

# HP Service Manager

for the Windows and Unix operating systems

Software Version: 9.30

---

## Event Services

Document Release Date: Sept 2011  
Software Release Date: July 2011



## Legal Notices

### Warranty

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

The information contained herein is subject to change without notice.

### Restricted Rights Legend

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

### Copyright Notice

© Copyright 2004-2011 Hewlett-Packard Development Company, L.P.

### Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Java is a registered trademark of Oracle and/or its affiliates.

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

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

UNIX® is a registered trademark of The Open Group.

## Documentation Updates

The title page of this document contains the following identifying information:

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

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

**<http://h20230.www2.hp.com/selfsolve/manuals>**

This site requires that you register for an HP Passport and sign in. To register for an HP Passport ID, go to:

**<http://h20229.www2.hp.com/passport-registration.html>**

Or click the **New users - please register** link on the HP Passport login page.

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

## Support

Visit the HP Software Support Online web site at:

**[www.hp.com/go/hpsoftwaresupport](http://www.hp.com/go/hpsoftwaresupport)**

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

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

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport user ID, go to:

**<http://h20229.www2.hp.com/passport-registration.html>**

To find more information about access levels, go to:

**[http://h20230.www2.hp.com/new\\_access\\_levels.jsp](http://h20230.www2.hp.com/new_access_levels.jsp)**

# Contents

- 1 Introduction** ..... 9
  - How Event Services works ..... 10
    - Event Services files ..... 11
  - Event Services flowchart ..... 12
  - Accessing Event Services ..... 13
- 2 Standard Event Operations** ..... 15
  - Event registration ..... 16
    - Reviewing event registration ..... 16
    - Global variables ..... 21
  - Input events ..... 22
    - Input event processing ..... 26
  - Output events ..... 28
    - Output fields ..... 29
- 3 Mapping and Filtering** ..... 31
  - Mapping ..... 32
    - The event map form ..... 32
    - Using event maps ..... 36
    - Global variables ..... 42
    - Mapping considerations for Inventory Management ..... 42
    - Building a new event map ..... 43
  - Event filters ..... 47
    - Header fields ..... 48
    - Blocking ..... 51
- 4 Change Management Event Services** ..... 55
  - Input events ..... 56
    - Input event registrations ..... 56
    - Setting up the external information string ..... 57
  - Synchronizing HP Service Manager with an external system ..... 59
    - Acknowledgments ..... 59
    - Sending complete output events ..... 60
  - Change event examples ..... 61
    - Input examples ..... 61
    - Output example ..... 61

<b>5</b>	<b>SCemail</b> .....	63
	Email events .....	64
	SCemail vs. SCAutoMail .....	65
	Event Services and HP Service Manager email .....	66
	Emailout Parameters in the sm.ini File .....	67
	SCEmail Background Processor .....	68
	Event Services and Format Control .....	69
	Writing an Incident Management eventout record .....	69
	Writing a Configuration Management eventout record .....	69
	Sending HP Service Manager mail to email .....	70
<b>6</b>	<b>Event Agent Operations</b> .....	71
	Event scheduling .....	72
	Reviewing scheduled events .....	72
	Maintaining agent status .....	74
	System startup .....	74
	System status window .....	75
<b>7</b>	<b>Format Control Options</b> .....	77
	Generating eventout records .....	78
	Format Control .....	78
	Incident Management .....	78
	Inventory and Configuration Management .....	79
	Generating page messages .....	81
	Format Control .....	81
	Incident Management .....	82
	Sending fax messages .....	83
	Format Control .....	83
	Creating output events .....	85
	Format Control .....	85
<b>A</b>	<b>Troubleshooting</b> .....	87
	Why are no incidents opening, even though there are pmo records in the Event Input queue? .....	87
	Why is email not being received after opening a incident? .....	89
	How do I send email only for those incidents with a priority code of “emergency”? .....	90
	How do I know HP Service Manger mail sent to myself was received? .....	90
	How do I quickly test sending a fax message? .....	91
	How do I quickly test whether the SCAuto Pager is properly installed? .....	91
	How do I test sending a report to my external program once SCAuto/SDK is installed? .....	91
	How do I allow incident events to be processed separately, so they aren’t held up by other events? ..	92
	How do I test notifications to external programs after installing SCAuto .....	92
<b>B</b>	<b>Common Events</b> .....	93
	Service Desk events .....	94
	Incident Management events .....	95
	Inventory Management events .....	96
	Change Management events .....	97

Request Management events	98
Service Level Management events	99
Standard events	100
CERPHR (1)	100
ERPHR (2)	100
ERPSTATES (1)	101
ERPSTATES (2)	101
HotNews	101
ICMapplication	102
ICMcomputer	102
ICMdevice	103
ICMdevicenode (1)	103
ICMdevicenode (2)	104
ICMdisplaydevice	104
ICMexample	104
ICMfurnishings	105
ICMhandhelds	105
ICMmainframe	106
ICMnetworkcomp	106
ICMofficeelec	107
ICMserver	107
ICMsoftwarelicense	108
ICMstorage	108
ICMtelecom	109
IND	109
PSSDELETE	110
SALESQUOTE	110
ScAcBrand	111
ScAcCompany	111
ScAcContacts	112
ScAcDept	112
ScAcDevice	113
ScAcLocation	113
ScAcModel	114
ScAcModelBundle	114
ScAcModelVendor	115
ScAcVendor	115
ScAcVendorBACK	116
approval	116
approval	117
cm3rin	117
cm3rinac	117
cm3rout	118
cm3tin	118
cm3tinac	118
cm3tout	118
dbadd	119

dbdel	119
dbupd	120
email	120
email	120
epmosmu	121
epmosmu	121
esmin	122
esmin	122
gie.	122
icma	123
icmd	123
icmswa	124
icmswd	124
icmu	124
opera	125
operd	126
operu	126
outageend	127
outagestart	127
page	128
pageclose	128
pageresp	129
pcsoftware	129
pmc.	130
pmc.	130
pmo	131
pmo	132
pmu	132
pmu	133
prgma.	133
prgmd.	134
prgmu	135
rmlin	135
rmoappr.	136
rmoin	136
rmqappr.	136
rmqin	137
slaresponse	137
smin	138
smout	138
submit	139
sysbull	140



# 1 Introduction

HP Service Manager Event Services provides a bi-directional interface between HP Service Manager and external systems. It is primarily used to interface with the SCAutomate products.



HP Service Manager web services provides additional functionality related to interfacing with external systems. For more information about web services, see the HP Service Manager Web Services Guide.

Some of the products using HP Service Manager Event Services are:

- SCEmail—allowing HP Service Manager to send email to the world
- SCMail (Unix) and SCMapi (Windows)—HP Service Manager two-way email
- SCAutomate products—third-party products to tie HP Service Manager to products such as Tivoli, HP OpenView, and so on
- Custom-written SCAutomate applications
- Connect-It—uses Event Services inbound only, but for outbound operations can read HP Service Manager directly

To accomplish the communication between products, you must establish a connection between HP Service Manager and the external system. The type of connection is dependent on the product and environment involved. In most cases, some type of TCP/IP connection is used, often involving the HP Service Manager listener.

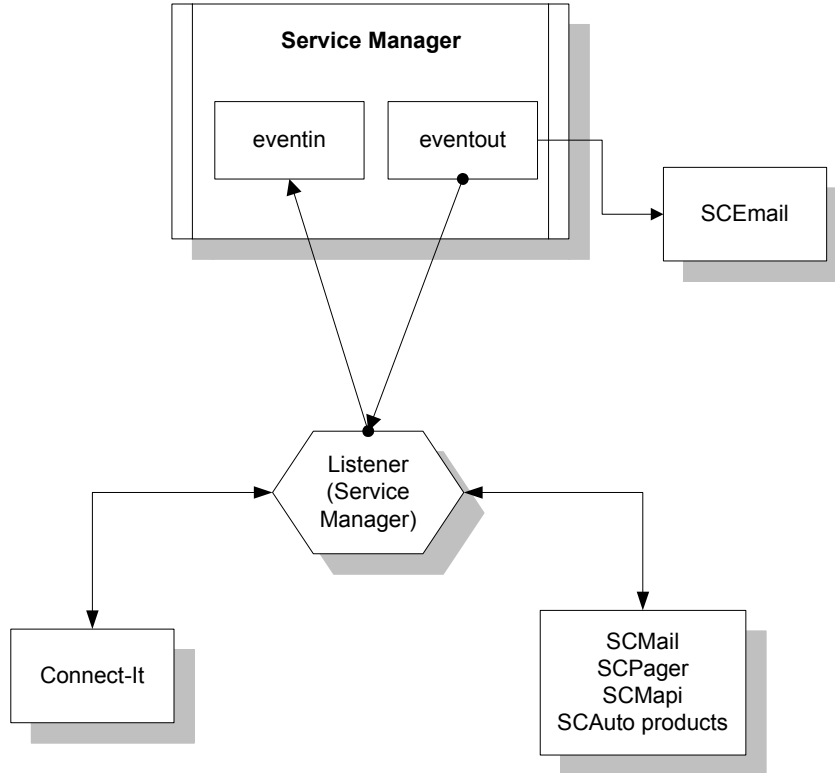
Once information comes into HP Service Manager, Event Services provides a series of standard applications to:

- Open, update, and close incidents.
- Open, update, and close calls.
- Add, update, and delete inventory items.
- Open, update, approve, and close changes and requests.

Similarly, standard applications are available for use within HP Service Manager to generate outbound information (such as emails). The standard applications come with predefined formats for the information. Through tailoring, you can change these formats and operations. The product is extensible; you can modify HP Service Manager Event Services to perform virtually any HP Service Manager operation on any table within the product.

## How Event Services works

Events entering and exiting HP Service Manager are routed differently depending upon the external system with which HP Service Manager is communicating. For some products, information is routed in one direction only. For others, events flow in both directions:



The following table describes the routing of events through external products currently supported:

File	Description
SCEmail	Outbound events only. Information is routed from HP Service Manager to SCEmail by an eventout record. SCEmail, a special executable that runs on the same platform as the HP Service Manager server, is designed to connect directly to the server, and responds to and processes only email eventout records.
Connect-It	Inbound events. Connect-It establishes a client connection to the HP Service Manager server through a listener and information is routed bi-directionally through it. Connect-It is specifically designed so inbound information must go through Event Services. However, outbound information may be read directly from anywhere in the system.
SCMail SCPager SCMapi SCAuto products	Inbound and outbound events. The products establish a client connection to the HP Service Manager server through a listener and information is routed bi-directionally through it.

## Event Services files

There are five principle tables in HP Service Manager that define events and how they work:

<b>File</b>	<b>Description</b>
eventregister	Defines the events that exist in the system. Event registration records also specify the eventmaps used to process events and defines the RAD application used for processing.
eventin	File used to move information into HP Service Manager from an external system. If a corresponding input eventregister record exists, external or internal applications can write records to the eventin file.
eventout	File used to move information from HP Service Manager into an external system. A specific type of an eventout record can be written only if a corresponding output eventregister record exists.
eventmap	Defines how information is parsed. Eventmaps 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 HP Service Manager Event Services.

Event Services (ES) uses a scheduler called event. You start and stop the event scheduler like any other HP Service Manager scheduler and process events in background or asynchronously.

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

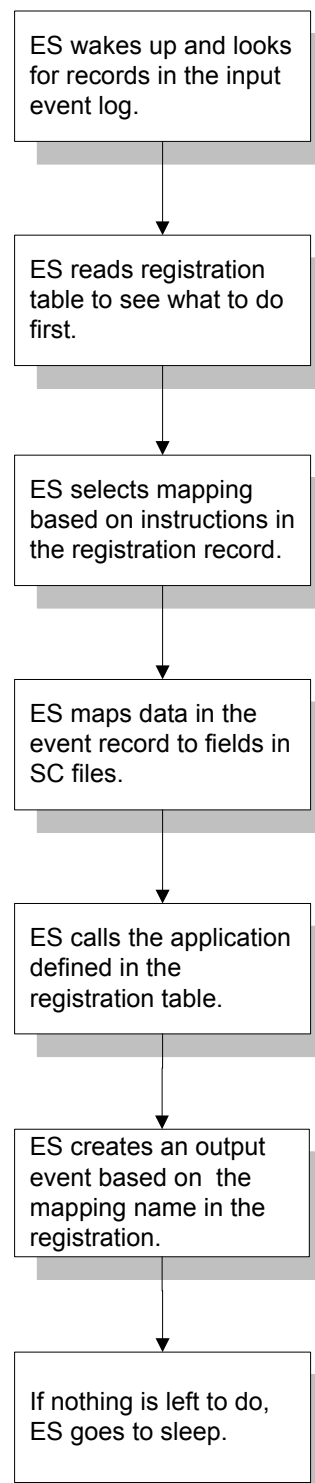
Mapping records contain instructions to move data from the eventin record to fields in HP Service Manager files.

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.

