	spool	1440	SYSTEM	11/09/01 08:36:00	00:01:59
	agent	1668	SYSTEM	11/09/01 08:36:01	00:00:25
	marquee	1728	SYSTEM	11/09/01 08:36:01	00:00:30
	lister	1756	SYSTEM	11/09/01 08:36:03	00:00:18
	linker	1804	SYSTEM	11/09/01 08:36:04	00:00:28
	event	1832	SYSTEM	11/09/01 08:36:05	00:00:45
	availability	1884	SYSTEM	11/09/01 08:36:06	00:00:49
	contract	1156	SYSTEM	11/09/01 08:36:07	00:00:41
	ocm	1920	SYSTEM	11/09/01 08:36:08	00:00:49
	alerit	756	SYSTEM	11/09/01 08:36:10	00:00:47
	falcon	808	Express-Windows NT	11/12/01 09:20:32	00:00:00

Figure 6-4: System Status Window

## Event Agent Check

From Event Services you can start and stop any SCAuto or event agent without respect to your client status as long as the ServiceCenter *problem* agent is active. Using this feature, agents are scheduled to start, and the *problem* agent is used as their activation agent. The specific agents controlled from this option include:

event	vsamin (SCAuto/OS/390)
vsamout (SCAuto/MVS)	scauto
scemail	netview (SCAuto for NetView OS/390 or NAPA)

### To maintain SCAuto agents

- 1 Access the Event Services menu.
- 2 Slect the Services tab.
- 3 Click Agent Status.



A form is displayed listing all the available agents.

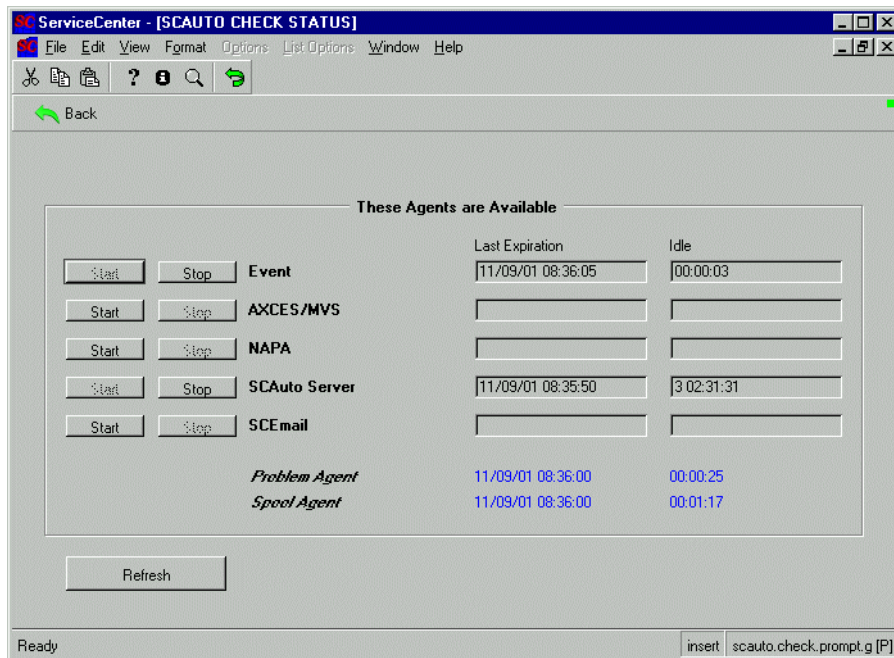


Figure 6-5: Agent Status start/stop control

For each active agent there will be a **Last Expiration** time and an **Idle** time. The **Last Expiration** time is the initialization time for the agent; the **Idle** time is the amount of time elapsed since the agent last woke up to check for work. If an agent is inactive, there will be no **Last Expiration** or **Idle** time, and the **Start** button is enabled.

- 4 Click **Start** to initialize the agent.

**Note:** The *sleep* interval is defined in the agent's *info* startup record.

- 5 Use the **Stop** button to disable an active agent.

---

**Important:** Since the **problem** agent schedules activation and deactivation of the agent, you must wait for it to wake up before your selected agent is started or stopped.

---

The OS/390 (MVS)/SCAuto agent automatically establishes both the **vsamin** and the **vsamout** agents.

You can define additional MVS/SCAuto agents to read from or write to other VSAM files, or event agents to selectively process input events, but these agents must be started and stopped using either the System Startup or the Status Window methods.

## The VSAM Information Record

The `vsaminfo` file contains records that reflect the status of external VSAM files read by Event Services tasks. This information is used by the scheduler to automatically open, update and close problems and to maintain inventory records in the database.

### Reviewing the `vsaminfo` record

To review the VSAM information record

- 1 Access the Event Services menu.
- 2 Select the Administration tab.
- 3 Click VSAM Information.





The VSAM information screen is displayed.

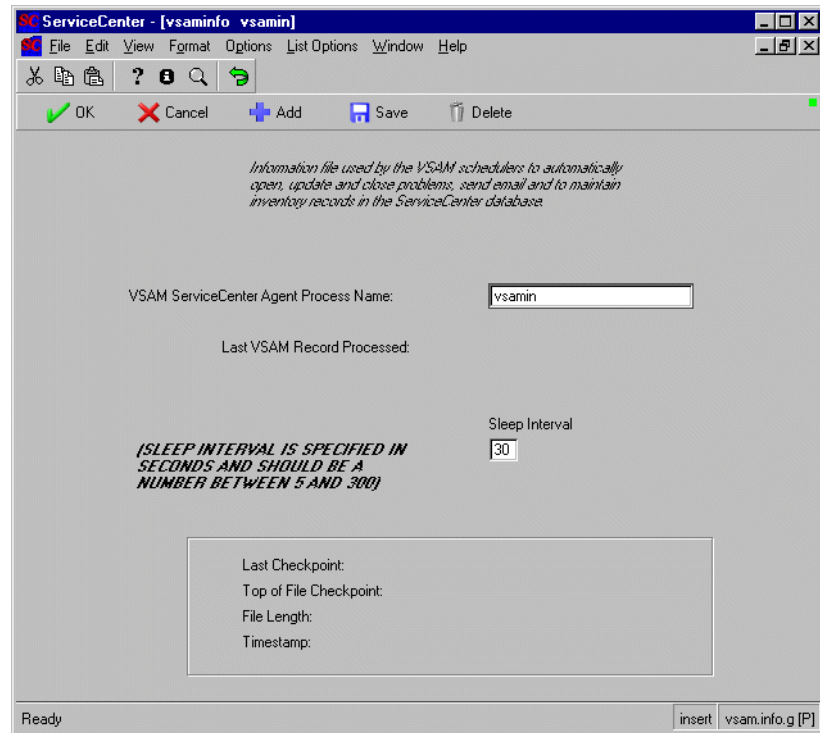


Figure 6-6: VSAM Information

## VSAM Information Fields

The encoded field names as utilized by the vsaminfo file have been included for reference only.

Field	Description
VSAM ServiceCenter Agent Process Name (name)	Name of the scheduler; this name must match the class in the schedule record.
Last VSAM Record Processed (item)	Index of the last VSAM record processed; do not modify this value.
Sleep Interval (sleep)	Number of seconds, between 5 and 300, to sleep if there is no NetView activity.
Last Checkpoint (sequence)	Checkpoint ID for the last record processed; do not modify this value.
Top of File Checkpoint (top)	Checkpoint ID for the first record in the VSAM file; do not modify this value.
File Length (length)	Length of the VSAM file (number of records; do not modify this value.
Timestamp (timestamp)	Timestamp in the last record processed; do not modify this value.

**Note:** The VSAM Information record is maintained by the `vsam.read` application.

## The NAPA Information Record

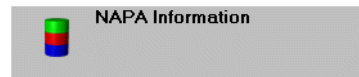
The vsaminfo file contains records that reflect the status of external VSAM files read by Event Services tasks. This information is used by the scheduler to automatically open, update and close problems and to maintain inventory records in the database. The NAPA Information is written by IBM's NetView products, and that information is fed to ServiceCenter via OS/390 (MVS)/SCAuto.

### Reviewing the napainfo record

To review the NAPA information record

- 1 Access the Event Services main menu.
- 2 Select the Administration tab.

### 3 Click on NAPA Information.



The NAPA information screen is displayed.