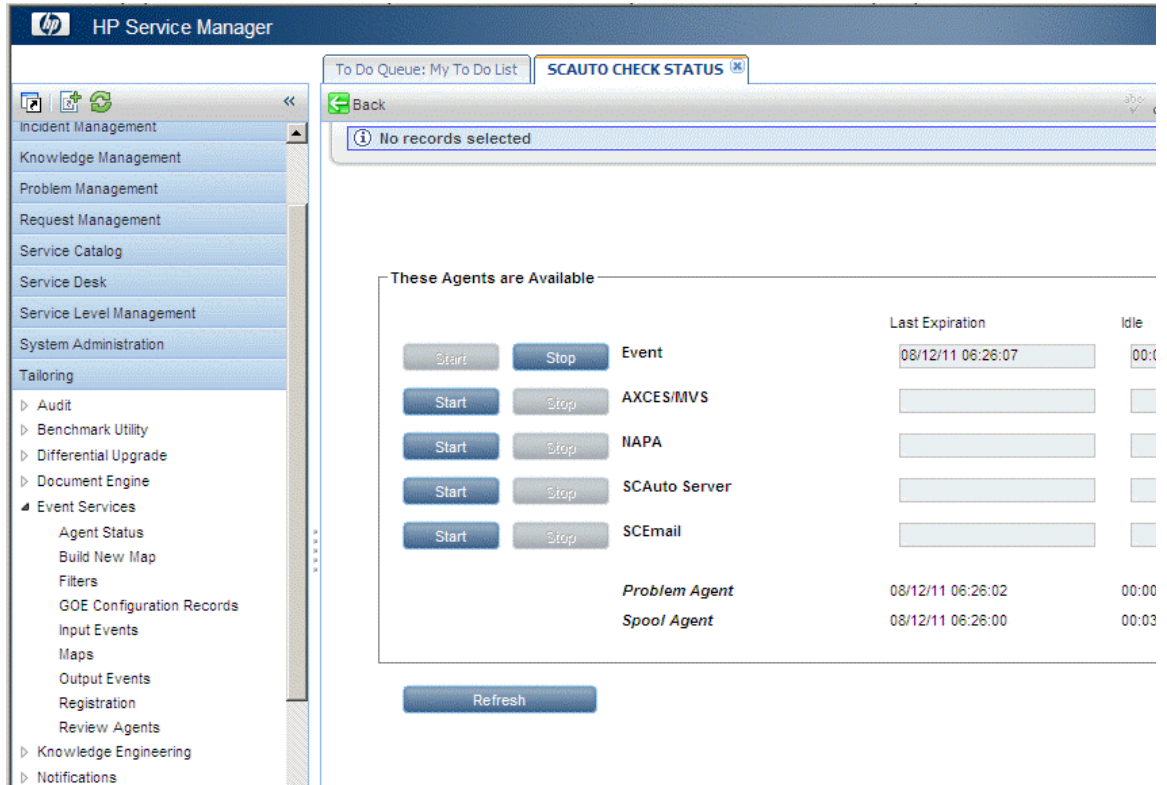


# Accessing Event Services

You must be a HP Service Manager system administrator to work in Event Services.

To access Event Services from the Web or Windows client:

- 1 From the HP Service Manager System Navigation pane, click **Tailoring > Event Services** to display the list of Event Services administration pages:



- 2 Choose the page for the area you want to access:

Page	Purpose
Agent Status	View the status of all agents.
Build New Map	Accesses an application which helps to quickly define a new eventmap for a HP Service Manager file.
Filters	Allows maintenance of event filters. Although general in scope so that you can use filters for any purpose, the primary focus is on incident filtering.
Input Events	Opens the eventin file for review. This file contains all events awaiting action by HP Service Manager and those that have been processed but not deleted.
Maps	Allows maintenance of existing eventmaps. Eventmaps define the relationship between data passed into and out of HP Service Manager in flat, delimited form, and fields in HP Service Manager files.

<b>Page</b>	<b>Purpose</b>
Output Events	Opens the eventout file for review. This file contains all HP Service Manager events awaiting action by an external application and those that have been processed but not deleted.
Registration	Accesses Event Services registration records. Each registration record provides the information HP Service Manager requires to process and event type.
Review Agents	Access the Event Scheduler.

---

## 2 Standard Event Operations

Events take many forms and occur at various times throughout the operation of the system. See [Common Events](#) on page 93 for some of the more commonly-used events.

The primary operations of HP Service Manager Event Services include:

- [Event registration](#) on page 16
- [Input events](#) on page 22
- [Output events](#) on page 28

# Event registration

All events are registered in the `eventregister` file. The `eventregister` file includes a unique event code and a sequence number. A single event can then execute a series of applications. In addition, it contains initialization statements, mapping information and instructions for calling the HP Service Manager application.

## Reviewing event registration

To review event registration:

From the HP Service Manager System Navigation pane, click **Tailoring > Event Services > Registration** to open the Event Registration form.

The screenshot shows the 'EVENT REGISTRATION' form in a web browser. At the top, there are search tabs for 'eventfilter Records' and 'eventregister Records'. Below the search area is a status bar that says 'No records selected'. The main form area is titled 'EVENT REGISTRATION' and contains several input fields: 'Event Code:' with a text box, 'Sequence:' with a text box, 'Input or Output?' with a dropdown menu, 'Translate?' with a dropdown menu, and a checkbox labeled 'Process input events synchronously?'. Below these fields are three tabs: 'Expressions', 'Basics', and 'Application'. The 'Expressions' tab is active and displays a table with approximately 10 empty rows. On the right side of the screenshot, there are two handwritten annotations in black text. The first annotation, with a bracket pointing to the header fields, says 'This area contains the Header fields.' The second annotation, with a bracket pointing to the 'Expressions' tab and its table, says 'This area contains the tabs. The form opens on the Expressions tab.'

*This area contains the Header fields.*

*This area contains the tabs. The form opens on the Expressions tab.*



## Header fields

Field (Internal Name)	Description
Event Code (evtype)	Unique code that identifies this registration.
Sequence (evseq)	Number used to order the sequence of RAD applications to be executed for a single device type.
Input or Output (evftype)	Flag to identify whether this registration is for an input or an output transaction; only input or output is acceptable.
Translate (evtranslate)	Indicates whether to translate to upper (uc) or lower (lc) case. The default value is no translation.
Process input events synchronously? (synch.process)	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 eventin queue.

## Expressions tab

Field (Internal Name)	Description
Expressions (evinit)	Array of statements that execute at run time to initialize variables or initiate action based on the contents of the data passed in the eventin (\$axces) and the eventregister (\$axces.register) records, and/or on global variables available at run time; the global variable \$axces.fields represents an array of the fields passed in the evfield field of the eventin record.

## Basics tab

The screenshot shows a web-based configuration interface for 'EVENT REGISTRATION'. At the top, there are tabs for 'To Do Queue: My To Do List', 'Search eventfilter Records', and 'Search eventregister Records'. Below the tabs is a navigation bar with 'Back', 'Add', 'Search', 'Find', 'Fill', and 'More' options. A status bar indicates 'No records selected'. The main content area is titled 'EVENT REGISTRATION' and contains several input fields and checkboxes. The 'Basics' tab is selected, showing fields for 'Event Map Name', 'Map Type', 'Format Name', 'Use Current Data?', and 'Delete Condition'. There are also fields for 'Event Code', 'Sequence', 'Input or Output?', 'Translate?', and a checkbox for 'Process input events synchronously?'.

Field (Internal Name)	Description
Event Map Name (evmap)	Name of the event map to use.
Map Type (evmaptype)	Determines the length of the map. Use this field for incoming events only. With Variable Length, the Event Map Name is not used, and all the incoming data has no fixed length. With Fixed Length, the Event Map Name is used, and the length is determined by the mapping definitions.
Format Name (evformat)	Used only for output events, the name of the format that displays the record.
Use Current Data? (evnullsub)	If the condition is true, this always substitutes 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 HP in the vendor field, the result of mapping keeps HP as the vendor. The event map allows specification of evnullsub on a field-by-field basis and overrides this default when set in an individual map record.
Delete Condition (evdelete)	A condition whose result determines whether to delete an event in record after it is successfully processed.

## Application tab

To Do Queue: My To Do List    Search eventfilter Records    Search eventregister Records

Back    Add    Search    Find    Fill    More

No records selected

### EVENT REGISTRATION

Event Code:     Input or Output?     Translate?

Sequence:      Process input events synchronously?

Expressions    Basics    Application

Application Name:

Execute Condition:

Description	Parameter Names	Parameter Values
<input type="text"/>	<input type="text"/>	<input type="text"/>

Application to Call on Error Condition:

Field (Internal Name)	Description
Application Name (evapp1)	Name of the RAD application to execute.
Execute Condition (evcondition)	A condition that, if true, allows the RAD application to be executed.
Description (comments)	Array used to describe the elements in the Parameter Names and Parameter Values fields.
Parameter Names (names)	Array of parameter field names that pass to the RAD application; these names must exist in the application file.
Parameter Values (values)	Array of variables or literals that correspond to the list of parameter names passed in the names field; the data types must match.
Application to Call on Error Condition (evgoto)	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 external applications process. In the following example, event code pmo identifies opening an incident.

**EVENT REGISTRATION**

Event Code:  Input or Output?:  Translate?:

Sequence:   Process input events synchronously?

Expressions Basics Application

```

$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=true
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=false
$axces.bypass.failed.validation=true

```

When a pmo event occurs, the system calls application `axces.apm` 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 the map to use.

The expression statements in the previous example set up different queries depending on the source of data. The SCAuto mail incident event uses `logical.name`.

**EVENT REGISTRATION**

Event Code:  Input or Output?:  Translate?:

Sequence:   Process input events synchronously?

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:

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 HP Service Manager SCAuto applications is email, which passes the mail record and the delimiter character. See [Common Events](#) on page 93 for a list of commonly-used events.

## Global variables

The following global variables are available when defining registration events:

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



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

- A fax event is a report that uses the FAX config record name as the printer name. The report writes 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 on conditions at problem open time.

# Input events

The input event log file is called event.in. 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 passes to HP Service Manager in a character string using a delimiter character to separate fields.

To review input events:

- 1 From the HP Service Manager System Navigation pane, click **Tailoring > Event Services > Input Events** to open the eventin (event.in.g) form.

The screenshot shows a web browser window titled "Search eventin Records". The browser address bar shows "To Do Queue: My To Do List" and "Search eventin Records". The browser toolbar includes "Back", "Add", "Search", "Find", "Fill", and "More". The browser status bar shows "No records selected". The main content area is titled "Event Services Input Queue". It contains several input fields: "Event Code", "Status", "System Sequence", "Time Stamps" (with sub-fields "First Expiration:" and "Time Processed:"), "User Information" (with sub-fields "User Name", "Password", "User Sequence"), "Incident Information" (with sub-fields "Network Name", "Cause Code", "Incident ID"), "Filter Information" (with sub-fields "Count", "Next Expiration:"), "System Option", "Field Separation Character", and "External Information String". There are tabs for "Details", "Messages", "Field List", and "Attachments". The "Details" tab is selected. The browser window also shows a "To Do Queue" button and a "Search eventin Records" button.

*This area contains the Header fields.*

*This area contains the tabs. The form opens on the Details tab.*

- 2 Click **Search** to display a record list of all input events.
- 3 Double-click on an event to display the record.

## Header fields

Field (Internal Name)	Description																						
Event Code (evtype)	The registration name for the event (required).																						
Status (evstatus)	<p>The result of the Event Manager action. If events are not deleted after processing, HP Service Manager 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 name is in the Network Name field.</td> </tr> <tr> <td>closed</td> <td>An incident has been closed; the Incident Number is in the Incident ID field.</td> </tr> <tr> <td>deleted</td> <td>An inventory item has been marked for deletion in the database; the device name is in the Network Name 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 Incident ID 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 Incident ID field.</td> </tr> </tbody> </table>	Status	Description	added	An inventory item has been added to the database; the device name is in the Network Name field.	closed	An incident has been closed; the Incident Number is in the Incident ID field.	deleted	An inventory item has been marked for deletion in the database; the device name is in the Network Name 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 Incident ID field.	processed	A software inventory item or change has been successfully processed.	updated	An incident has been updated; the Incident Number is in the Incident ID field.
Status	Description																						
added	An inventory item has been added to the database; the device name is in the Network Name field.																						
closed	An incident has been closed; the Incident Number is in the Incident ID field.																						
deleted	An inventory item has been marked for deletion in the database; the device name is in the Network Name 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 Incident ID field.																						
processed	A software inventory item or change has been successfully processed.																						
updated	An incident has been updated; the Incident Number is in the Incident ID field.																						
System Sequence (evsysseq)	System-assigned sequence number, for event tracking (system provided).																						
First Expiration (evtime)	The time the event occurred (required).																						
Time Processed (evtimestamp)	The system time translation of the actual time that HP Service Manager processed the event.																						

## Details tab

Field (Internal Name)	Description
User Name (evuser)	The event user name; if passed, it is the operator name (optional).
Password (evpswd)	The password for the event user (optional).
User Sequence (evusrseq)	User-assigned sequence number, used to trace an event through the HP Service Manager system (for example, an external reference number; optional).
Network Name (evnetnm)	Used in filtering, the unique network name of a device (system defined by Event Services).
Cause Code (evcode)	Used in filtering, an event code sent to Event Manager (system defined by Event Services).
Incident ID (evid)	Problem character ID; used in filtering (system defined by Event Services).
Count (evcount)	Used in filtering, the number of events for a specific transaction (system defined by Event Services).
Next Expiration (evexpire)	Used in filtering, the time when an incident is opened (system assigned by Event Services).
System Option (evsysopt)	Code to identify system options (optional).
Field Separation Character (evsepchar)	Character used to separate fields in the evfields. The field substitutes the caret (^) character if null.
External Information String (evfields)	<p>Data describing the event, with fields separated by the evsepchar character; specific positions in the evfields field are reserved for application dependent data.</p> <p>For example:</p> <pre>falcon^max@hp.com^falcon;susie;root^ Re: meeting this afternoon^Tuesday, 23 January 2004 16:41:07</pre> <p>In this example, max@hp sends falcon an email, with carbon copies to falcon, susie, and root. The subject is Meeting this afternoon, and the text follows the subject.</p> <p>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 the default.</p>



## Messages tab

To Do Queue: My To Do List Search eventin Records

Back Add Search Find Fill More

No records selected

### Event Services Input Queue

**Event Code**

Event Code:

Status:

System Sequence:

**Time Stamps**

First Expiration:

Time Processed:

Details Messages **Field List** Attachments

Field	Description
Messages (evmsg)	Any messages generated during event processing.

## Field List tab

To Do Queue: My To Do List Search eventin Records

Back Add Search Find Fill More

No records selected

### Event Services Input Queue

**Event Code**

Event Code:

Status:

System Sequence:

**Time Stamps**

First Expiration:

Time Processed:

Details Messages Field List **Attachments**

To display this list, you must enable the debug option in the Registration for this Event Type. To do so, enter

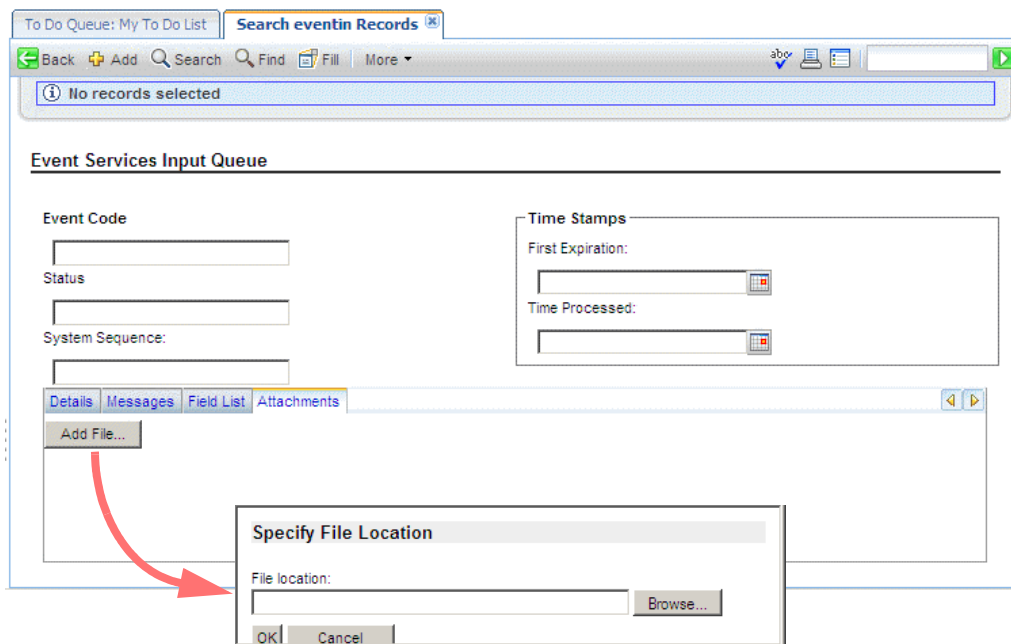
Saxces.debug.on=true

in the Initializations section.

Field (Internal Name)	Description
Field List (evlist)	Array, built by the Event Manager, of the fields in the evfield field; available to eventmap as \$axces.fields.

- Note: The evlist field refreshes in the application after use. If you need to view it for debugging or trace purposes, you must first set \$axces.debug=true in your event registration initialization expressions. The maximum size of the evfields data is 16,000 bytes. Use this feature with discretion because email messages are often quite large.

## Attachments tab



Use the Attachments tab to include various objects related to the Input record. To insert files, click the **Add File** button.

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

## Input event processing

An external application, such as SCAuto/SDK or SCAuto for NetView OS/390, adds all records in the eventin file. External programs manipulate the eventin records.

For example, SCAuto supports an event called email. Electronic mail can be received from external sources and passed to HP Service Manager 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.

- Records in the eventin file that have been processed do not contain a First Expiration value in the 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 terminates for that event and the specific error message writes to the `eventin` Messages and to HP Service Manager `msglog` file.

Once you install and test SCAuto, do one of the following:

- Set all delete flags in the registration records to true.
- Use the HP Service Manager purge/archive routines to schedule cleaning up the file on a regular basis.

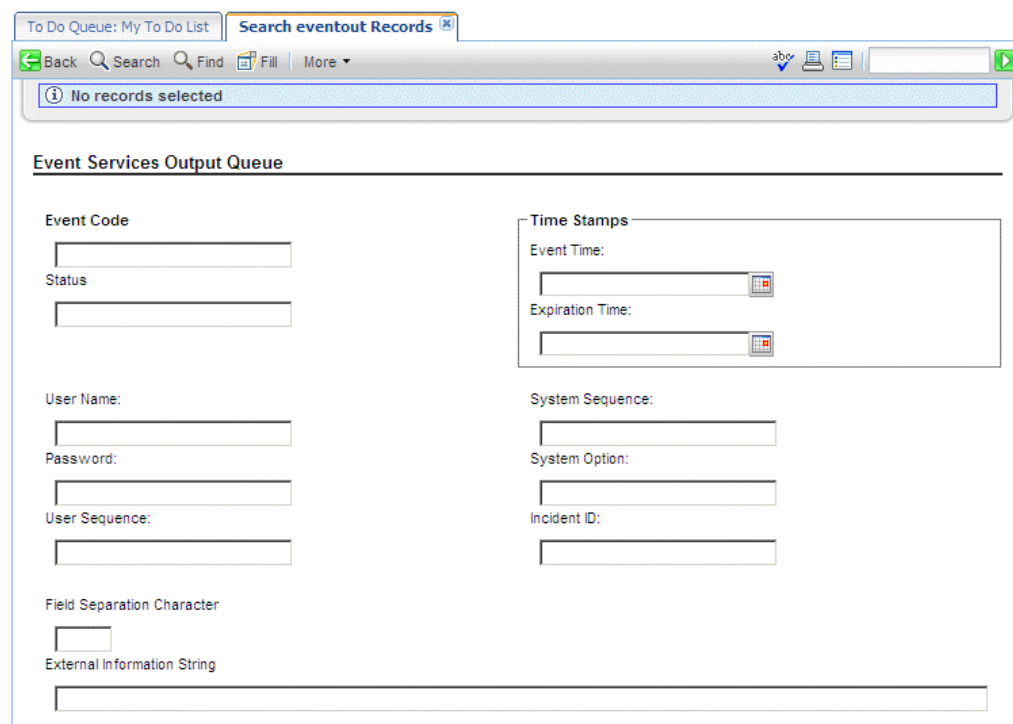
See Database Administration in the HP Service Manager online help for additional information about the Purge/Archive Utility.

## Output events

The output event log is called `eventout`. It contains a record for each event processed by Event Services applications and instructions that external software (for example, pager numbers to notify service technicians) uses. Data passes to external applications in a character string using a delimiter character to separate fields.

To review output events:

- 1 From the HP Service Manager System Navigation pane, click **Tailoring > Event Services > Output Events** to open the `eventout` (`event.out.g`) form.



The screenshot shows a web browser window titled "Search eventout Records". The browser's address bar shows "No records selected". The main content area is titled "Event Services Output Queue" and contains several input fields for searching records:

- Event Code:** A text input field.
- Status:** A text input field.
- Time Stamps:** A container with two date pickers: "Event Time:" and "Expiration Time:".
- User Name:** A text input field.
- Password:** A text input field.
- User Sequence:** A text input field.
- System Sequence:** A text input field.
- System Option:** A text input field.
- Incident ID:** A text input field.
- Field Separation Character:** A small text input field.
- External Information String:** A large text area at the bottom.

- 2 Click **Search** to display a record list of current output events.
- 3 Double-click an event to display the record.

## Output fields

The encoded field names recorded in the eventout file are for reference only.

<b>Field (Internal Name)</b>	<b>Description</b>
Event Code (evtype)	Registration name for the event (required).
Status (evstatus)	Result of the action Event Manager; the actions are: opened, updated, closed, added, deleted, filtered, or error.
Event Time (evtime)	Time the event occurred.
Expiration Time (evexpire)	Expiration time for an event. The time when the event scheduler processes the eventout record; if the field is NULL, no processing occurs.
User Name (evuser)	Event user name (optional).
Password (evpswd)	Password for the event user (optional).
User Sequence (evusrseq)	User-defined sequence number for event tracking.
System Sequence (evsysseq)	System-assigned sequence number, for event tracking (system provided); used when external software restarts the eventout monitoring pointer.
System Option (evsysopt)	Code to identify system options (optional).
Incident ID (evid)	Problem character ID (incident number).
Field Separation Character (evsepchar)	Same as for Input fields. See <a href="#">Input event processing</a> on page 26.
External Information String (evfields)	Same as for Input fields. See <a href="#">Input event processing</a> on page 26.

Records in the eventout file that are processed do not contain an expiration date. Normally, events are deleted from the eventout file after processing unless an error occurs. A flag in the 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.



Use the HP Service Manager purge/archive routines to schedule cleaning up the eventout file on a regular basis. See Database Administration in the HP Service Manager online help for additional information about the Purge/Archive Utility.

External programs manipulate the eventout records.



---

## 3 Mapping and Filtering

Once HP Service Manager events are created, incoming or outgoing, processes must be implemented to manage and direct the events. Event mapping and event filtering take the event and constituent data, and then direct it in specified ways to create results within other areas of the system.

This chapter describes these processes in the following sections:

- [Mapping](#) on page 32
- [Event filters](#) on page 47

# Mapping

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



Event Maps provided with Event Services describe standard events. Changing the relative position of data in the information exchanged between HP Service Manager and the external applications may cause standard events to fail. Create new maps for non-standard events rather than modifying existing maps.

## The event map form

To review event maps:

From the HP Service Manager System Navigator pane, click **Tailoring > Event Services > Maps**.

The screenshot shows the 'Event Map' configuration form. At the top, there's a navigation bar with 'To Do Queue: My To Do List' and 'Search eventmap Records'. Below that is a toolbar with 'Back', 'Add', 'Search', 'Find', 'Fill', and 'More'. A status bar indicates 'No records selected'. The main form area is titled 'Event Map' and contains several input fields and dropdown menus. The 'Map Name' field is followed by 'Sequence', 'Position', and 'Length'. To the right are 'Type' and 'Fixed or Variable' dropdowns. Below this is a tabbed interface with 'Basics' and 'Expressions' tabs. The 'Basics' tab is active and contains fields for 'File Name', 'Query', 'Field Name', 'Data Type', 'Nullsub', 'Translate', 'Array Information', 'Element Type', 'Element Separator', and 'Element Length'. The 'Expressions' tab is currently inactive.

*This area contains the Header fields.*

*The area contains the tabs; the form opens on the Basics tab.*



## Header fields

Encoded field input names recorded in the eventmap file are included in parenthesis for reference only.

<b>Field (Internal name)</b>	<b>Description</b>
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.
Fixed or Variable (evmaptyp)	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, dependencies may exist that necessitate a prescribed order for field mapping; used in icm* maps.
Position (evindex)	Number corresponding to the relative position of data in the evfields field for the eventin record.
Length (evlength)	If evmaptyp is Fixed Length, you must provide the length of the field.

## Basics tab

<b>Field (Internal name)</b>	<b>Description</b>
File Name (evfile)	Name of the file from (for output) or into (for input) which data is 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 (that is, 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 is 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 \$axces.field 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; the Build Event Maps process sets this value and is automatically set when the event map record is being added or updated. If evdtype is Array, you must complete the required fields in the Array Information section of the form.

<b>Field (Internal name)</b>	<b>Description</b>
Translate (evxlate)	Indicates whether to translate the field value to uppercase (uc) or lowercase (lc); the default is to not translate.
Element Type (eveltype)	Data type of array elements. If eveltype is Structure, you must enter a different separator for the evsepchar and evsepchar.struc fields. If eveltype contains a value other 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; the default is the   (pipe symbol).
Element Length (evitmlng)	If not NULL, defines the length of each element in array type fields. This field does not apply if eveltype 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 the ` (grave accent).



## 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 HP Service Manager files, and data in HP Service Manager files to a sequence of delimited fields for export to external applications.

For example, when a HP Service Manager user sends mail, specific fields in the HP Service Manager mail file are populated. These include user.to, user.from, user.array, subject and text. When sending email, you must map the information in these fields 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 required fields in the HP Service Manager mail file.

The screenshot displays the 'eventmap' configuration window. At the top, a table lists several records. The second record is highlighted in yellow and circled in red. Below the table, the 'Event Map' configuration form is shown, with the 'Position' field and the 'Field Name' field also circled in red.

Map Name	Seq	Pos	File Name	Field Name	Query
email	1	1	mail	user.to	
email	1	2	mail	user.from	
email	1	3	mail	user.array	
email	1	4	mail	subject	

**Event Map Configuration Form:**

- Map Name: email
- Sequence: 1
- Position: 2
- Type: Input
- Fixed or Variable: [ ]
- Length: [ ]
- File Name: mail
- Query: [ ]
- Field Name: user.from
- Nullsub: [ ]
- Data Type: Character
- Translate: [ ]
- Array Information:
  - Element Type: [ ]
  - Element Separator: [ ]
  - Element Length: [ ]
  - Element Separator (structure): [ ]

As shown in this record, the user.from field in the HP Service Manager mail file has a position of 2, and is the second field in the delimited text string written to the Field List for the eventin record.

For output, the contents of the user.from field in the HP Service Manager mail file is placed in the second position in the External Information String field of the eventout record. The Type field is changed to output.

Map Name	Seq	Pos	File Name	Field Name	Query
email	1	1	mail	user.to	
email	1	2	mail	user.from	
email	1	3	mail	user.array	
email	1	4	mail	subject	

Count Records: 1 to 10 of 10 | Pages: 1 | Show: 100 records per page

Event Map Configuration:

Map Name: email | Type: **Output** | Fixed or Variable:

Sequence: 1 | Position: 2 | Length:

Basics | Expressions

File Name: mail

Query:

Field Name: user.from | Nullsub:

Data Type: Character | Translate:

Array Information

Element Type:  | Element Separator:  | Element Separator (structure):

Element Length:

If the mapping records for email are deleted, HP Service Manager uses the default as shown in this Output example that the system provides when you install Event Services.

Event Services also handles mapping to multiple files. For example, SCAuto for NetView OS/390 and SCAuto can 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 using discovery processes in external applications such as OpenView and passed to HP Service Manager via SCAuto, both files are updated.