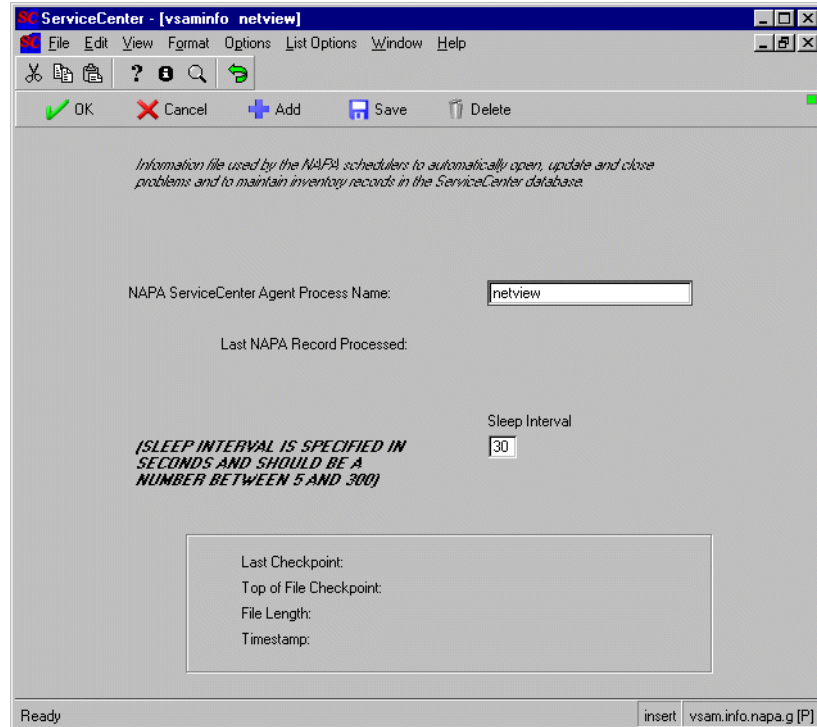


Figure 6-7: NAPA Information

## NAPA information fields

The encoded field names as utilized by the vsaminfo file have been included for reference only.

Field	Description
NAPA ServiceCenter Agent Process Name ( <i>name</i> )	Name of the scheduler; this name must match the class in the schedule record
Last NAPA Record Processed ( <i>item</i> )	Index of the last VSAM record processed; <b>Note:</b> Do not modify this value.
Sleep Interval ( <i>sleep</i> )	Number of seconds, between 5 and 300, to sleep if there is no NetView activity.
Last Checkpoint ( <i>sequence</i> )	Checkpoint ID for the last record processed <b>Note:</b> Do not modify this value.
Top of File Checkpoint ( <i>top</i> )	Checkpoint ID for the first record in the VSAM file <b>Note:</b> Do not modify this value.
File Length ( <i>length</i> )	Length of the VSAM file (number of records) <b>Note:</b> Do not modify this value.
Timestamp ( <i>timestamp</i> )	Timestamp in the last record processed <b>Note:</b> Do not modify this value.

The NAPA Information record is maintained by the `vsam.read` application.

# 7 SCemail

## CHAPTER

SCemail provides a monitor to handle ServiceCenter email events. This monitor connects ServiceCenter into standard email facilities and allows ServiceCenter operators and applications to send mail via email. Any mail system that supports SMTP (Simple Mail Transfer Protocol) or has an SMTP gateway or bridge can receive email from SCemail. SMTP is not required however to use SCemail. Mail support in OS/390 (formerly MVS) (mainframe) environments is extended to support any email that can be processed by the TSO *Transmit* command. Mail support on Windows NT systems includes support for MAPI-compliant mail servers.

This chapter contains the following sections:

- *Email Events* on page 220
- *SCemail vs. SCAutoMail* on page 220
- *Sending ServiceCenter Mail to email* on page 220
- *Changes to Existing ServiceCenter Mail Utility* on page 223
- *SCemail* on page 224
- *Sending email* on page 231

## Email Events

A standard email event that ServiceCenter creates is the opening of a problem with a valid **Contacts** field. This event can be used to notify individuals of a problem in their area of expertise. Also email events can be created using the **User Utilities Send Mail** function.

In addition to the standard creation of email events in ServiceCenter, any RAD application can create an event. An example of this would be implementing email notification for problems which reach a certain status.

## SCemail vs. SCAutoMail

SCemail is not the same product as SCAutomate Mail. SCemail only sends mail from ServiceCenter; it does not receive mail from external mail applications. SCemail runs as a stand-alone application, whereas SCAutomate Mail is an SCAutomate client adapter. More information on SCAutomate Mail can be found in the *SCAutomate Applications for Windows NT and Unix Guide*, also see the *ServiceCenter 4.0 Client Server Installation for OS/390 (MVS)* for details on using this service on the OS/390 platform.

## Sending ServiceCenter Mail to email

Sending ServiceCenter mail to email users is a quick process. Your System Admin must login and change the user's operator file to point to the external email address for that user.

### To modify a user's operator file for email

- 1 Login to ServiceCenter using a client with *SysAdmin* authority.

- 2 Select the Utilities tab in the system administrator's home menu.

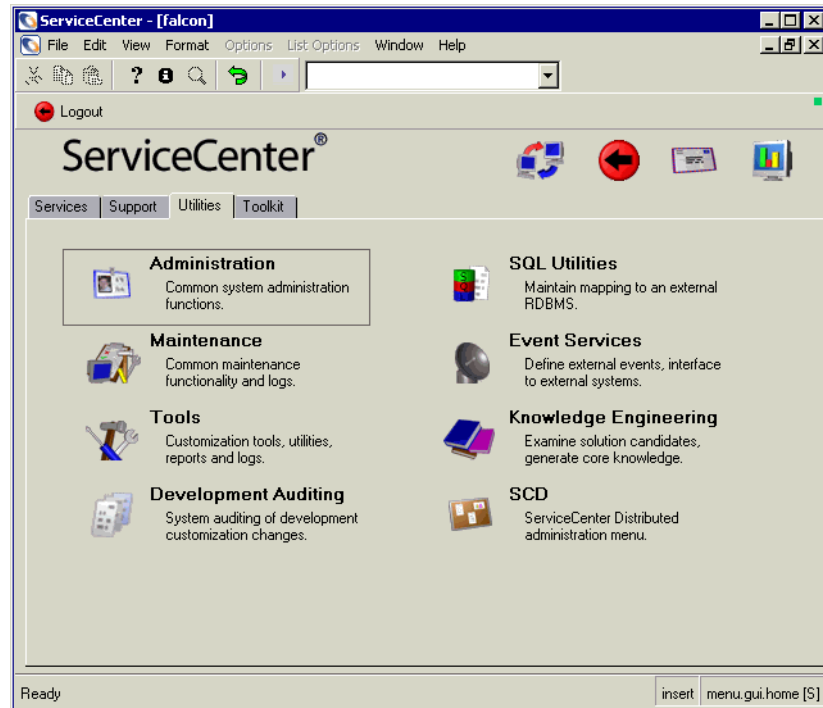


Figure 7-1: Utilities tab in the system administrator's home menu

- 3 Click Administration.

The Administration menu is displayed.

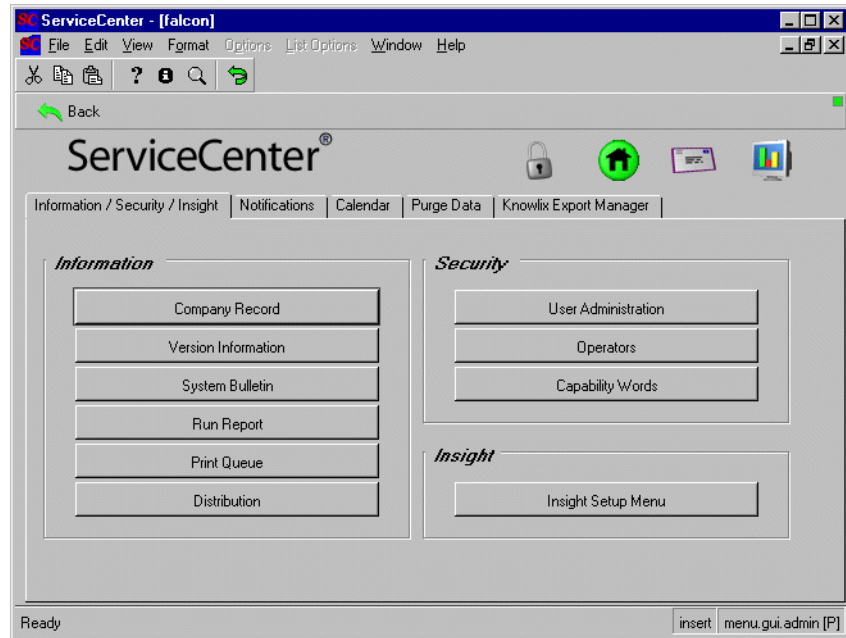


Figure 7-2: Administrator's menu

- 4 Click Operators.  
A blank operator record is displayed.
- 5 Enter the login name of the operator whose record is to be modified in the Login Name field.
- 6 Press Enter.
- 7 Enter the email address for that respective user in the Email Addr. field of the Notification tab.



- 8 Click Save to record the operator record.