**Event Map**

Map Name:  Type:  Fixed or Variable:

Sequence:  Position:  Length:

**Basics Expressions**

File Name:

Query:

Field Name:  Nullsub:

Data Type:  Translate:

**Basics Expressions**

Initialization

Condition for Mapping:

Post-Map Instructions

```

if (index(type in Saxces.target, $G.devtypes.all)=0) then (comments in Saxces.target=comments in Saxces.target+{"Originally of type: "+type in $
format.name in Saxces.target="device."+type in Saxces.target
if (format.name in Saxces.target="device.device") then (format.name in Saxces.target="device")
$attribute.file=type in Saxces.target

```

The first step in preparing to map multiple files is to identify the attribute file. This is achieved by using an expression (see line 4) in the Post Map Instructions to set the variable `$attribute.file` to the value in the type field of the device (TARGET) record.

➤ 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. The query passed in the Registration file already selected the record, therefore no query is necessary.

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

➤ While `$axces.target` and `$axces.field` have special meaning within Event Services, `$attribute.file` is an arbitrary global variable name.

**Event Map**

Map Name:  Type:  Fixed or Variable Length:

Sequence:  Position:  Length:

**Basics** | Expressions

File Name:

Query:

Field Name:  Nullsub:

Data Type:  Translate:

Array Information

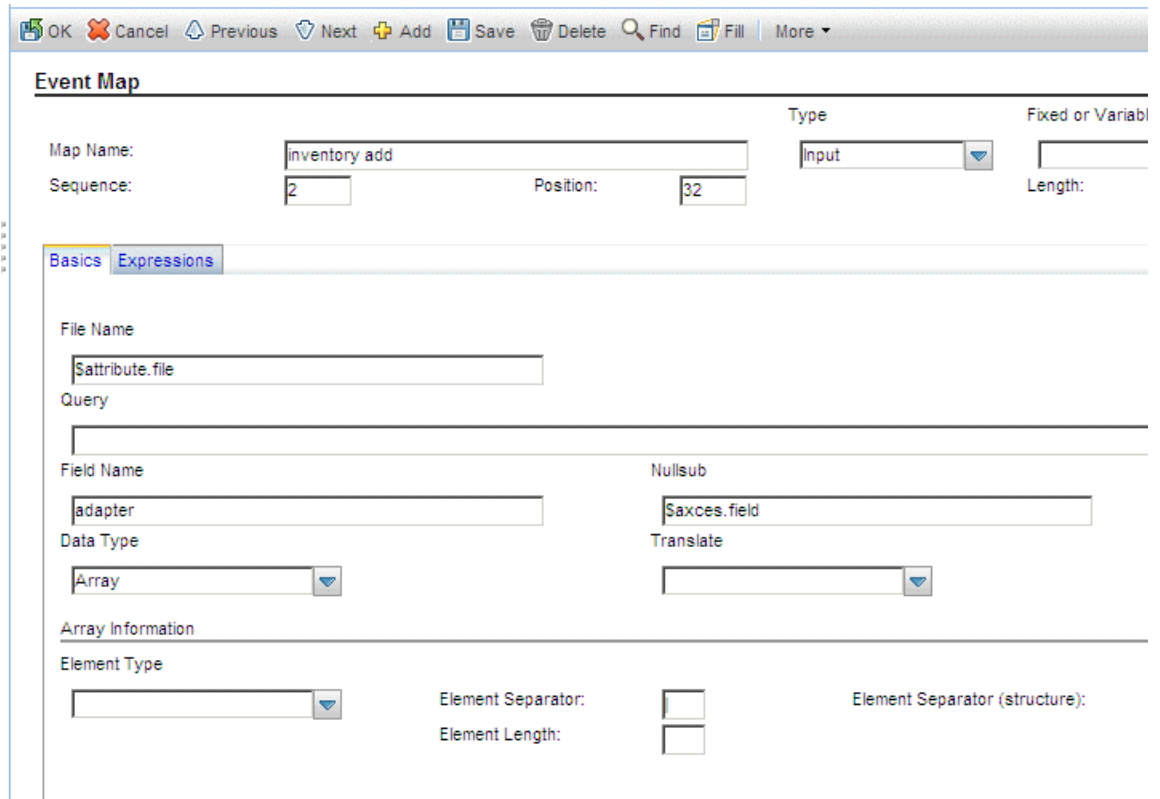
Element Type:  Element Separator:  Element Separator (structure):

Element Length:

When all fields are mapped into the device file, the next map record has a Sequence of 2, 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 shown in the previous example) 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 (as shown in the previous example) of the `evfields` array field, which is represented by the `$aces.fields` variable in the `eventin` record.



The screenshot shows the 'Event Map' configuration window. The 'Map Name' is 'inventory add', 'Sequence' is '2', and 'Position' is '32'. The 'Type' is set to 'input'. The 'File Name' is '\$attribute.file'. The 'Field Name' is 'adapter' and the 'Data Type' is 'Array'. The 'Nullsub' is '\$aces.field'. The 'Array Information' section includes 'Element Type', 'Element Separator', 'Element Length', and 'Element Separator (structure)', all of which are currently empty.

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



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 online validation with user feedback, but you can check field values and make substitutions based on processing statements.

In the preceding record, the value in `network.name` replaces `logical.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. Event Services applications handles data type and case conversions as long as the Field Type field is correctly identified and the data is written to the descriptor structure.

▶ 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 HP Service Manager.

!! 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. For example, if you add 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`. The field must exist in the target file before any instruction can manipulate it. Ensure the data type is correctly identified.

▶ 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 event in record.
\$axces.fields	Represents the evlist field in the event in 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 HP Service Manager database to which event information is posted.
\$axces.notriml	If set to true, any blank spaces or tabs at the end of the field are not removed.
\$axces.notrimr	If set to true, any blank spaces or tabs at the beginning of the field are not removed.



When email events are sent to HP Service Manager, leading spaces, trailing spaces, and tabs are not removed from the text field.

## Mapping considerations for Inventory Management

While HP Service Manager 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. You can often fully describe a device using only the fields in the `device` file.

The map record for the `type` field (field #9 in standard events) defines how HP Service Manager selects and displays information about a device once the data is added. The `type` field in the `device` file refers directly to the associated attribute file of each device. If no attribute file associated with a device, the `type` field must contain `device` or be empty (NULL).

Similarly, the `format.name` field in the `device` record defines the name of the form that displays the device within HP Service Manager and, by extension, the name of the join file that temporarily stores information for review and update. The `formatctrl` record for the format name stored in the `device` record must contain `device` as the file name for all device types that do not have associated attribute files.

If an external agent detects an unknown device type, HP Service Manager 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 Message list for the event but the device is still added or updated in the HP Service Manager 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

You can build both input and output event maps for any file in HP Service Manager.

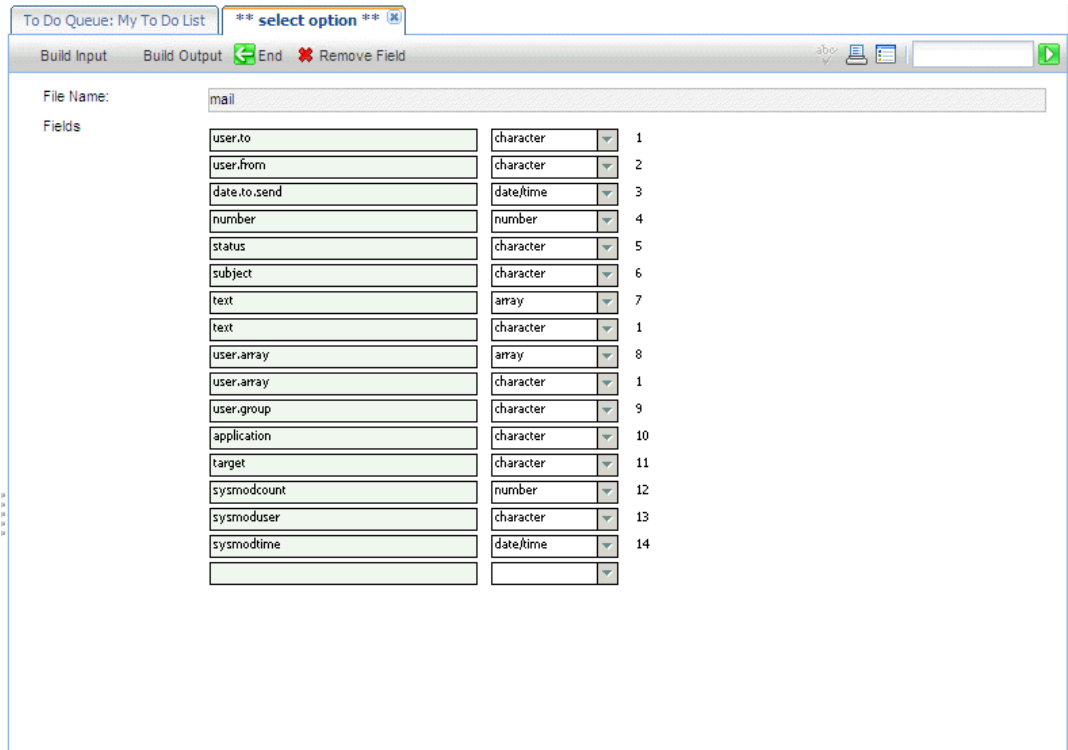
To build a new map:

- 1 From the HP Service Manager System Navigator pane, click **Tailoring > Event Services > Build New Map** to open the Build Event Mapping form.
- 2 Type the Map Name and a Source file name, for example:

The screenshot shows a web-based form titled "Build Event Mapping". The form is part of a larger application window with a title bar that includes "To Do Queue: My To Do List" and a tab labeled "\*\* enter map and file names \*\*". Below the title bar is a status bar indicating "No records selected". The main content area of the form has the heading "Build Event Mapping" and a prompt: "Please enter the event map name and source file." There are two input fields: "Enter Event Map Name:" with the text "test" entered, and "Enter Source File Name:" with the text "mail" entered. At the bottom of the form, there are three buttons: "< Previous", "Next >", and "Cancel".

Each mapping must have a unique name.

- 3 Press **Enter** to open a list of field names and data types for the file you selected.



If you do not provide a source file name, HP Service Manager displays a record list of files where you can make a selection.



HP Service Manager issues a warning if the event map name 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 is not removed.

The top of the screen contains buttons that allow you to manipulate and build a map record.

Button	Property
Build Input	Builds the records that map information from the eventin file to the selected HP Service Manager file.
Build Output	Builds the records to map information from the selected HP Service Manager file to a formatted string to be passed to SCAuto using 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.



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. An external event, such as SCAutomate, or SCAuto for NetView OS/390, passes data into the HP Service Manager eventin file in a field called fields. Each element is separated from the others with a delimiter, or separation character. In the following example, the ^ character separates the five fields.

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

Internally, Event Services converts this string to a list (\$aces.fields):

```
john@hp
falcon
toby;al;joe
Meeting today
Tue 12 Augo1
```

The event processor assumes that fields with a type of date/time are in the time zone of the HP Service Manager system (that is, the time zone defined in the System Wide Company Record). If the event background process has an operator record, that time zone for that operator is used. For synchronous processing, the session processing the event handles the date/time in the time zone where it is defined.

Mapping defines the link between the elements in the internal list (evlist) and fields in a HP Service Manager file. The first field, john@hp, is mapped to the user.to field for the mail file.

The screenshot shows the 'Event Map' configuration window in HP Service Manager. The window title is 'Search eventmap Records'. The main area is titled 'Event Map' and contains several fields and controls:

- Map Name:** email
- Sequence:** (empty)
- Position:** (empty)
- Type:** Input
- Fixed or Variable:** (empty)
- Length:** (empty)

Below the main fields, there are two tabs: 'Basics' and 'Expressions'. The 'Basics' tab is selected and contains the following fields:

- File Name:** mail
- Query:** (empty)
- Field Name:** user.to
- Data Type:** Character
- Nullsub:** (empty)
- Translate:** (empty)

At the bottom, there is an 'Array Information' section with the following fields:

- Element Type:** (empty)
- Element Separator:** (checkbox, unchecked)
- Element Length:** (checkbox, unchecked)
- Element Separator (structure):** (checkbox, unchecked)

For best results when building new maps that use 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.

# Event filters

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

To review event filters:

From the HP Service Manager System Navigation pane, click **Tailoring > Event Services > Filters**.

The screenshot shows a web application window titled 'eventfilter'. At the top, there is a table with columns 'Event Type', 'Network Name', and 'Block?'. The first row is highlighted in yellow and contains 'pmo' under 'Event Type' and 'AXCES' under 'Network Name'. The second row contains 'icma'. Below the table is a pagination bar showing 'Count Records 1 to 2 of 2', 'Pages: 1', and 'Show 50 records per page'. A toolbar with icons for 'OK', 'Cancel', 'Previous', 'Next', 'Add', 'Save', 'Delete', 'Find', 'Fill', and 'More' is located below the pagination bar. The main content area is titled 'EVENT FILTERS' and contains several input fields: 'Event Type' (with 'pmo' entered), 'User Name' (empty), and a tabbed interface with 'Internal Filters' selected. Under 'Internal Filters', there are two filter rules. The first rule has 'Index: 3', 'Value: example', and 'Condition: and'. The second rule has 'Index: 2', 'Value: example'. At the bottom, there is a checkbox for 'Block Events?' (unchecked) and two date pickers for 'Start Blocking at:' and 'End Blocking at:'.