The screenshot shows the ServiceCenter application window titled "ServiceCenter - [User profile JACK]". The menu bar includes File, Edit, View, Format, Options, List Options, Window, and Help. The toolbar contains icons for OK, Cancel, Previous, Next, Add, Save, Delete, Views, Finc, Fill, and Fill User R. The main window is titled "OPERATOR RECORD" and has several tabs: General, Security, Login/Contact Profiles, Startup, Notification, Security Groups, and Billing Information. The "General" tab is selected, displaying the following fields:

- Email Addr.: jack@peregrine.com
- FAX: 858-481-1751
- Mailbox: (empty)
- Printer: (empty)
- Client Printing:
  - Lines Per Page: (empty)
  - Page Width: (empty)
  - Header Format: (empty)
- Paging Information:
  - Phone No.: 858-481-8000
  - Vendor: (empty)
  - Group: (empty)
  - Type: (dropdown menu)
  - Name: (empty)
  - PIN: (empty)
  - Voice MB: (empty)
  - 2-Way Page?

The status bar at the bottom of the window shows "Ready" and "insert operator.g [P]".

Figure 7-3: Configuring email notification

## Changes to Existing ServiceCenter Mail Utility

The existing ServiceCenter Mail Utility checks the operator file for valid operator names before allowing mail to be sent. The Event Services version of this application expands the checking for valid users to those defined in the ServiceCenter contacts file.

The purpose of the checking is to obtain the email address from the operator or contacts file's email field. If the addressee's name does not select a record from either file, ServiceCenter assumes that there is no such addressee and does not send mail. You can override this default by creating a Format Control record named *login.event* and, in the Calculations section, setting the add condition to *true* and the calculation expression to the following:

```
$email.noaddr.ok=true
```

This causes ServiceCenter to assume that whatever name is passed to the email event as the addressee is the complete email address and attempts to send mail using that address.

## SCemail

The next three sections present high-level overviews of how SCemail works in Windows NT, Unix and OS/390 environments.

### Windows NT

SCemail is a program to allow sending of email within ServiceCenter. Under Windows NT, SCemail uses the Messaging Application Program Interface—MAPI. Microsoft Exchange, Lotus Notes, Lotus cc:Mail and other mail vendors support this interface.

#### Mail Profiles

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

For example, your SCemail profile could be **Joe**, however that profile contains the MS Exchange, cc:Mail, Lotus Notes etc. login and mailbox account information which allow you to interface with those systems, for example, **M:\mail, JJohnson**.

When using SCemail, you need to sign on using the SCemail profile, not the external mail account or login names. It is for this reason that a unique SCemail profile needs to be established for each user, in addition to having a standard mail account.

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

---

**Important:** SCemail does not work under Windows NT 3.51 unless MAPI is upgraded.

---

It is highly recommended that SCemail be given its own MAPI profile and its own mailbox or mail account. This mail account will act as a gateway from ServiceCenter.

## Adding a New Profile

The mail products being used may have additional documentation on this operation, consult this documentation before continuing with this process.

### To add a new profile

- 1 Locate and open the **Control Panel** on your Windows Desktop.
- 2 Select and open the *Mail* or *Mail and Fax* icon.
- 3 Click **Show Profiles** to display the profiles for your computer.
- 4 Click **Add**.  
The Setup Wizard is displayed.
- 5 Select the service to use (SCemail will only use one service, so do not select more than one).
- 6 Name and configure the profile as directed by the wizard.

This is where you assign the mailbox or mail user that SCemail will use.

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

## Starting SCemail

To send mail, SCemail must login to Windows mail.

**Note:** ServiceCenter must be already installed and operational.

### To start SCemail and login to Windows-based email

- 1 From a DOS prompt, change to your ServiceCenter RUN directory.  
For example, `cd c:\scserver\run`
- 2 Once in the ServiceCenter RUN directory, enter `scemail` followed by the mail profile name. You will need to use double-quotes if the profile name contains spaces.  
For example, `scemail "<profile name>"`

This starts the SCemail background processor.

- 3 Check the `sc.log` file (usually located in the top level of your ServiceCenter folder) to verify that the SCemail background processor started successfully.

If the processor started successfully, the `sc.log` file will display the following message: *SCemail: Initializing.*

### SCemail on Windows NT Helpful Hint:

There are different syntax variations when entering your address in the operator record. You should enter the name as it appears in the address book of an external mail client. You may also use SMTP style addresses of the form `username@host.com`.

Once you have made the above corresponding changes to the operator record, any user that has access to send mail should be able to send ServiceCenter mail. If mail sent from ServiceCenter is undeliverable, it is returned to the user with an error message.

## Optional Parameters

The following optional parameters may be given when starting the SCemail background processor:

Parameter	Description
-keepmail	Do not delete mail events once sent successfully.
-sleep <n>	Number of seconds to sleep between checking for events and mail. Default is 10 seconds.
-gui	Allow a pop-up dialog if additional login information is required (no profile was passed on the command line, or a password is required).
-debug	Print more diagnostics to sc.log. This turns on -keepmail as well.

**Note:** SCemail will also process the *sc.ini* file for additional parameters, which may also be passed on the command line (i.e., *-log:file* will place the SCemail diagnostics in a different file).

### Additional Windows NT Compatibility And Setup Notes

If you will be using Lotus Notes, the following restrictions apply:

- Lotus Notes version 4.11 or higher is required to work with MAPI.
- Be sure to install Windows Messaging (a part of Windows NT 4.0), Microsoft Outlook, cc:Mail, or other MAPI-compliant mail client before installing Lotus Notes. This applies even if you will not use those mail clients, as Lotus Notes will not install the necessary MAPI libraries.
- After setting up a profile for use with Lotus Notes, edit the properties of the profile and select the Delivery tab. Change the selection under *Deliver new mail to the following location* to read **Lotus Notes Message Store**.
- When SCemail starts, it prompts for a password, even if one is given on the command line, regardless of the -gui parameter.
- Do not install Microsoft Office 97 on the machine that is running Lotus Notes and SCemail. Office 97 upgrades MAPI automatically to a version that does not work well with Lotus Notes, and may not work with other MAPI service providers. This restriction holds for Lotus Notes 4.5, but may be removed in a later version.

If you will be using Lotus cc:Mail, the following restrictions apply:

- Lotus cc:Mail for Windows version 7 or higher is required to work with MAPI. (This means that the release 6, or DB8, postoffice is required).
- If the profile has a password, then you must pass the **-gui** flag when you start SCemail, otherwise SCemail will terminate with an error. This can be avoided by selecting the **Remember Password** checkbox when logging in with a normal cc:Mail client.
- Periodically check the **Outbox** of SCemail's profile for deleted messages to be purged.
- SCemail only runs as a Windows NT service if the mail service providers are *tightly coupled*. This is true even if SCemail is started from ServiceCenter, as ServiceCenter runs as a Windows service. As of this writing, the only mail service provider that does this is Microsoft Exchange Server. For other mail service providers, SCemail must be run from an interactive desktop.

## Unix

Unix email support consists of a daemon(scemail) process which reads output email events and sends them to the addressed parties. Figure 7-4 on page 228 illustrates the SCAuto email monitor.

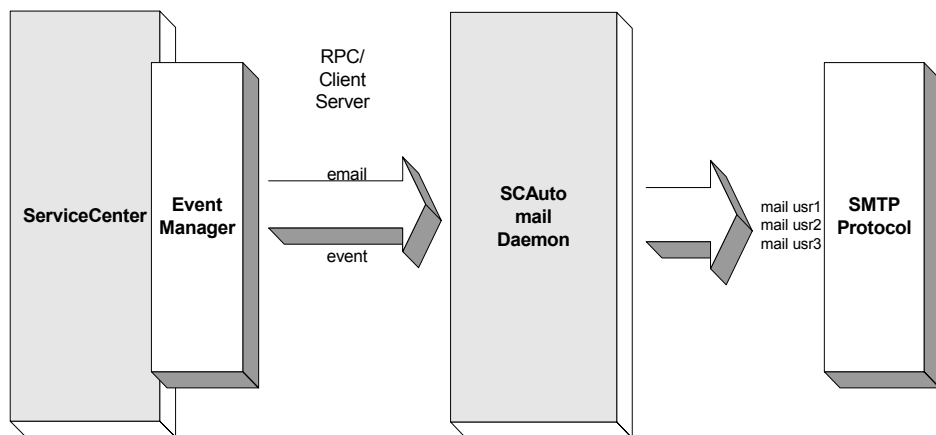


Figure 7-4: SCemail Unix email Monitor

- The mail monitor reads mail from ServiceCenter and delivers it to the SMTP network. Output mail is formatted by the ServiceCenter mail routines to contain the mailing address.
- Two files are created in the runtime directory: a checkpoint file and a log file. The checkpoint file maintains a pointer to the ServiceCenter eventout file that keeps redundant mail from being sent after a restart. If the checkpoint file is not found, all mail events are sent. The log file contains error and execution information.
- Unix SCemail requires the standard Unix mail utility.

## OS/390

OS/390 email support consists of a batch TSO address space which reads output email events and sends them to the addressed parties using the TRANSMIT command. The figure below illustrates the OS/390 SCemail monitor. See the *Client Server Installation Guide for OS/390 (MVS)* for details on using this service on the OS/390 platform.

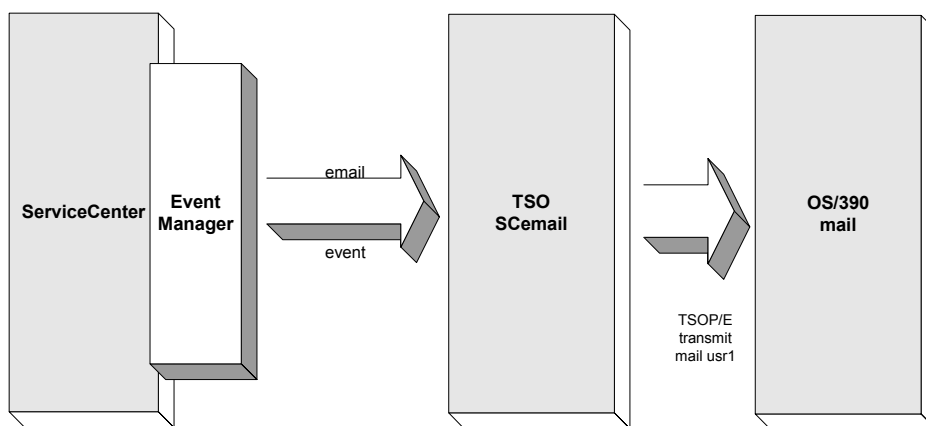


Figure 7-5: SCemail OS/390 email Monitor

Application	Description
message.fc	Called from Format Control, sends messages under user control.

## Parameters

Name	Value	Default
index	Message Level: 1 Information 2 Action 3 Error	1
prompt	Message Class	msg
text	Message Text	none
name	User Name	operator()
string1	Message Name	none
number1	Message Number	none
query	Mail Class	none
names.1	Mail Target	none

## Programming Considerations

- In text mode, different Message Levels generate messages with different attribute settings. For example, error messages are red, while information messages are white.
- The Message Class should match one of the records in the `msgclass` file, for example, *problem close*. To send email, for example, there must be a `msgclass` record with a type of *email* for the Message Class name specified.
- The Message Text can be either a string or an array. You can generate an array of the screen contents using the `genout()` function, for example, and then insert lines of text at the top of the array.  
For information regarding setting the appropriate array properties, see the Text section of The Tools Palette chapter in *System Tailoring, Volume 1*.
- The User Names can contain either a list of operator names or a single operator name. For internal (SC) messages, the name(s) in User Names must be operator Ids defined in the operator table. For email type messages (Message Class *email*) the User Names must be either operator IDs defined in the operator table or `contact.names` defined in the `contacts` file, and an email address must be specified in the relevant table.
- The Message Name parameter is used to identify the message. In SC applications it is usually the name of the application or application area that generates the message. This parameter is not required.



- The **Message Number** parameter is used to identify a message within the area specified by the **Message Name** parameter. This parameter is not required.
- The **Mail Class** parameter is used within the Incident Management applications to identify the problem number so that mail already sent can be selected and updated. If used, it should contain the string `pm.main` and the **Mail Target** should also be supplied. This parameter is specific to Incident Management and is not required.
- The **Mail Target** parameter, when used, must contain the problem number (in number form). This parameter is specific to Incident Management and is not required.

## Sending email

### Using Format Control

SCAutomate supports a generic email function. email events can be written to the eventout file via a subroutine call to *message.fc* from Format Control.

Parameter Name	Parameter Value
index	1
prompt	msg
text	Message Text
name	operator()
string1	Message Name
number1	Message Number
query	Mail Class
names,1	Mail Target

## From Incident Management

Incident Management uses message classes to determine how messages should be handled when incidents are opened, updated (including escalation) and closed.

To configure ServiceCenter to always send email to all members of the assignment group when an incident is opened

- 1 Select the Utilities tab in the system administrator's home menu.
- 2 Click **Administration**.  
The Administration menu is displayed (Figure 7-2 on page 222).
- 3 Select the Notifications tab.

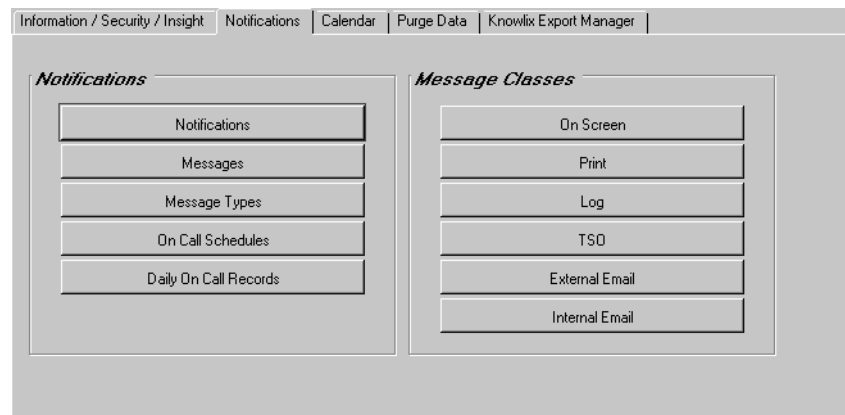


Figure 7-6: Notifications tab in the Administration menu

- 4 Click **External email**.

## 5 Create a record with a Class Name of problem open.

The screenshot shows a window titled "ServiceCenter - [msgclass]" with a menu bar (File, Edit, View, Format, Options, List Options, Window, Help) and a toolbar with icons for Back, Add, and Search. The main area is titled "Message Class File" and contains the following fields:

- Class Name:** A text box containing "problem open".
- Description:** A text box containing "Send email notification to those users listed below when a problem is opened."
- Type:** A dropdown menu set to "email".
- ALWAYS send to THESE users (true/false):** A checkbox set to "true".
- User Names:** A list of text boxes containing "BOB.HELPDESK" and "SUSIE.SUPERTECH", with three empty boxes below.

The status bar at the bottom shows "Ready" on the left and "insert | msgclass.email.g [UP]" on the right.

Figure 7-7: Email message class record on problem open event

- 6 Click Add to save the record.
- 7 To send email with the same rules upon update, escalation and close, use the same procedure to add records for *problem update* and *problem close*.

**Note:** If you need more discrimination on when to send email, for example, if you only want to send email to the **Contact Name** when a problem is closed, use the method described in [Using Format Control](#) on page 231 to utilize the Format Control for the category and function used (for example, *problem.software.close*).



# 8

## Format Control Options

---

### CHAPTER

This chapter discusses using Format Control to generate certain output in Event Services.

This chapter contains the following sections:

- *Generating eventout Records* on page 236
- *Generating Page Messages* on page 240
- *Sending Fax Messages* on page 244
- *Creating Output Events* on page 245

# Generating eventout Records

Format Control can be used to generate **eventout** records in Incident Management and Inventory and Configuration Management.

## Format Control

Application	Description
aces.write	Called from Format Control, builds an <b>eventout</b> record used by the SCAutomate interface.

### Parameters

Name	Value	Default
record	The record to be written	none
name	The name of registration type	none
string1	The separation character	^
text	The system sequence ID	system generated
prompt	The user sequence ID	none
query	The user name	operator name

### Programming Considerations

- The record parameter is required. The application will exit if this parameters is not provided.
- The registration name must exist in the eventregister file. If it does not, the application will exit.
- If no eventregister record with a type of output can be found, the input registration record will be used.
- Mapping is defined either by the format name or the map name. For most SCAuto/SDK events you will want to use the Map Name to properly format fields.

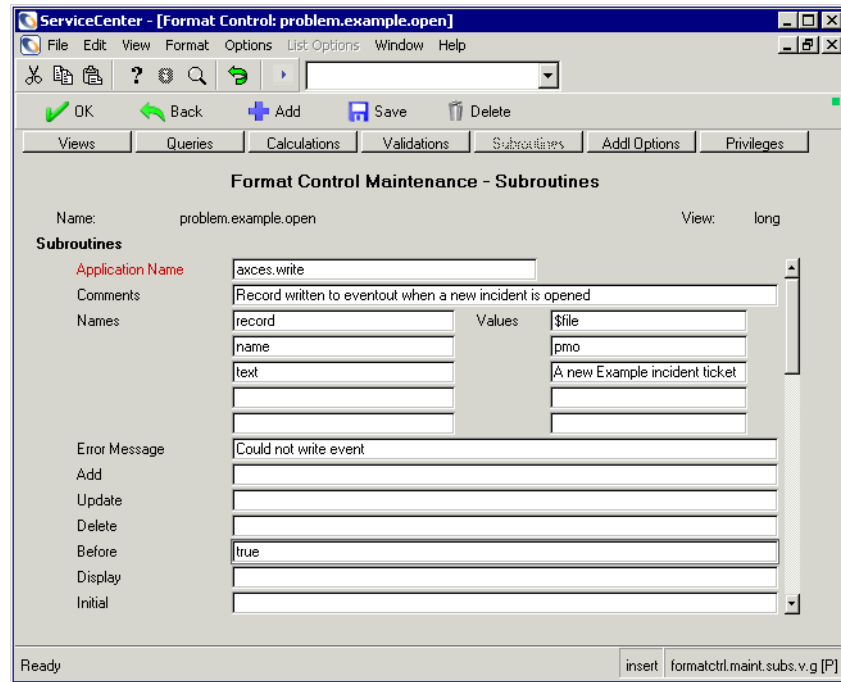
- If you define a separation character, make sure it is not one that occurs naturally in fields in the event.
- The system sequence ID will be generated by ServiceCenter unless you supply one. Its maximum length is sixteen (16) characters.