## Header fields

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

## External Filters tab

Field (Internal name)	Description
Index (evindex1)	Position in the eventin record evfields field that identifies the first mask field.
Value (evvalue1)	Value that causes the event to be masked if it appears in the position indicated by evindex1 in the evfields field for the eventin record.
Condition (evcondition)	Value of and or or that concatenates the clauses built with the evindex and evvalue fields.
Index (evindex2)	Position in the evfields field for the eventin record that identifies the second mask field.
Value (evvalue2)	Value that causes the event to be masked if it appears in the position indicated by evindex2 in the evfields field for the eventin record.
Block Events? (evblock)	Logical field that indicates whether events are blocked entirely; this field is required when blocking events from being written to the eventin table by the external scheduler. See <a href="#">Blocking</a> on page 51 for more information.
Start Blocking at (evstime)	Beginning time for masking events.
End Blocking at (evetime)	Ending time for masking events.



## Internal Filters tab

Field (Internal name)	Description
Initial Statements (evinit)	Array of statements that execute at run time to initialize variables or initiate action based on the contents of the data passed in the eventin record and/or on global variables available at run time; the global variable \$axces.fields represents 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, block the event and cause the registered application to exit normally; the status in the eventin record is then filtered.

## Additional Incident Filters tab

The screenshot shows a web browser window titled "Search eventfilter Records". The address bar shows "To Do Queue: My To Do List" and "Search eventfilter Records (x)". The browser's navigation bar includes "Back", "Add", "Search", "Find", "Fill", and "More". A status bar at the top indicates "No records selected". Below this, the heading "EVENT FILTERS" is displayed. There are two input fields: "Event Type:" and "User Name:". A tabbed interface below has three tabs: "External Filters", "Internal Filters", and "Additional Incident Filters" (which is selected). Under the "Additional Incident Filters" tab, there are seven input fields arranged in two columns: "Network Name", "Cause Code", "Event Interval", "Recurrence Count", and "Recurrence Interval".

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

## Blocking

The external SCAuto application use the External Filters tab of the filter record to prevent the insertion of event in records in the HP Service Manager 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 event in 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 customized administration.

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

The screenshot shows the 'EVENT FILTERS' configuration interface. At the top, there are tabs for 'External Filters', 'Internal Filters', and 'Additional Incident Filters'. The 'External Filters' tab is active. Below the tabs, there are fields for 'Event Type' (set to 'pmo') and 'User Name' (empty). The filter configuration section includes two filter rules. The first rule has an 'Index' of 3 and a 'Value' of 'example'. The second rule has an 'Index' of 2 and a 'Value' of 'example'. The 'Condition' between the two rules is set to 'and'. At the bottom, the 'Block Events?' checkbox is checked, and the 'Start Blocking at' and 'End Blocking at' times are set to 08:00 and 17:00 respectively.

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. Use these fields 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 contains the following message:

```
hp^hp^^6 58916865^Node Down^^^^SNMP Trap^net.hware^^^^^^^^^^^
```

The caret (^) character separates fields in the message. The first field, which references the logical name of the device (see [Mapping](#) on page 32), contains hp. To block the insertion of all incident open events reported for the device hp, type pmo in the Event Type field, 2 in the first Index field and hp in the first Value field.



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

To block incident open events from both hp and another server named dolphin, type information as previously described and type or in the Condition field, 2 in the second Index field and dolphin in the second Value field. If you specify a condition (and or or), then you must complete both Index and both Value fields.



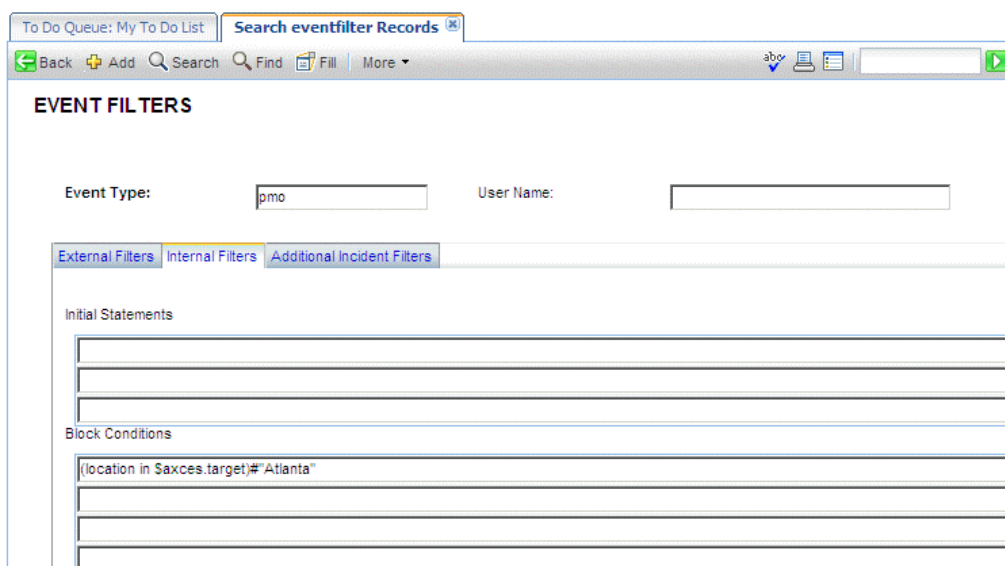
To prevent insertion of records in the eventin file, the Block field must be true.

In the following tab, all inventory add (icma) events are blocked between 08:00 and 17:00 if they come from either the hp1 or dolphin server. This action avoids unnecessary adds and updates if installation activity is scheduled to occur on the network during this time.

The screenshot shows a web browser window titled "Search eventfilter Records". The browser's address bar shows "To Do Queue: My To Do List" and "Search eventfilter Records". The browser's navigation bar includes "Back", "Add", "Search", "Find", "Fill", and "More". A status bar at the top indicates "No records selected". The main content area is titled "EVENT FILTERS". It contains a form with the following fields: "Event Type:" with a text box containing "icma", and "User Name:" with an empty text box. Below this are three tabs: "External Filters", "Internal Filters", and "Additional Incident Filters". The "External Filters" tab is selected. It contains two filter rules. The first rule has "Index:" with a text box containing "3", "Value:" with a text box containing "hp1", and "Condition:" with a dropdown menu showing "or". The second rule has "Index:" with a text box containing "2", "Value:" with a text box containing "dolphin". At the bottom of the form, there is a checkbox labeled "Block Events?" which is checked. To the right of the checkbox are two time pickers: "Start Blocking at:" with a value of "08:00" and "End Blocking at:" with a value of "17:00".

The number of filters available for external blocking is unlimited; the external process (SCAuto) 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.

Once records are 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 must contain all internal blocking conditions. If an eventin record satisfies one of the Block Conditions, it is updated to reflect a Status of blocked. The event action (for example, incident open or inventory add) does not take place.



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 event in record and the Cause Code is the same as the cause code specified in the event in 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 event in 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 event in 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:

```
hp^hp^6 58916865^Node Down^^^SNMP Trap^net.hware^^^^^^^^^^
```

The queries are:

```
hp^hp^6 58916865^Node Down^^^SNMP Trap^net.hware^^^^^^^^^^
```

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 event in 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 hp with cause code of SNMP 2.0 from opening an incident unless three events are received within a ten minute interval.

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

The filter in the following record prevents any events from server hp from opening an incident unless 3 events are received and remain active for more than ten minutes.

External Filters		Internal Filters		Additional Incident Filters	
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"/>		

---

## 4 Change Management Event Services

The Change Management application of HP Service Manager is fully supported by Event Services. This allows users outside of the HP Service Manager system to perform all standard functionality of Change Management from an external system, for example, SAP or PeopleSoft. The Event Services implementation is bidirectional, allowing external systems to synchronize with the HP Service Manager system.

This chapter provides the HP Service Manager 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 56
- [Synchronizing HP Service Manager with an external system](#) on page 59
- [Change event examples](#) on page 61

## Input events

A correctly formatted eventin record must be created within HP Service Manager to use an external system to produce an action within the HP Service Manager Change Management application. You can format the eventin record with an SCAutomate product.

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

Field (Internal name)	Description
Event Code (evtype)	Name of the corresponding Event Registration record to use for this event. This must always be cm3rin for changes and cm3tin for tasks.
User Name (evuser)	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 (evfields)	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
cm3rin	Input	cm3r	axces.cm3	Used for Changes
cm3tin	Input	cm3t	axces.cm3	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 event record. This field carries the specific data of the change or task into the HP Service Manager 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 HP Service Manager.
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.).
1	Change/Task number of the object to be acted upon. This field is blank when opening a change/task.

### Determining the correct change/task

The first two EIS fields determine the unique identifier of the change or task both in HP Service Manager 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 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 HP Service Manager number is not specified, the system attempts to find the correct record by comparing the `foreign.id` to this field.

### Supported actions

Event Services uses the third field of the EIS 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

<b>Action</b>	<b>Description</b>
update	Update an existing change/task
close	Close current phase and advance to the next phase, if applicable.
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).

# Synchronizing HP Service Manager with an external system

When HP Service Manager 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 HP Service Manager 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
<code>cm3rinac</code>	Output	<code>cm3ack</code>	<code>axces.write</code>	Used for Changes
<code>cm3tinac</code>	Output	<code>cm3ack</code>	<code>axces.write</code>	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 HP Service Manager.
3	Action Token indicating which action was performed on this object (open, update, and so on).
4	The status of the <code>eventin</code> 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, 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 has the following specific fields:

Field	Value
evtype	cm3rin
evuser	falcon

The EIS is:

```
^CM01^open^^^MAC^^^falcon^^^bob.helpdesk^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^Mo
ve 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 example

### Using cm3messages to output changes when updated

Entering cm3rout in the cm3rupdate record triggers an output event whenever a change is updated.



# 5 SCemail

SCemail provides a monitor to handle HP Service Manager email events. This monitor connects HP Service Manager into standard email facilities and allows HP Service Manager operators and applications to send HTML emails. Any mail system that supports Simple Mail Transfer Protocol (SMTP) or has an SMTP gateway or bridge can receive email from SCemail.



Note: Only messages enclosed within HTML tags can be sent out as HTML emails, otherwise messages are sent as plain text.

This chapter contains the following sections:

- [Email events](#) on page 64
- [SCemail vs. SCAutoMail](#) on page 65
- [Event Services and HP Service Manager email](#) on page 66
- [Emailout Parameters in the sm.ini File](#) on page 67
- [SCEmail Background Processor](#) on page 68
- [Event Services and Format Control](#) on page 69
- [Sending HP Service Manager mail to email](#) on page 70

## Email events

A standard email event that HP Service Manager creates is the opening of an incident with a valid Contacts field. This event can notify individuals of an incident in their area of expertise. You also can create email events using the User Services Send Mail function.

In addition to the standard creation of email events in HP Service Manager, any RAD application can create an event. An example of this is implementing email notification for problems that reach a specified status.



## SCemail vs. SCAutoMail

SCemail is not the same product as SCAutomate Mail. SCemail only sends mail from HP Service Manager; it does not receive mail from external mail applications. SCemail runs as a stand-alone application; SCAutomate Mail is an SCAutomate client adapter.

## Event Services and HP Service Manager email

The HP Service Manager 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 HP Service Manager contacts file.

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

This causes HP Service Manager 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.

## Emailout Parameters in the sm.ini File

SCEmail requires the setup of Emailout parameters in the sm.ini file before it takes effect. The system administrator must set these parameters from the HP Service Manager server's OS command prompt or from the initialization file (sm.ini). The following list shows desired Emailout parameters for SCEmail:

- `smtpHost`: specifies the name of the SMTP server host for client requests.
- `smtpPort`: defines the communications port SMTP uses.
- `smtpUsername`: defines the account name of the SMTP server.
- `smtpPassword`: identifies the password the HP Service Manager server uses to bind to the SMTP server.
- `smtpTLS`: defines whether SMTP requires Transport Layer Security (TLS) authentication to send emails.
- `smtpEnableSSL`: defines whether SSL should be used for SMTP operations.
- `smtpSSLPort`: defines the port number for SSL connection.
- `mailFrom`: specifies the descriptive name or other identifier of the sender of an email. This parameter should be set in the format of email address.

## SCEmail Background Processor

You can add the following optional parameters when starting the SCEmail background processor.

<b>Parameter</b>	<b>Description</b>
-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.
-debug	Print more diagnostics to sc.log. This also turns on -keepmail.
-clean	Don't put extraneous headers in mail. Example: HP Service Manager Operator: falcon



Note: SCEmail also processes the sc.ini file for additional parameters and can pass the parameters on the command line (for example, -log:file places the SCEmail diagnostics in a different file).

## Event Services and Format Control

SCAutomate supports a generic email function. Use the Format Control RAD function, `message.fc`, to write email events to the eventout file.

### Writing an Incident Management eventout record

When Event Services opens, updates, or closes problems, 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 using the SCAuto external interface. You can elect to write to the eventout file when Service Desk operators open and close incidents 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 are selected from the record. The application is called as a Format Control subroutine passing two parameters; the first is the record from which data is mapped, and the second is the Event Type, as defined in the Event Register.

### Writing a Configuration Management eventout record

When Event Services adds, updates, or deletes configuration items, 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 is passed to an external process using the SCAuto external interface. You can elect to write to the eventout file when operators maintain configuration items so that the information passes to the external interface.

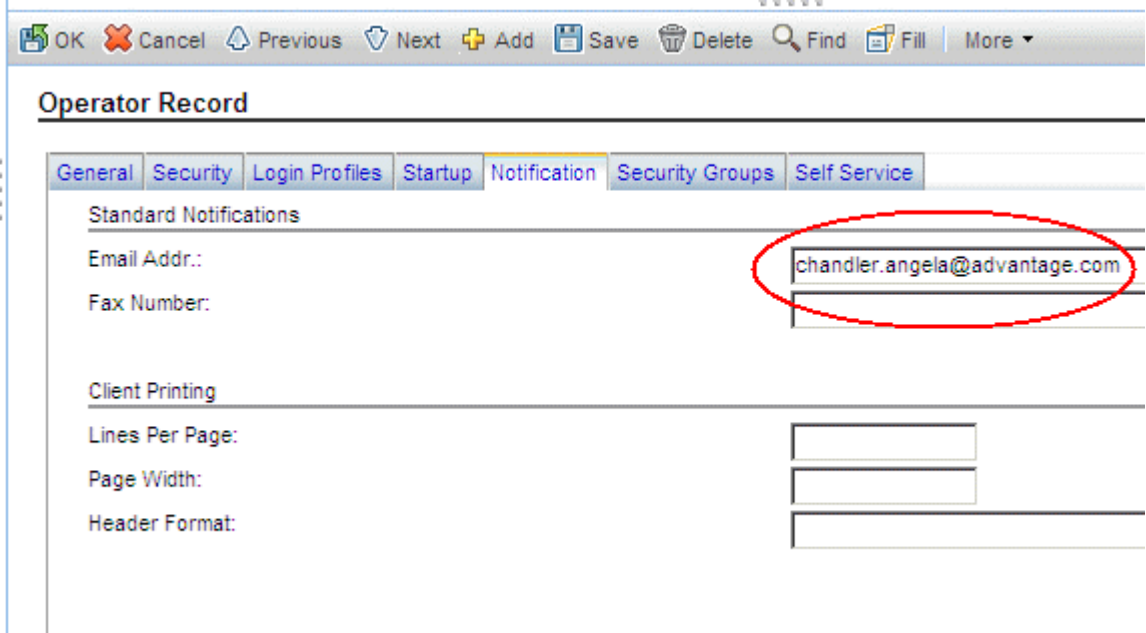
The `axces.write` application creates a character string of fields from a structure and writes them to eventout. An Event Registration record identifies the event type and the name of the Event Map records that defines which fields are selected from the record. The application is called as a Format Control subroutine passing two parameters; the first is the record from which data is mapped, and the second is the Event Type, as defined in the Event Register.

# Sending HP Service Manager mail to email

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

To modify a user operator file for email:

- 1 Login to HP Service Manager using a client with SysAdmin authority.
- 2 From the HP Service Manager System Navigation pane, click **System Administration > Ongoing Maintenance > Operators** to open the Operator form.
- 3 Enter search criteria to find the desired user, then click **Search**.
- 4 On the Notifications tab, enter the email address:



The screenshot shows the 'Operator Record' form in HP Service Manager. The 'Notification' tab is selected, and the 'Email Addr.' field is circled in red, containing the text 'chandler.angela@advantage.com'. The form also includes sections for 'Standard Notifications' and 'Client Printing' with various input fields.

- 5 Click Save to record the operator record.

---

## 6 Event Agent Operations

Automatic monitors within HP Service Manager, known as agents, can be set to collect data and create events appropriately within the system. You can use the Event Scheduler to set up these agents, or you can activate them automatically or manually (by user input).

Information about event agents includes:

- [Event scheduling](#) on page 72
- [Maintaining agent status](#) on page 74

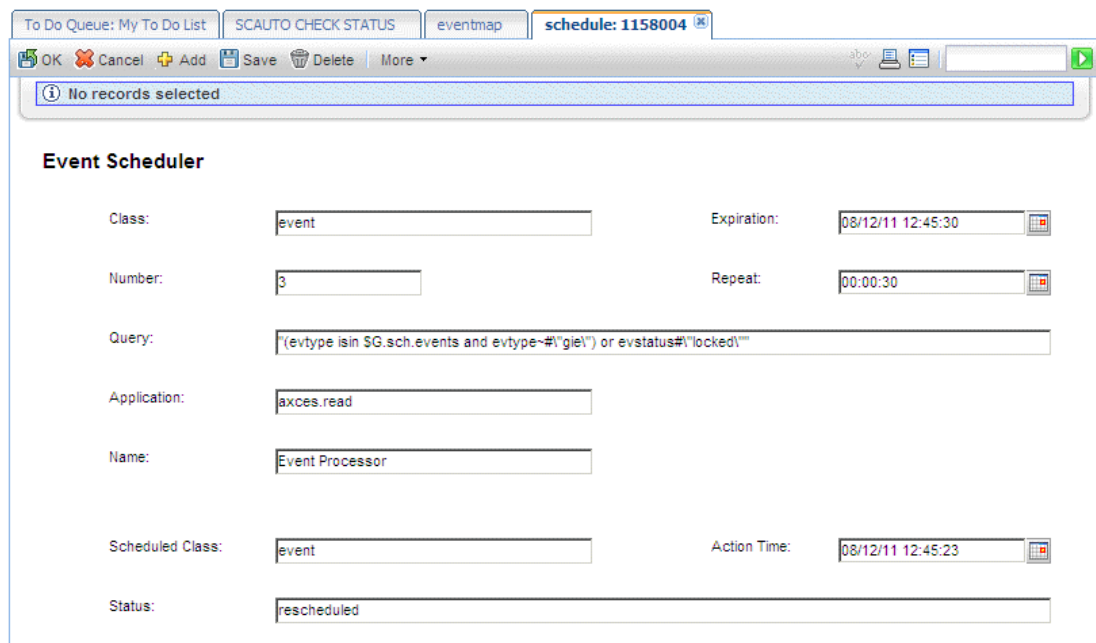
## Event scheduling

The schedule file contains a record for each SCAuto agent. It contains instructions indicating how often the agent reads a queue, and which application to execute if the read returns records.

### Reviewing scheduled events

To review SCAuto event schedules:

From the HP Service Manager System Navigator pane, click **Tailoring > Event Services > Review Agents**. The Event Scheduler displays:



The screenshot shows a web-based interface for the Event Scheduler. At the top, there are tabs for 'To Do Queue: My To Do List', 'SCAUTO CHECK STATUS', 'eventmap', and 'schedule: 1158004'. Below the tabs is a toolbar with buttons for 'OK', 'Cancel', 'Add', 'Save', 'Delete', and 'More'. A status bar indicates 'No records selected'. The main area is titled 'Event Scheduler' and contains several input fields:

- Class:** event
- Expiration:** 08/12/11 12:45:30
- Number:** 3
- Repeat:** 00:00:30
- Query:** (evtype isin SG.sch.events and evtype~#"glie") or evstatus#"locked"
- Application:** axces.read
- Name:** Event Processor
- Scheduled Class:** event
- Action Time:** 08/12/11 12:45:23
- Status:** rescheduled

### Schedule fields

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

Field (Internal name)	Description
Class (class)	Schedule class; must match the name of the agent as defined in the info startup record.
Expiration (expiration)	Data and time when the agent next is activated.
Number (number)	Unique number to identify the schedule record.
Repeat (repeat)	Interval defining the sleep time for the application.
Query (query)	Optional query that you can combine with the class to allow multiple agents.
Application (application)	Name of the HP Service Manager application that the agent calls.



Field (Internal name)	Description
Name (name)	Name associated with the agent. For OS/390/SCAuto and SCAuto for NetView OS/390 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, HP Service Manager 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 fails with an error.
Scheduled Class (sched.class)	Class that was used when the application last executed.
Action Time (action.time)	Last time the application executed.
Status (status)	Current status of the event scheduler: application running rescheduled application failed due to error

When the event agent starts, the event schedule record must have a Class of event (or the name you specify for the event scheduler) 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, particularly when inventory activity is high, preventing incident management activity.

Use the Query field to further define what type of event to select from the event in 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.



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



The agent processor attempts to restart any applications that ended while running (that have a status of application running). If you change for one of your agents, ensure there are no other agents with the same schedule class and a status of application running.

If you define a query for use against the event in file, ensure it is fully-keyed for maximum performance.

# Maintaining agent status

You can start and stop agents within HP Service Manager by using:

- System startup
- Status window
- Event agent check

## System startup

To view the startup info record:

- 1 From the HP Service Manager System Navigation pane, click **System Administration > Ongoing Maintenance > System > Startup Information**.
- 2 The Background Processor Initialization Registry form displays.
- 3 Enter startup in the Type field.
- 4 Click **Search**. The startup information displays:

The screenshot shows a web browser window displaying the 'Background Processor Initialization Registry' form. The form has a title bar with 'OK', 'Cancel', 'Add', 'Save', 'Delete', and 'More' buttons. The main content area is titled 'Background Processor Initialization Registry' and contains the following fields:

- Type: startup
- Description: system startup default
- Processor Information section:
  - Name: despooler
  - Supress Restart? (checkbox, unchecked)
  - RAD Application: scheduler
  - Class: spool
  - Wakeup Interval (secs.): 300
  - Priority: 0

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

## System status window

From the HP Service Manager System Navigation pane, click **System Status** or type **status** in the command line to display the system status window.

From this window you can start or stop (kill) individual agents by name.

The screenshot shows the 'System Status' window in HP Service Manager. At the top, there's a navigation bar with 'To Do Queue: My To Do List' and 'System Status'. Below that, a message says 'No records selected'. A status bar indicates 'TOTAL USERS: 1 - use Refresh Display to refresh statistics'. On the left, there are several control buttons: 'Refresh Display', 'Start Scheduler', 'Broadcast', 'Show Locks', 'Display Options', 'System Monitor', 'Summary', and 'Execute Commands'. The 'Start Scheduler' button is highlighted. To the right is a table of running agents.

Command	User Name	PID	Device ID
	falcon	3208	SOAP-Web
	TRCLIENT	3208	SYSTEM
	sync	1660	SYSTEM
	alert	1660	SYSTEM
	ocm	1660	SYSTEM
	contract	1660	SYSTEM
	availability	1660	SYSTEM
	event	1660	SYSTEM
	linker	1660	SYSTEM
	lister	1660	SYSTEM
	sla	1660	SYSTEM
	change	1660	SYSTEM
	problem	1660	SYSTEM
	report	1660	SYSTEM
	spool	1660	SYSTEM
	TRCLIENT	1660	SYSTEM
	ThreadControllerid-background	1660	SYSTEM
	ThreadControllerid-37718	3208	SYSTEM

Typically event agents are started and stopped automatically as needed by the HP Service Manager server. In some cases such as when recovering a system, you can start event agents manually by clicking the **Start Scheduler** button.



---

# 7 Format Control Options

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

This chapter contains the following sections:

- [Generating eventout records](#) on page 78
- [Generating page messages](#) on page 81
- [Sending fax messages](#) on page 83
- [Creating output events](#) on page 85

# Generating eventout records

Use Format Control to generate eventout records in Incident Management and Inventory and Configuration Management.

## Format Control

Application	Description
axces.write	Called from Format Control, builds an eventout record that the SCAutomate interface uses.

## Parameters

Name	Value	Default
record	The record to be written	none
name	The name of registration type	none
string1	The separation character	caret (^)
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 closes if this parameter is not provided.
- The registration name must exist in the eventregister file. If it does not, the application closes.
- If no eventregister record with a type of output can be found, the input registration record is used.
- Mapping is defined either by the format name or the map name. For most SCAuto/SDK events, use the Map Name to properly format fields.
- If you define a separation character, ensure that it is not one that occurs naturally in fields in the event.
- HP Service Manager generates the system sequence ID unless you supply one. The maximum length is 16 characters.

## Incident Management

When Event Services opens, updates, or closes problems, 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 using the SCAuto 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 `aces.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 are selected from the record. The application is called as a Format Control subroutine passing two parameters; the first is the record from which data is 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
<code>record</code>	<code>\$file</code>
<code>name</code>	<code>pmo</code>

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

➤ The Incident Management category example writes an `eventout` record for each open, update and close action.

➤ The standard event requires that specified fields populate in a specific 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 Event Services adds, updates, or deletes inventory items, 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 is passed to an external process using the `SCAuto` external interface. You can elect to write to the `eventout` file when operators maintain inventory items so that the information passes to the external interface.

The `aces.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 that defines which fields are selected from the record. The application is called as a Format Control subroutine passing two parameters; the first is the record from

which data is 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.

<b>Parameter Name</b>	<b>Parameter Value</b>
record	\$file
name	icma



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 are written to the eventout file using a subroutine call to `axces.page` from Format Control.

<b>Application</b>	<b>Description</b>
<code>axces.page</code>	Called from Format Control, builds an eventout record that the Telalert pager <code>axces</code> interface uses.

## Parameters

<b>Name</b>	<b>Value</b>	<b>Default</b>
<code>name</code>	The name of the contact	none
<code>prompt</code>	The numeric message	none
<code>text</code>	The alphanumeric message	none
<code>string1</code>	The separation character	caret (^)
<code>query</code>	The page response code	none
<code>values</code>	a list of addressees	none
<code>names,1</code>	a pager phone number	none
<code>names,2</code>	a pager PIN number	none
<code>names,3</code>	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 closes 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 receive a page as long as the associated contacts or operator record contains a pager phone number. Duplicate names receive only one page.
- The output event substitutes "" whenever a field is NULL except where noted.
- 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 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 is not processed. There are fields in the contacts file that define pager vendor, phone number, PIN, and so on. Complete these fields properly for successful paging to occur.