## Incident Management

When problems are opened, updated or closed by Event Services, a record may be written to the `eventout` file. This record contains information from the problem (described in the output `eventmap` record for the event) that is passed to an external process via the SCAuto/IPAS external interface. You can elect to write to the `eventout` file when Help Desk operators open and close tickets so that the information is passed to the external interface.

The `axces.write` application creates a character string of fields from a structure and writes them to `eventout`. An Event Registration record identifies the event type and the name of the Event Map records used to define which fields will be selected from the record. The application should be called as a Format Control subroutine passing two parameters - the first is the record from which data will be mapped, and the second is the Event Type, as defined in the Event Register. For example, to write an `eventout` record when an *example* type incident is opened, use the following parameters:

Parameter Name	Parameter Value
record	\$file
name	pmo



**Figure 8-1: Format Control Subroutine Setup—expanded form**

To write to the eventout file on *problem close*, the Format Control would be attached to the *problem.example.close* format. In each case, the subroutine is called if the condition for **add** returns *true*.

**Note:** The Incident Management category *example* writes an eventout record for each open, update and close action.

**Note:** The standard event, described in the *ServiceCenter SCAutomate for Windows NT and UNIX*, and *SCAauto for NetView OS/390 Guide*, requires that certain fields be populated in a particular position in the information passed to eventout:

- The first position is reserved for the email address.
- The second position is reserved for the incident number.
- The fourth position is reserved for a time stamp (such as problem open or problem close time).
- The eighteenth position is reserved for the logical name of the device.
- The thirty-fifth position is reserved for the network name of the device.



For standard events, these fields must be populated and must remain in their relative positions in the character string. The **eventmap** records for **output** define and maintain this information.

## Inventory and Configuration Management

When inventory items are added, updated or deleted by Event Services, a record may be written to the **eventout** file. This record contains information from the device record (described in the output **eventmap** record for the event) that will be passed to an external process via the SCAuto external interface. You can elect to write to the **eventout** file when operators maintain inventory items so that the information is passed to the external interface.

The **axces.write** application creates a character string of fields from a structure and writes them to **eventout**. An Event Registration record identifies the event type and the name of the Event Map records used to define which fields will be selected from the record. The application should be called as a Format Control subroutine passing two parameters - the first is the record from which data will be mapped, and the second is the Event Type, as defined in the Event Register. For example, to write an **eventout** record when a new device is added, use the following parameters:

Parameter Name	Parameter Value
record	\$file
name	icma

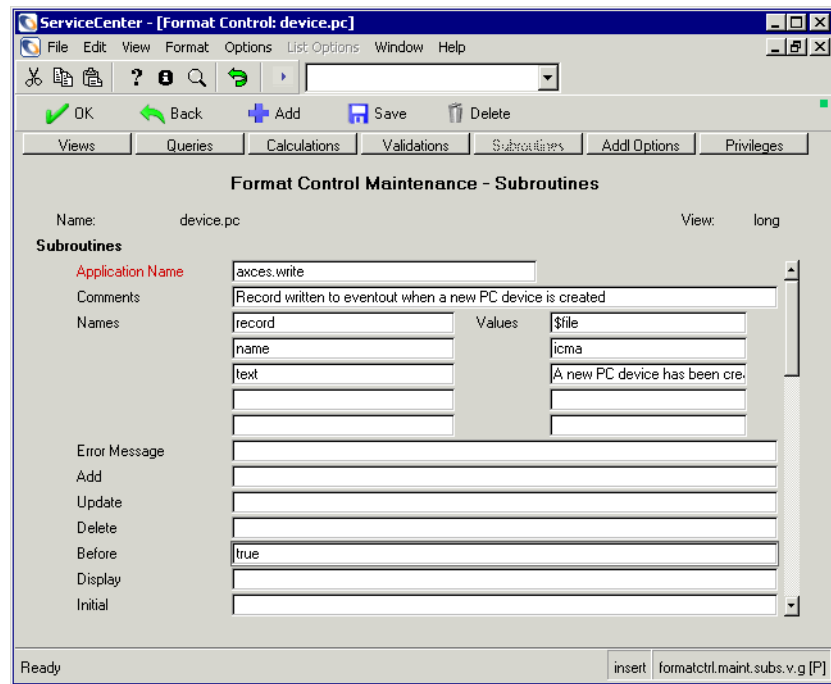


Figure 8-2: device.pc Format Control Subroutine

**Note:** The Inventory device type *example* writes an eventout record for each add, update and delete operation.

## Generating Page Messages

### Format Control

SCAutomate supports a generic **page** function. Page events can be written to the eventout file via a subroutine call to *axces.page* from Format Control.

Application	Description
axces.page	Called from Format Control, builds an eventout record used by the Telalert pager axces interface.

## Parameters

Name	Value	Default
name	The name of the contact	none
prompt	The numeric message	none
text	The alphanumeric message	none
string1	The separation character	^
query	The page response code	none
values	a list of addressees	none
names,1	a pager phone number	none
names,2	a pager PIN number	none
names,3	the name of a group	none

## Programming Considerations

- The name parameter or the names, 3 parameter or the values parameter or the names, 1 parameter is required. The application will exit if one of these parameters is not provided.
- If more than one of the name parameters (that is, name, values and names,3) is provided, all will receive a page as long as the associated contacts or operator record contains a pager phone number. Duplicate names will receive only one page.
- The output event substitutes "" whenever a field is NULL except where noted below.
- The output event concatenates fields from the contacts record as follows: Pager Vendor (telalert if NULL), Pager Name, Pager Group, Pager Type, Pager Phone #, Pager Pin #, Voice Mailbox, Numeric Message, Text Message. Fields are separated by the separation character.
- If a Pager Group is identified in the contacts record, the Pager Phone # is not passed.
- The page event is written directly to the eventout file.
- The group referred to by the names,3 parameter is defined in the *distgroup* file with a type of *page*.

- While you can simply pass a pager phone number and a message to `axces.page`, usually a contact or operator name is provided since the pager instructions are stored in the contacts file.
- If there is no record in the contacts or operator file matching the value passed in the contacts parameter (or one of the entries in the values parameter, or one of the operators defined in the group named in the `names,3` parameter), a page event will not be processed. There are fields in the contacts file that define pager vendor, phone number, PIN, etc. These fields must be completed properly for successful paging to occur.
- If a parameter is passed in the query parameter, it will be used by the `pageresp` input event to identify what type of event processing should occur. For example, to update a particular problem with the response from a page, pass `pm` and the problem number (for example, `pm9700123`). The registration record determines the application to call by examining the data in the first position of the `evfields` field.

## Incident Management

Format Control is used to determine rules for sending a page when opening, updating or closing problems. For testing purposes, the category called *example* sends a page upon problem open if the Contact Name field is completed. To extend the service to other categories (or upon update, close or alert), access their associated Format Control and copy information from the *problem.example.open* Format Control record's subroutine definition for `axces.page`. For instance, if you want to page the Contact Name when a software problem reaches each alert stage, copy the `axces.page` subroutine definition from the *problem.example.open* Format Control record to the *problem.software.alerts* Format Control record.

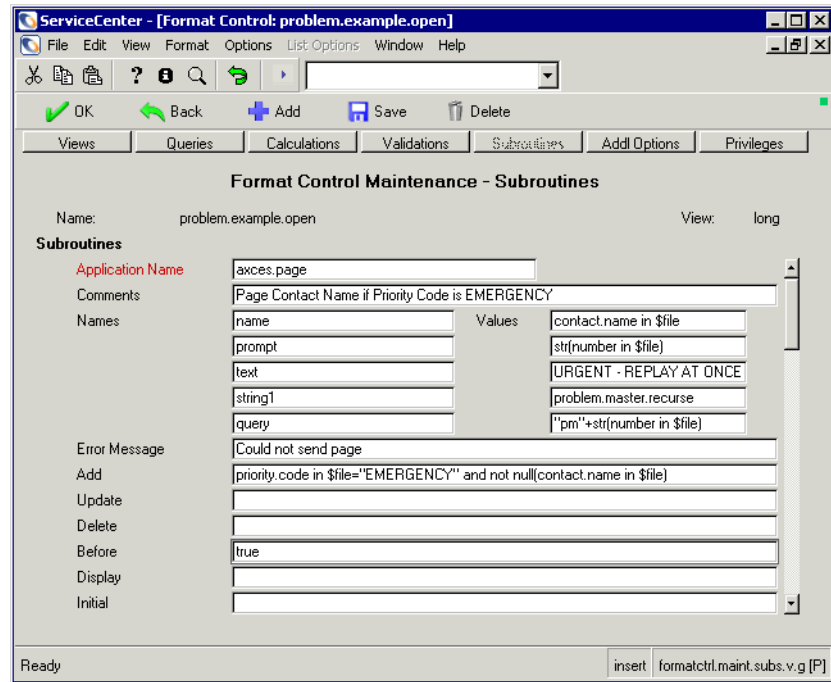


Figure 8-3: problem.example.open Format Control Subroutine

## Sending Fax Messages

SCAutomate supports a generic **fax** function using the Replix FAX product. Fax events can be written to the `eventout` file via a subroutine call to `axces.fax` from Format Control or from the Send a FAX button on the Event Services menu. You can also send any report or any mail message as a fax.

To support report Fax output, a record of type **FAX** must exist in the ServiceCenter `config` table. This record is used to limit the number of pages sent by a fax message. You must supply the device name at the time the report (or printout) is generated. Fax messages generated from the Send a Fax button or from ServiceCenter mail, or via Format Control, do not require a config record. By definition, their size cannot exceed 32,000 bytes.

### Format Control

Application	Description
<code>axces.fax</code>	Called from Format Control, builds an <code>eventout</code> record used by the Replix FAX <code>axces</code> interface.

#### Parameters

Name	Value	Default
<code>names,1</code>	The name of the sender	none
<code>name</code>	The name of the recipient	none
<code>prompt</code>	The FAX phone number	none
<code>string1</code>	The separator character	^
<code>query</code>	The name of the company	none
<code>text</code>	The format name or text string	none
<code>names,2</code>	The FAX title	none
<code>record</code>	The record variable	none

## Programming Considerations

- The name or prompt parameter is required; the application will exit if one of these parameters is not provided.
- If the contacts file is searched for a record with contact.name equal to the value passed in name. If no record is found, or if the selected record does not have a fax number defined, the fax is not sent.
- If a record variable is passed in the record parameter, pass the format name in the text parameter. The application uses *genout()* to build the fax output. Alternatively, you can pass a string in text; the string must use the pipe symbol (|) to separate lines of text.
- The output is written directly to the eventout table.

## Creating Output Events

### Format Control

You can use ServiceCenter's Format Control processing to create output events based upon business rules. These events include paging, sending email messages and sending Fax documents. For more complete information and examples of Format Control utilities within the ServiceCenter and SCAutomate environments, please refer to the *Format Control* section of *System Tailoring, Volume 1*.

Application	Description
axces.fax.msg	Called from Format Control, builds schedule record that sends a fax.

### Parameters

Name	Value	Default
file	A completed mail record	none
boolean1	The background flag	false

## Programming Considerations

- You must pass only a mail record to this application. In Format Control you can set one up using secondary queries and using a query of false.
- The file parameter is required; the application will exit if this parameter is not provided. Pass the file variable that contains the mail record.
- The user.array field in the file variable must be populated with at least one name.
- Both the contacts and the operator tables (in that order) are searched for each name in the user.array field; if no fax number is defined in the selected record (or if no record is selected) and the background flag is false a prompt will allow you to enter the recipient name and telephone number.
- A separate fax will be sent to each name in the user.array field.
- Records are added to the spool file, and the background spool scheduler uses runoff to add records to the eventout file.
- A FAX config record must exist.
- The runoff application must have a compile date later than 5/14/96; reference SCR 7343.



# A Basic Troubleshooting

## APPENDIX

If you have followed all the directions and are still encountering issues with the SCAutomate implementation, refer to the following common questions. Check these items and resolutions before contacting Peregrine Customer Support.

## Frequently Asked Questions

### Why are no problems opening, even though there are pmo records in the Event Input queue?

- 1 Verify the records in the queue been have processed.
  - If the records have been processed, there should be no **Event Time** value.
  - The **Status** field should contain a value.
  - Any messages should appear in the **Messages** field.
- 2 Verify there is an active event agent.
  - a Click Agent Status on the Services tab of the Event Services menu.
  - b Open the event agent.
    - The **Stop** button should be enabled.
    - A **Start Time** and an **Idle Time** should be displayed.

- c Click **Refresh** to reset idle time to 00:00:00. It should begin increasing again.
  - d If the **Start** button is enabled and there is no **Start and Idle Time**, click **Start** and wait until the *problem* agent recycles.
- 3 Verify the following, and then wait for the event processor to recycle:
  - a The event schedule record exists.
  - b The **Class** field has a value of event.
  - c The **Status** field has a value of rescheduled.
- 4 If there is an active event agent, check the Event Registration table.
  - Are there entries for Event *pmo* with a Type of *input*?
  - Is the Execute Condition *true*?
  - Compare the content of the *pmo* registration to the values documented in *Reviewing Event Registration* on page 40..
- 5 Verify there are event maps matching the **Event Map Name** values in the registration record.
  - The same rules apply to all event types, not just *pmo*.
- 6 Verify that an active category is provided.

### **Why am I not receiving email even after installing ServiceCenter and opening a problem?**

- 1 Verify you are a member of the assignment group for the problem.  
If not, you will not receive notification of any kind.
- 2 Determine whether you are attempting to send mail to yourself when you open a problem.  
ServiceCenter does not send mail to the individual who is opening, updating or closing a problem, regardless of their membership in the assignment group.
- 3 Log on as someone else.
- 4 Open a new problem.
- 5 Determine whether the operator to whom you are sending mail has an email address specified in his or her operator record.
- 6 Make sure it is correct.

- 7 Check the **Message Class** file for External Email records.
  - Is there one for *problem open*?
- 8 If not, add one.
- 9 Verify there are records in the event output queue with a **type** of *email*.
- 10 If so, determine whether the scemail agent or another email agent is active.
  - a Click **Agent Status** on the Services tab of the Event Services menu.
  - b Open the Event agent.

The Stop button should be enabled, and a **Start Time** and an **Idle Time** should be displayed.
  - c Click **Refresh** to reset idle time to 00:00:00. It should begin increasing again.
  - d If the **Start** button is enabled and there is no **Start** and **Idle Time**, click **Start** and wait until the *problem* agent recycles.
- 11 Determine if there is an output type event registration record for *email*.
- 12 Compare its contents to those described in *Reviewing Event Registration* on page 40.
- 13 If the SCEMAIL agent or another email agent is active and you still do not receive mail, kill the agent.
- 14 Open a problem and check the event output queue for new events with a type of *email*.
- 15 If a new email event is added to the queue, restart the SCEMAIL agent or another email agent.

When the mail has been sent, the event will be either deleted (if the -d flag is set) or updated.

---

**Important:** Always check the ServiceCenter Message Log and any external log files for errors. All SCAutomate errors are logged with a class of *event management errors*.

---

## How do I send email only when I open problems with a priority code of “emergency”?

- 1 Click the **Administration** button on the Utilities tab of the ServiceCenter main menu.
- 2 Click the **External email** button to open the message class file.
- 3 Remove any External Email record for *problem open*.
- 4 Return to the ServiceCenter main menu.
- 5 Click the **Tools** button on the Utilities menu.
- 6 Click the **Macro** button.
- 7 Search for the incidents macro that sends the email.
- 8 Change the **Condition** field value to:  
`nullsub(priority.code in $L.new, “”)="1”`

## How do I know mail sent to myself was received?

- 1 From the ServiceCenter main menu, click the Mailbox icon.
  - 2 Click **Read Mail**.
  - 3 Click **All Mail**.
- Your message should appear in the list of mail messages.

## How do I quickly test sending a fax message?

- 1 From the Services tab of the Event Services Menu, click **Send a FAX**.
- 2 Complete a message.
- 3 Click **FAX**.
- 4 Review the event output queue for an event of type *fax*.

## How do I quickly test whether the SCAuto Pager is properly installed?

- 1 From the Services tab of the Event Services Menu, click **Send a Page**.
- 2 Complete a message.
- 3 Click **Page**.
- 4 If you are not paged within a minute or two, check to make sure the SCAUTO agent is active. Use the following procedure to do so:
  - Click the **Status** button on the ServiceCenter main menu.  
There should be an entry under **User Name** for SCAUTO.

- 5 If SCAUTO is not active, you can start it if you are running an express client or are logged on from the server using *scenter*.
  - a Click **Start Scheduler**.
  - b Click on the entry for *scauto.startup*.
  
- 6 If the SCAUTO agent is active and you still do not receive a page,
  - a kill the agent, by placing a **k** in the command column beside agent.
  - b Click **Execute Commands**.
  - c Send a new page and check the event output queue for new events with a type of *page*.
  
- 7 If a new page event is added to the queue, restart the SCAUTO agent.

When the page has been sent, the event will be either deleted (if the **-d** flag is set) or updated.

### **How do I test sending a problem to my external program once SCAuto/SDK is installed?**

- 1 From the Services tab of the Event Services Menu, click **Write an Output Event**.
- 2 Click **Problem**.