- If a parameter is passed in the query parameter, the `pageresp` input event uses it to identify the type of event processing that occurs. For example, to update a specific 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 determines 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 subroutine definition for the `axces.page`. For example, 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.

## Sending fax messages

SCAutomate supports a generic fax function using the Replix FAX product. You can write fax events to the eventout file using 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 HP Service Manager config table. This record limits the number of pages that a fax message sends. 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 HP Service Manager mail, or using Format Control, do not require a configuration record. By definition, their size cannot exceed 32,000 bytes.

### Format Control

Application	Description
<code>axces.fax</code>	Called from Format Control, builds an eventout 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	caret (^)
<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 Format Control processing to create output events based on 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 HP Service Manager and SCAutomate environments, see Tailoring in the HP Service Manager online help.

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 closes 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 allow you to enter the recipient name and telephone number.
- A separate fax is 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 configuration record must exist.
- The runoff application must have a compile date later than 5/14/96; reference SCR 7343.



# A Troubleshooting

The problems and solutions described in this appendix provide some assistance with configuring and using Event Services. In some cases the problem described may not match exactly the problem you are experiencing, but your problem may be exhibiting similar symptoms. In these case parts of the solutions described may be useful in resolving the issue you are having.

This chapter contains the following topics:

- [Why are no incidents opening, even though there are pmo records in the Event Input queue?](#) on page 87
- [Why is email not being received after opening a incident?](#) on page 89
- [How do I send email only for those incidents with a priority code of “emergency?”](#) on page 90
- [How do I know HP Service Manger mail sent to myself was received?](#) on page 90
- [How do I quickly test sending a fax message?](#) on page 91
- [How do I quickly test whether the SCAuto Pager is properly installed?](#) on page 91
- [How do I test sending a report to my external program once SCAuto/SDK is installed?](#) on page 91
- [How do I allow incident events to be processed separately, so they aren’t held up by other events?](#) on page 92
- [How do I test notifications to external programs after installing SCAuto](#) on page 92

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

Note: this troubleshooting procedure generally applies to all event types, not just *pmo*.

- 1 Verify the records in the queue have processed. From the HP Service Manager System Navigation pane, choose **Tailoring > Event Services > Input Events**. Then enter *pmo* in the Event Code field, and click Search.
  - If the records have processed, there should be a value in the Time Processed field.
  - The Status field should contain a value.
  - Any messages should appear in the Messages tab.
- 2 Verify there is an active event agent. From the HP Service Manager System Navigation pane, choose **Tailoring > Event Services > Agent Status**.
  - a For the Event agent, the Stop button should be enabled and a Last Expiration and an Idle time should appear.
  - b Click the Refresh button to reset the idle time to 00:00:00. It should begin increasing again.

- c If the Start button is enabled and there is no start and idle time, click Start and wait until the Event agent recycles.
  - 3 Verify the event scheduler. From the HP Service Manager System Navigation pane, choose **Tailoring > Event Services > Review Events**.
    - a The Class field should have a value of event.
    - b The Status field should have a value of rescheduled.
  - 4 If there is an active event agent ([step 2](#)), verify the event registration. From the HP Service Manager System Navigation pane, choose **Tailoring > Event Services > Registration**. Then enter *pmo* in the Event Code field, and click Search.
    - There should be entries for Event *pmo* with a Type of *input*.
    - On the Application tab, the Execute Condition should be *true*.
    - Verify the content in all fields of the *pmo* registration. See [Reviewing event registration](#) on page 16.
  - 5 Verify there is an event map matching the Event Map Name value specified in the registration record, Basics tab:

The image shows two overlapping windows from the HP Service Manager interface. The top window is titled "EVENT REGISTRATION" and has tabs for "Expressions", "Basics", and "Application". In the "Basics" tab, the "Event Code" field contains "pmo" and the "Event Map Name" field contains "problem open". The "Input or Output?" dropdown is set to "Input", and the "Process input events synchro" checkbox is checked. The bottom window is titled "Event Map" and has tabs for "Basics" and "Expressions". In the "Basics" tab, the "Map Name" field contains "problem open", the "Sequence" field contains "1", and the "Position" field contains "1". The "Type" dropdown is set to "Input". The "File Name" field contains "probsummary".



- On the Event Services Input Queue page, verify that all required fields in the External Information String, such as category, are provided and are valid. They must synch up with the mapped fields on the Event Map page:

System Option:

Field Separation Character:

External Information String:

Map Name	Seq	Pos	File Name	Field Name
problem open	1	1	probsummary	logical.name
problem open	1	2	probsummary	network.name
problem open	1	3	probsummary	reference.no
problem open	1	4	probsummary	cause.code
problem open	1	5	probsummary	\$ax.field.name
problem open	1	6	probsummary	action.?

## Why is email not being received after opening a incident?

- Verify that the intended email recipient is a member of the assignment group for the incident. Note that HP Service Manager does not send mail to the individual who is opening, updating or closing an incident, regardless of their membership in the assignment group.
 

To set or verify assignment groups, choose **System Administration > Ongoing Maintenance > Groups > Incident Management Assignments**.
- Verify that the intended email recipient has a valid email address specified in his or her operator record.
 

To set or verify operator properties, choose **System Administration > Ongoing Maintenance > Operators**. Search for the operator, then view the Notifications tab.
- Check the notification behavior for the IM Open class.
  - From the HP Service Manager System Navigation pane, choose **Tailoring > Notifications > Notifications**. Then enter IM Open in the Name field, and click Search.
  - Verify the conditions equate to true for the records where Notify Method is email.
- Verify there are records in the event output queue with a type of *email*. From the HP Service Manager System Navigation pane, choose **Tailoring > Event Services > Output Events**. Then enter email in the Event Code field, and click Search.
- Verify that the SCEMAIL agent or another email agent is active.
  - From the HP Service Manager System Navigation pane, choose **Tailoring > Event Services > Agent Status**.
  - For the SCEMAIL agent, the Stop button should be enabled and a last expiration and an idle time should appear.

- c Click the Refresh button to reset the idle time to 00:00:00. It should begin increasing again.
  - d If the Start button is enabled and there is no last expiration and idle time, click Start and wait until the SCEMAIL agent recycles.
- 6 If the SCEMAIL agent or another email agent is active and you still do not receive mail:
- a Stop the agent as described in [step 5](#).
  - b Open an incident and check the event output queue for new events with an Event Code of *email*. From the HP Service Manager System Navigation pane, choose **Tailoring > Event Services > Output Events**.
  - c If a new email event is added to the queue, restart the SCEMAIL agent or another email agent.
- Note that when the mail has been sent, the event is deleted if the Delete Condition field or `-d` flag is true.
- 7 Verify the event registration record.
- a From the HP Service Manager System Navigation pane, choose **Tailoring > Event Services > Event Registration**.
  - b Find the record with Event Code of *email*. Verify its contents as described in [Reviewing event registration](#) on page 16.

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

To view the Message Log choose **System Administration > Base System Configuration > Monitoring > Message Log**.

## How do I send email only for those incidents with a priority code of "emergency?"

There are two ways of sending email - by using notifications or by using macros.

If you are using notifications:

- 1 Click **Tailoring > Notifications > Notifications**.
- 2 Search for IM Open in the Name field.
- 3 For classes where Notify Method is *email*, modify the Condition field to include:  
(priority.code in \$L.new, "")="1"

If you are using macros:

- 1 Click **Tailoring > Tailoring Tools > Macros**.
- 2 Select the incidents macro that sends the email.
- 3 Change the Condition field value to:  
nullsub(priority.code in \$L.new, "")="1"

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

From the HP Service Manager System Navigation pane, choose **Miscellaneous > ServiceManager Mail**.

Your message should appear in the list of mail messages.

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

- 1 Click System Administration > Ongoing Maintenance > Communication Utilities > Send a FAX.
- 2 Create a sample message.
- 3 Click the Send FAX button.
- 4 From the HP Service Manager System Navigation pane, choose **Tailoring > Event Services > Output Events**.
- 5 Type fax for Event Code and click Search.
- 6 Review the event output queue.

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

- 1 Click System Administration > Ongoing Maintenance > Communication Utilities > Send a Page.
- 2 Create a sample page message.
- 3 Click the Page button.
- 4 If you are not paged within a minute or two, ensure the SCAUTO agent is active. Use the following procedure to do so:
  - a From the HP Service Manager System Navigation pane, choose **Tailoring > Event Services > Agent Status**.
  - b If SCAUTO Server is not active, start it.
- 5 If the SCAUTO agent is active and you still do not receive a page:
  - a From the HP Service Manager System Navigation pane, choose **Miscellaneous > System Status**. Stop (kill) the agent, by placing a lowercase letter 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.
- 6 If a new page event is added to the queue, restart the SCAUTO agent as described in [step 4](#).

After the page is sent, the event is deleted if the Delete Condition field or the `-d` flag is set to true.

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

- 1 Click **System Administration > Ongoing Maintenance > Communication Utilities > Write an Output Event**.
- 2 Select Incident Management.
- 3 Click Write Event.

- 4 Note the incident number.
- 5 Choose **Tailoring > Event Services > Output Event**.
- 6 Search for an event with a event code of pmo.

The Event Services Output Queue should include the test incident report you generated.

## How do I allow incident events to be processed separately, so they aren't held up by other events?

- 1 Choose **Tailoring > Event Services > Review Agents**.
- 2 Open the Event agent, and modify it as follows:
  - a Set the Class field to *probevent*.
  - a Modify the query field for the event agent to read `evtype~#"pm"`.
  - a Click Add to create the new event.
- 3 Open the Event agent again, and modify the query field for the probevent agent to read `evtype#"pm"`. Save the changes.
- 4 Manage the new probevent agent in the same way you manage the existing event agent.

## How do I test notifications to external programs after installing SCAuto

To send a test notification of a new device to an external program after installing SCAuto:

- 1 Choose **System Administration > Ongoing Maintenance > Communication Utilities > Write an Output Event**.
- 2 Select Configuration Management.
- 3 Click Write Event.
- 4 Note the event type.
- 5 Choose **Tailoring > Event Services > Output Events**.
- 6 Search for an event with a event code of icma.

The Event Services Output Queue should include the test item you generated.

---

## B Common Events

HP Service Manager delivers out-of-box events with the Event Services application.

Event Services provides a standard interface for user-defined applications and applications described in this section. You can call any RAD application that does not require user I/O as an event services application.

This appendix contains the following sections:

- [Service Desk events](#) on page 94
- [Incident Management events](#) on page 95
- [Inventory Management events](#) on page 96
- [Change Management events](#) on page 97
- [Request Management events](#) on page 98
- [Service Level Management events](#) on page 99
- [Standard events](#) on page 100

## Service Desk events

<b>Event</b>	<b>Description</b>
smin	Service Desk incoming service request or help issue.
smout	Service Desk Output event.

## Incident Management events

<b>Event</b>	<b>Description</b>
pmo	Opens an incident.
pmu	Updates an incident.
pmc	Closes an incident.

## Inventory Management events

<b>Event</b>	<b>Description</b>
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

<b>Event</b>	<b>Description</b>
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

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

## Service Level Management events

<b>Event</b>	<b>Description</b>
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

HP Service Manager event registration currently supports events enabling integration with ERP, SAP, and other external system interfaces. The following section contains the RAD routines that each event calls. For additional information about the parameters available for these routines, see the HP Service Manager online help. Where applicable, the parameter descriptions that follow contain information specific to the event.



Note: the events are listed in two sets: upper case event names appear first in alphabetical order, followed by lower case event names also in alphabetical order.

### CERPHR (1)

<b>Event type</b>	Input	
<b>Description</b>	This event establishes contact with the ERP system using eventmap contactserp.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	prompt	In this case, use contactserp.
	string1	In this case, use contacts.
	text	In this case, use add.
	query	In this case, use contact.name=1 in \$axces.fields.
	boolean1	In this case, the value is true.
	cond.input	In this case, the value is false.
	name	In this case, use operator .scauto.

### ERPHR (2)

<b>Event type</b>	Output
<b>Description</b>	This event uses eventmap contactserp.
<b>Routine called</b>	axces.write

## ERPSTATES (1)

<b>Event type</b>	Input	
<b>Description</b>	This event determines the state of the ERP system using eventmap stateerp.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	prompt	In this case, use stateerp.
	string1	In this case, use state.
	text	In this case, use add.
	query	In this case use, state.code=1 in \$axces.fields.
	boolean1	In this case, the value is true.
	cond.input	In this case, the value is false.
	name	In this case, use operator.scauto.

## ERPSTATES (2)

<b>Event type</b>	Input
<b>Description</b>	This event uses eventmap stateerp.
<b>Routine called</b>	axces.write

## HotNews

<b>Event type</b>	<b>Output</b>
<b>Description</b>	HotNews defines an eventout type of HotNews.
<b>Routine called</b>	None

## ICMapplication

<b>Event type</b>	<b>Input</b>	
<b>Description</b>	HP Service Manager inventory regulation event when a device of this type is added to the system.	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use icm application.
	string1	In this case, device.
	text	In this case, use add.
	query	In this case, logical.name=1 in \$axces.fields.
	boolean1	In this case, the value is false.
	name	In this case, use ICMapplication.

## ICMcomputer

<b>Event type</b>	<b>Input</b>	
<b>Description</b>	HP Service Manager inventory regulation event when a device of this type is added to the system.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use icm computer.
	string1	In this case, join computer.
	text	In this case, use add.
	query	In this case, logical.name=1 in \$axces.fields.
	boolean1	In this case, the value is false.
	name	In this case, use ICMcomputer.

## ICMdevice

<b>Event type</b>	Input	