The first record in the *probsummary* file is written to the eventout queue.
- 3 Open a problem using the category **example**.
- 4 Note the problem number.
- 5 Open the Event Output queue.
- 6 Search for an event with a **Type** of *pmo* and a **Fields** field beginning with ^ followed by the problem number of the problem created in step 3.

## How do I test sending a new device to my external program once SCAuto/SDK is installed?

- 1 From the Services tab of the Event Services Menu, click **Write an Output Event**.

- 2 Click **Inventory**.

The first record in the device file is written to the eventout queue.

- 3 Add a new device of type **example**.
- 4 Note the Logical Name.
- 5 Open the Event Output queue.
- 6 Search for an event with a **Type** of *icma* and a **Fields** field beginning with the Logical Name added in step 3.

### How do I set the category from my message when I am opening problems via email?

- 1 Put each field assignment on a separate line in your mail message, uniquely identified by a label.
- 2 Use mapping expressions to extract the information and populate the appropriate fields in the problem

Example:

The mail message looks like this:

Fri, 12 Jan 01 14:40:41 -08:00

Re: Test to assign a category

John Jones <john@mac.acme.com>

CATEGORY: example

This is line 1 of the text of mail.

This is line 2 of the text of mail.

In the eventin record, the **evfields** field should appear as follows:

```
xjohn^^^^Fri, 12 Jan 01 14:40:41 -08:00|Re: Test to assign a category|John Jones<john@hp800.peregrine.com>|CATEGORY: example||This is line 1 of the text of mail.|This is line 2 of the text of mail.|^^^^^^^^^^^^^^^^John Jones <john@mac.acme.com>^^
```

- In the *problem open* event map record for the **category** field, enter the following *Initialization* statements:

```
$axtype=type in $axces.target
if (index("axmail", evuser in $axces)>0) then $axtype=type in $axces.target
if (index("axmail", evuser in $axces)>0) then ($ax.action=denu((action in
$axces.target);$axl=lng($ax.action))
if (index("axmail", evuser in $axces)>0) then for $axpos = 1 to $axl do
($axt=$axpos in $ax.action;if $axt#"CATEGORY" then ($axtype=substr($axt,
10, lng($axt) - 9);$ax.action=delete($ax.action, $axpos);action in
$axces.target=$ax.action))
```

- 3 Then enter these Instructions:

```
if (index("axmail", evuser in $axces)>0) then category in $axces.target=$axtype
cleanup($axtype);cleanup($axt);cleanup($axpos);cleanup($axl)
cleanup($ax.action)
```

This procedure (substituting other field names) allows specification of any problem field values within the body of the email message as long as the map record in which the instructions are entered has a higher sequence number than that of the **action** (or **update.action**) field.

### **Can I have my problem events processed separately, so they aren't held up by other events?**

- 1 Copy the event agent to a new agent called (for example) *probevent*.
- 2 Copy its associated info record, substituting *probevent* for *event*.
- 3 Modify the event agent's query field to say *evtype~#"pm"*.
- 4 Modify the *probevent* agent's query field to say *evtype#"pm"*.

You can do the same thing for output events created by the SCAuto/OS/390 agents.





# Index

## Symbols

\$attribute.file 167, 168

## A

accessing

GUI client 21–26

text client 26

agents 23

starting and stopping 210

applications

aces.fax 244

aces.page 240

aces.problem 45

aces.write 237, 239

mail, changes to 223–224

user-defined 153

aces 171

aces.fax 244

aces.field 162, 168, 171

aces.fields 169, 171

aces.notriml 172

aces.notrimr 172

aces.page 240

aces.register 171

aces.scource 171

aces.target 168, 171

aces.write 237, 239

## B

block condition 187

blocking of events 180, 183–188

## C

Change Management

and events 15

approval fields 200

EIS approval actions 200

event examples 203–204

input events 198–200

input events, external information string  
199–200

input events, registrations 198

launching ND from 190

output events 202

using with external systems 197

character, delimiter 30, 37, 46, 176

cm3rin 103

cm3rinac 105

cm3rout 105

cm3tin 106

cm3tinac 106

cm3tout 107

code, event 30, 40, 41, 198

components 18–19

condition, block 187

Connect.It 15, 17

contacts file, for validating mail 223–224

containers, OLE 36

CTSCPYP

input event 49

output event 50

- CTSIMP
    - input event 50
    - output event 51
  - CTSIMP2 52
  - CTSRQCLS
    - input event 53
  - CTSRQOPN
    - 1 54
    - 2 55
    - output event 56
  - CTSRQUPD
    - input event 57
    - output event 58
  - CTSTKCLS
    - input event 58
    - output event 58
  - CTSTKOPN
    - 1 60
    - 2 61
    - output event 60
  - CTSTKUPD
    - input event 62
    - output event 63
- D**
- dbadd 107
  - dbdel 108
  - dbupd 109
  - delimiter character 30, 37, 46, 176
  - device records
    - information 193
- E**
- education services 13
  - EIS 199
  - EIS (External Information String) 199–200
    - approval fields 200
    - data fields 200
  - email 15, 46
    - default address 223–224
    - events 220
    - events, creating in RAD 220
    - interface 16
    - output event 110
    - receiving ServiceCenter mail 220–223
    - sample events 36
    - sending 23
    - sending with emergency priority 250
    - sending, Format Control 231
    - sending, Incident Management 232–233
    - troubleshooting 248–249
  - epmc
    - input event 111
    - output event 112
  - epmo
    - input event 112
    - output event 113
  - epmu
    - input event 115
    - output event 116
  - ERP Interface 19
  - ERPHR
    - input event 64
    - output event 64
  - ERPSTATES
    - input event 65
    - output event 65
  - event code 30, 40, 41, 198
  - event filters
    - reviewing 180
  - event maps
    - form 160
  - event register file 40
  - event registration
    - fields 41–44
  - event scheduler
    - in Event Services flow 20
    - using 206
  - Event Services
    - components 18–19
    - output events, fields 38–39
  - eventin 198
    - VSAM events 209
  - eventout records
    - EIS fields 201
    - generating, Format Control 236–237
    - generating, Incident Management 237–239
    - generating, Inventory Management 239–240
    - VSAM events 209

- events
  - approval 103, 104
  - as background task 170
  - blocking of 180, 183, 188
  - cm3rin 104
  - cm3rinac 105
  - cm3rout 105
  - cm3tin 106
  - cm3tinac 106
  - cm3tout 107
  - commonly-used 47–49
  - CTSCP, input 49
  - CTSCP, output 50
  - CTSIMP, input 50
  - CTSIMP, output 51
  - CTSIMP2, output 52
  - CTSRQCLS, input 53
  - CTSRQOPN, input 54, 55
  - CTSRQOPN, output 56
  - CTSRQUPD, input 57
  - CTSRQUPD, output 58
  - CTSTKCLS, input 58
  - CTSTKCLS, output 58
  - CTSTKOPN, input 60, 61
  - CTSTKOPN, output 60
  - CTSTKUPD, input 62
  - CTSTKUPD, output 63
  - dbadd 107
  - dbdel 108
  - dbupd 109
  - defined 15
  - email 36
  - email, output 110
  - epmc, input 111
  - epmc, output 112
  - epmo, input 112
  - epmo, output 113
  - epmosmu 114, 115
  - epmu, input 115
  - epmu, output 116
  - ERPHR, input 64
  - ERPHR, output 64
  - ERPSTATES, input 65
  - ERPSTATES, output 65
  - esmin 117
  - fax messages 153
  - filtering 19, 25, 180–188
  - filtering of 25, 180, 188
  - filtering, blocking 180, 183–188
  - filtering, fields 181–183
  - generic administration 154–155
  - GetResRM 70, 71
  - GetResRML 71
  - gie 118
  - HotNews 72
  - icma 119, 121
  - ICMapplication 66
  - ICMcomputer 66
  - icmd 120
  - ICMdevice 67
  - ICMdisplaydevice 68
  - ICMexample 68
  - ICMfurnishings 68
  - ICMhandhelds 68
  - ICMmainframe 69
  - ICMnetworkcomponents 69
  - ICMofficeelectronics 69
  - ICMserver 69
  - ICMsoftwarelicense 69
  - ICMstorage 70
  - icmswa 121
  - icmswd 121
  - ICMtelecom 70
  - icmu 122
  - ICMworkstation 70
  - incident tickets 193
  - IND 72
  - input 16, 24, 30–37
  - input, fields 31–36
  - input, processing 36–37
  - mapping 19, 25, 158–179
  - mapping of 25, 158, 179
  - mapping, creating a map 173–179
  - mapping, creating an ICM map 178–179
  - mapping, expressions for non-scalar fields 177
  - mapping, fields 160–164
  - mapping, Inventory Management 172
  - mapping, ND device information 194
  - mapping, ND-detected problems 195

- mapping, rules for creating 176–177
- mapping, use of 176
- mapping, using 165–171
- mblpmc 127
- mblpmo 128
- mblpmu 129
- mlbcm3tc 123
- mlbocmlc 125
- mlbocmlu 126
- ND (Network Discovery) 193
- NDpmc 73
- NDpmo 74
- networkcomponents 69
- opera 130
- operd 131
- operu 132
- outageend 133
- outagestart 133
- output 16, 24, 37, 39
- output fields 38–39
- page 133, 153
- pageclose 134
- pageresp 135
- pcsoftware 136
- pmc
  - input 137
  - output 138
- pmo 45
- pmo, input 139
- pmo, output 140
- pmu, input 141
- pmu, output 142
- prgma 142
- prgmd 143
- prgmu 144
- PSSDELETE 75
- registration 40–46
- registration of 19, 25, 40, 46
- registration, global variables 153
- rmlin 145
- rmoappr 145
- rmoin 146
- rmqappr 146
- rmqin 147
- SALESQUOTE 75
- SAPGRT 75
- SAPGRT, input 76
- SAPGTE 76
- SAPHR, input 77
- SAPHR, output 78
- SAPHRMD 78
- SAPORD, input 79
- SAPORD, output 79
- sapordl 147
- SAPORDQ, header 80
- SAPQTE, input 81
- SAPQTE, output 80
- sapqtel
  - output 148
- sapqtel, input 148
- SAPQTEQ, header 81
- saprecl 149
- SAPREQ, input 82
- SAPREQ, output 82
- sapreql, input 149
- sapreql, output 149
- SAPREQO, component 82
- ScAcBrand 83
- ScAcCompany 84
- ScAcContacts 85
- ScAcDept 86
- ScAcDevice 87
- ScAcLocation 88
- ScAcModel 89
- ScAcModelBundle 90
- ScAcModelVendor 91
- ScAcVendor 92
- ScAcVendorBACK 93, 94
- ScFcOrderLine 94
- slaresponse 150
- smin 150
- smout 151
- statuses 32
- submit 151
- sysbull 152
- TcScCompany 95
- TcScCompDel 95
- TcScContacts 96
- TcScDept 97
- TcScDeptDel 98

- TcScDeptdel 99
- TcScLocDel 99
- type 181
- WMI 101, 102
- XIND 103
- evexpire 209
- evfields 242
- evgoto 44
- External Information String (EIS) 199–200
- external system, synchronizing SC with 201, 202

## F

- fax messages 23, 153, 244–245
  - Format Control, sending with 244
  - testing 250
- fields
  - evfields 242
  - mapping 160, 164
- files
  - event register 40
  - eventfilter 180
  - eventin 30, 36, 158, 176
  - eventmap 36, 158, 165
  - eventout 37, 39, 158
  - eventregister 37, 40
  - mail 165
  - schedule 206
- filtering of events 25, 180–188
  - fields 181–183
- flowchart
  - process flow 20
  - workflow from external sources 17–18
- Format Control
  - eventout records 236–237
  - fax messages, sending with 244
  - mapping considerations for ICM 172
  - output events, creating with 245–246
  - page events, creating with 240–242

## G

- Get.It! 15, 17
- gie 118
- global variables
  - \$attribute.file 167, 168
  - \$axces 171

- \$axces.field 162, 168, 171
- \$axces.fields 169, 171
- \$axces.notriml 172
- \$axces.notrimr 172
- \$axces.register 171
- \$axces.source 171
- \$axces.target 168, 171
- registration events 153

## I

- icma 119
- ICMapplication 66
- ICMcomputer 66
- icmd 120
- ICMdevice 67
- ICMdisplaydevice 68
- ICMexample 68
- ICMfurnishings 68
- ICMhandhelds 68
- ICMmainframe 69
- ICMnetworkcomponents 69
- ICMofficeelectronics 69
- ICMserver 69
  - input event 69
- ICMsoftwarelicense 69
- ICMstorage 70
- icmswa 121
- icmswd 121
- ICMtelecom 70
- icmu 122
- ICMworkstation 70
- Incident Management
  - creating eventout records 237–239
  - events 15
  - ND, launching 190
  - ND, opening and closing tickets 194–195
  - page events, creating 242–243
- incident tickets
  - event information 193
- input events 16, 24
  - Change Management 198–200
  - Change Management, external information string 199–200
  - Change Management, registrations 198
  - CTSCPYPY 49

CTSIMP 50  
 CTSRQCLS 53  
 CTSRQUPD 57  
 CTSTKCLS 58  
 CTSTKUPD 62  
 epmc 111  
 epmo 112  
 epmu 115  
 ERPSTATES 65  
 examples 203  
 fields 31–36  
 OLE containers 36  
 processing 36–37  
 SAPGRT 76  
 SAPHR 77  
 SAPORD 79  
 SAPQTE 81  
 SAPREQ 82  
 Interface, ERP 19  
 Inventory and Configuration Management  
     creating a map 178–179  
 Inventory Management  
     and events 15  
     eventout records, creating with 239–240

## L

listeners 17

## M

maintenance, vsam 25  
 mapping  
     events, creating a map 25, 173–179  
     events, ND detected problems 195  
     events, ND device information 194  
 mapping arrays of structures 163  
 mapping fields 160–164  
 mapping of events 25, 158–179  
     creating an ICM map 178–179  
     Inventory Management 172  
     rules for creating 176–177  
     using maps 165–171  
 messages, fax 23, 153, 244, 245

## N

NAPA 15, 16  
 napainfo record 216  
 ND (Network Discovery)  
     device records 193  
     events 193  
     features 190  
     Incident Management, opening and closing  
         tickets 194–195  
     launching 190  
     modules 191–193  
     ServiceCenter integration 190–193  
 NDpmc 73  
     input event 73  
     output event 73  
 NDpmo 74  
     input event 74  
     output event 74

## O

OLE containers 36  
 opera 130  
 operator file  
     for validating mail 223–224  
 operd 131  
 operu 132  
 outageend 133  
 outagestart 133  
 output events 16, 24, 37–39  
     Change Management 202  
     creating with Format Control 245–246  
 CTSCPY 50  
 CTSIMP 51  
 CTSRQOPN 56  
 CTSRQUPD 58  
 CTSTKCLS 58  
 CTSTKOPN 60  
 CTSTKUPD 63  
 epmc 112  
 epmo 113  
 epmu 116  
 ERPHR 64  
 ERPSTATES 65  
 examples 204  
 fields 38–39

SAPHR 78  
 SAPORD 79  
 SAPQTE 80  
 SAPREQ 82

## P

page events  
   Format Control 240–242  
   Incident Management 242–243  
   sending 19, 23, 133, 153  
 pageclose 134  
 pageresp 135  
 PeopleSoft 19, 26  
 pmc 137, 138  
 pmo 139, 140  
 pmo events 45  
 pmu 141, 142  
 prgma 142  
 prgmd 143  
 prgmu 144  
 products  
   Connect.It 15  
   Get.It! 15  
   NAPA 15, 16  
   SCAutomate 15  
 program 39  
 properly 200  
 Purge/Archive 37, 39

## R

RAD  
   creating email events 220  
   subroutines, axces.fax 244  
   subroutines, axces.page 240  
   subroutines, axces.write 237, 239  
 record, napainfo 216  
 register file, event 40  
 registration of events 25, 40–46  
 rmlin 145  
 rmoappr 145  
 rmoin 146  
 rmqappr 146  
 rmqin 147

## S

SAP 19, 26  
 SAPGRT  
   input event 76  
 SAPGTE 76  
 SAPHR  
   input event 77  
 SAPHR output event 78  
 SAPHRMD 78  
 SAPORD  
   input event 79  
   output event 79  
 sapordl 147  
 SAPORDQ 80  
 SAPQTE  
   input event 81  
   output event 80  
 sapqtel 148  
 SAPQTEQ 81  
 saprecl 149  
 SAPREQ  
   input event 82  
   output event 82  
 sapreql 149  
 SAPREQO 82  
 SCAuto  
   event schedules 206  
   pager 250  
 SCAuto/SDK 251–253  
 SCAutoMail 220  
 SCAutomate 15  
 SCemail 17, 220, 224–231  
   compared with SCAutoMail 220  
   optional parameters 227–228  
   OS/390 229–231  
   profile, adding 225  
   starting 226  
   Unix 228–229  
   Windows NT 224–225  
 schedule 206  
 scheduler  
   VSAM 215  
   vsam.write 209  
 SCMail 17  
 SCMail 17

- SCPager 17
- ServiceCenter
  - applications defined 12
  - knowledge requirements 12
  - mail, sent to email 220–223
  - overview 12
- slaresponse 150
- smin 150
- smout 151
- splayed 195
- submit 151
- synchronizing SC with external system 201–202
- sysbull 152

## T

- training services 13
- troubleshooting
  - events 247–253
- type, event 181

## V

- validating
  - email addresses 223–224
- vsam 17, 25
  - vsaminfo 25
- vsam maintenance 25
- VSAM scheduler 215
- vsam.write scheduler 209
- vsaminfo 214, 216