<b>Description</b>	Use this event type when you add data records to the device file. These events use the icm device eventmaps.	
<b>Routine called</b>	icm.process.event	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	prompt	In this case, use icm device.
	string1	In this case, device.
	text	In this case, use add.
	query	In this case, logical.name=1 in \$axces.fields.
	boolean1	The default value is false.
	name	In this case, use ICMdevice.

## ICMdevicenode (1)

<b>Event type</b>	Input	
<b>Description</b>	Use this event type to add or update information about a network node (a device that appears as a discrete item in a network) to the Inventory Configuration Management application. These events use the icm networkcomponents mappings.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use icm networkcomponents.
	string1	In this case, joinnetworkcomponents.
	text	In this case, use add.
	query	In this case, logical.name=1 in \$axces.fields.
	boolean1	In this case, the value is false.
	name	In this case, use ICMnetworkcomponents.
	name,1	Specifies the formatctrl record to use for processing.

## ICMdevicenode (2)

<b>Event type</b>	Input
<b>Description</b>	If you use the ICMdevicenode event to send information to HP Service Manager, after the initial database operation completes, this secondary event registration causes a logging record to write to the eventout table.
<b>Routine called</b>	axces.write

## ICMdisplaydevice

<b>Event type</b>	Input	
<b>Description</b>	HP Service Manager inventory regulation event when a device of this type is added to the system.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use icm displaydevice.
	string1	In this case, joindisplaydevice.
	text	In this case, use add.
	query	In this case, logical.name=1 in \$axces.fields.
	boolean1	In this case, the value is false.
	name	In this case, use ICMdisplaydevice.

## ICMexample

<b>Event type</b>	Input
<b>Description</b>	HP Service Manager inventory regulation event when a device of this type is added to the system.
<b>Routine called</b>	axces.database



<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use icm example.
	string1	In this case, joinexample.
	text	In this case, use add.
	query	In this case, logical.name=1 in \$axces.fields.
	boolean1	In this case, the value is false.
	name	In this case, use ICMexample.

## ICMfurnishings

<b>Event type</b>	Input	
<b>Description</b>	HP Service Manager inventory regulation event when a device of this type is added to the system.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use icm furnishings.
	string1	In this case, joinfurnishings.
	text	In this case, use add.
	query	In this case, logical.name=1 in \$axces.fields.
	boolean1	In this case, the value is false.
	name	In this case, use ICMfurnishings.

## ICMhandhelds

<b>Event type</b>	Input	
<b>Description</b>	HP Service Manager inventory regulation event when a device of this type is added to the system.	
<b>Routine called</b>	axces.database	

<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use icm handhelds.
	string1	In this case, joinhandhelds.
	text	In this case, use add.
	query	In this case, logical.name=1 in \$axces.fields.
	boolean1	In this case, the value is false.
	name	In this case, use ICMhandhelds.

## ICMmainframe

<b>Event type</b>	Input	
<b>Description</b>	HP Service Manager inventory regulation event when a device of this type is added to the system.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use icm mainframe.
	string1	In this case, joinmainframe.
	text	In this case, use add.
	query	In this case, logical.name=1 in \$axces.fields.
	boolean1	In this case, the value is false.
	name	In this case, use ICMmainframe.

## ICMnetworkcomp

<b>Event type</b>	Input	
<b>Description</b>	HP Service Manager inventory regulation event when a device of this type is added to the system.	
<b>Routine called</b>	axces.database	

<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	string1	In this case, joinnetworkcomponents.
	text	In this case, use add.
	query	In this case, logical.name=1 in \$axces.fields.
	boolean1	In this case, the value is false.
	name	In this case, use ICMnetworkcomponents.

## ICMofficeelec

<b>Event type</b>	Input	
<b>Description</b>	HP Service Manager inventory regulation event when a device of this type is added to the system.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	string1	In this case, joinofficeelectronics.
	text	In this case, use add.
	query	In this case, logical.name=1 in \$axces.fields.
	boolean1	In this case, the value is false.
	name	In this case, use ICMofficeelectronics.

## ICMserver

<b>Event type</b>	Input	
<b>Description</b>	Use this event type to add or update information about a server to the Inventory Configuration Management application. These events use the icm computer mappings (since servers are a subtype of computers, you can reuse the mappings).	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use icm computer.

	string1	In this case, joincomputer.
	text	In this case, use add.
	query	In this case, logical.name=1 in \$axces.fields.
	boolean1	In this case, the value is false.
	name	In this case, use ICMserver.
	name,1	Specifies the formatctrl record to use for processing.

## ICMsoftwarelicense

<b>Event type</b>	Input	
<b>Description</b>	HP Service Manager inventory regulation event when a device of this type is added to the system.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use icm softwarelicense.
	string1	In this case, joinsoftwarelicense.
	text	In this case, use add.
	query	In this case, logical.name=1 in \$axces.fields.
	boolean1	In this case, the value is false.
	name	In this case, use ICMsoftwarelicense.

## ICMstorage

<b>Event type</b>	Input
<b>Description</b>	HP Service Manager inventory regulation event when a device of this type is added to the system.
<b>Routine called</b>	axces.database

<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use icm storage.
	string1	In this case, joinstorage.
	text	In this case, use add.
	query	In this case, logical.name=1 in \$axces.fields.
	boolean1	In this case, the value is false.
	name	In this case, use ICMstorage.

## ICMtelecom

<b>Event type</b>	Input	
<b>Description</b>	HP Service Manager inventory regulation event when a device of this type is added to the system.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use icm telecom.
	string1	In this case, jointelecom.
	text	In this case, use add.
	query	In this case, logical.name=1 in \$axces.fields.
	boolean1	In this case, the value is false.
	name	In this case, use ICMtelecom.

## IND

<b>Event type</b>	Input
<b>Description</b>	An event that adds or updates inventory items to the device file.
<b>Routine called</b>	scauto.inventory

<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use evmap in \$axces.register.
	string1	In this case, use device.
	text	In this case, use add.
	query	In this case, logical.name=7 in \$axces.fields.
	boolean1	In this case, the value is true.
	name	In this case, use icma, the legacy name.

## PSSDELETE

<b>Event type</b>	Input	
<b>Description</b>	This event deletes selected records from a HP Service Manager file.	
<b>Routine called</b>	pss.delete	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use evmap in \$axces.register.
	string1	In this case, use \$L.name.

## SALESQUOTE

<b>Event type</b>	Input	
<b>Description</b>	This event moves an eventin record to the eventout file and changes the evtype.	
<b>Routine called</b>	axces.move.intoout	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	name	In this case, SALESORDERPARSE.

## ScAcBrand

<b>Event type</b>	Input	
<b>Description</b>	This event allows HP Service Manager and AssetCenter to integrate data from the HP Service Manager vendor file to the corresponding AssetCenter file.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use ScAcBrand.
	string1	In this case, use vendor.
	text	In this case, use add.
	query	In this case, vendor = 1 in \$axces.fields.
	boolean1	In this case, the value is false.
	cond.input	In this case, the value is true.
	name	In this case, use vendor.

## ScAcCompany

<b>Event type</b>	Input	
<b>Description</b>	This event allows HP Service Manager and AssetCenter to integrate data from the HP Service Manager company file to the corresponding AssetCenter file.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use ScAcCompany.
	string1	In this case, use company.
	text	In this case, use add.
	query	In this case, customer.id = 1 in \$axces.fields.
	boolean1	In this case, the value is false.
	cond.input	In this case, the value is true.
	name	In this case, use company.

## ScAcContacts

<b>Event type</b>	Input	
<b>Description</b>	This event allows HP Service Manager and AssetCenter to integrate data from the HP Service Manager contacts file to the corresponding AssetCenter file.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use ScAcContacts.
	string1	In this case, use contacts.
	text	In this case, use add.
	query	In this case, contact.name = 1 in \$axces.fields.
	boolean1	In this case, the value is false.
	cond.input	In this case, the value is true.
	name	In this case, use contacts.

## ScAcDept

<b>Event type</b>	Input	
<b>Description</b>	This event allows HP Service Manager and AssetCenter to integrate data from the HP Service Manager department file to the corresponding AssetCenter file.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use ScAcDept.
	string1	In this case, use dept.
	text	In this case, use add.
	query	In this case, dept.id = 1 in \$axces.fields.
	boolean1	In this case, the value is false.
	cond.input	In this case, the value is true.
	name	In this case, use dept.



## ScAcDevice

<b>Event type</b>	Input	
<b>Description</b>	This event allows HP Service Manager and AssetCenter to integrate data from the HP Service Manager device file to the corresponding AssetCenter file.	
<b>Routine called</b>	scauto.inventory	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use evmap in \$axces.register.
	string1	In this case, use device.
	text	In this case, use add.
	query	In this case, logical.name = 1 in \$axces.fields.
	boolean1	In this case, the value is false.
	name	In this case, use i cma.

## ScAcLocation

<b>Event type</b>	Input	
<b>Description</b>	This event allows HP Service Manager and AssetCenter to integrate data from the HP Service Manager location file to the corresponding AssetCenter file.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use ScAcLocation.
	string1	In this case, use location.
	text	In this case, use add.
	query	In this case, location = 2 in \$axces.fields.
	boolean1	In this case, the value is false.
	cond.input	In this case, the value is true.
	name	In this case, use location.

## ScAcModel

<b>Event type</b>	Input	
<b>Description</b>	This event allows HP Service Manager and AssetCenter to integrate data from the HP Service Manager model file to the corresponding AssetCenter file.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use ScAcModel.
	string1	In this case, use model.
	text	In this case, use add.
	query	In this case, part.no = 1 in \$axces.fields.
	boolean1	In this case, the value is val("false",4).
	cond.input	In this case, the value is true.
	name	In this case, use model.

## ScAcModelBundle

<b>Event type</b>	Input	
<b>Description</b>	This event allows HP Service Manager and AssetCenter to integrate data from the HP Service Manager model file to the corresponding AssetCenter file.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use ScAcModelBundle.
	string1	In this case, use model.
	text	In this case, use add.
	query	In this case, part.no = 1 in \$axces.fields.
	boolean1	In this case, the value is val("false",4).
	cond.input	In this case, the value is true.
	name	In this case, use model.

## ScAcModelVendor

<b>Event type</b>	Input	
<b>Description</b>	This event allows HP Service Manager and AssetCenter to integrate data from the HP Service Manager modelvendor file to the corresponding AssetCenter file.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use ScAcModelVendor.
	string1	In this case, use modelvendor.
	text	In this case, use add.
	query	In this case, part.no = 1 in \$axces.fields and vendor = 2 in \$axces.fields.
	boolean1	In this case, the value is false.
	cond.input	In this case, the value is true.
	name	In this case, use modelvendor.

## ScAcVendor

<b>Event type</b>	Input
<b>Description</b>	This event allows HP Service Manager and AssetCenter to integrate data from the HP Service Manager vendor file to the corresponding AssetCenter file.
<b>Routine called</b>	axces.database

<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use ScAcVendor.
	string1	In this case, use vendor.
	text	In this case, use add.
	query	In this case, vendor = 2 in \$axces.fields.
	boolean1	In this case, the value is false.
	cond.input	In this case, the value is true.
	name	In this case, use vendor.

## ScAcVendorBACK

<b>Event type</b>	Input	
<b>Description</b>	This event allows HP Service Manager and AssetCenter to integrate data from the HP Service Manager vendor file to the corresponding AssetCenter file.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use ScAcVendor.
	string1	In this case, use vendor.
	text	In this case, use add.
	query	In this case, vendor=9 in \$axces.fields.
	boolean1	In this case, the value is true.
	cond.input	In this case, the value is false.

## approval

<b>Event type</b>	Output
<b>Description</b>	This event sends approvals for Request Management and Change Management.
<b>Routine called</b>	axces.write

## approval

<b>Event type</b>	Input	
<b>Description</b>	This event processes approvals for Request Management and Change Management.	
<b>Routine called</b>	es.approval	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	text	In this case, use ApprovalLog.
	name	In this case, evmap in \$axces.register.
	cond.input	In this case, the value is false.

## cm3rin

<b>Event type</b>	Input	
<b>Description</b>	Use this event for all incoming change events.	
<b>Routine called</b>	axces.cm3	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	Use the \$axces variable.
	prompt	In this case, evmap.
	text	In this case, use 3 in evlist in \$axces.
	boolean 1	In this case, the value is true.
	string1	In this case, use cm3r.

## cm3rinac

<b>Event type</b>	Output
<b>Description</b>	This is sent if the write eventout is set to true. This returns failed events so the calling application receives notification when an error occurs.
<b>Routine called</b>	axces.write

## cm3rout

<b>Event type</b>	Output
<b>Description</b>	This is created when a cm3 message is created and you enter cm3rout in axces.out.
<b>Routine called</b>	axces.write

## cm3tin

<b>Event type</b>	Input	
<b>Description</b>	Use this for all incoming change tasks.	
<b>Routine called</b>	axces.cm3	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	prompt	In this case, evmap in \$axces.register.
	text	In this case, use 3 in evlist in \$axces.
	boolean1	In this case, the value is true.
	string1	In this case, use cm3t.

## cm3tinac

<b>Event type</b>	Output
<b>Description</b>	This is sent if the write eventout is set to true. This returns failed events so the calling application is notified when an error occurs.
<b>Routine called</b>	axces.write

## cm3tout

<b>Event type</b>	Output
<b>Description</b>	This is created when a cm3 message fires and you enter cm3tout in axces.out.
<b>Routine called</b>	axces.write

## dbadd

<b>Event type</b>	Input	
<b>Description</b>	This adds an item to a specified HP Service Manager file when you satisfy the filter criteria. It updates the file if the item already exists.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	prompt	In this case, use scauto test.
	string1	In this case, use scautotest.
	text	In this case, use add.
	query	In this case, use field.1=1 in \$axces.fields.
	boolean1	In this case, the value is true.
	cond.input	In this case, the value is true.
	name	In this case, use scautotest.

## dbdel

<b>Event type</b>	Input	
<b>Description</b>	This deletes an item from a specified HP Service Manager file if the filter criteria are satisfied.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	prompt	In this case, use scauto test.
	string1	In this case, use scautotest.
	text	In this case, use delete.
	query	In this causes field.1=1 in \$axces.fields.
	boolean1	In this case, the value is true.
	cond.input	In this case, the value is true.
	condition,1	In this case, the value is false.
name	In this case, use scautotest.	

## dbupd

<b>Event type</b>	Input	
<b>Description</b>	This updates an item to a specified HP Service Manager file when you satisfy the filter criteria.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	prompt	In this case, use scauto test.
	string1	In this case, use scautotest.
	text	In this case, use update.
	query	In this case, field.1=1 in \$axces.fields.
	boolean1	In this case, the value is true.
	cond.input	In this case, the value is true.
	name	In this case, use scautotest.

## email

<b>Event type</b>	Output	
<b>Description</b>	This is the standard interface to convert HP Service Manager mail to standard email format.	
<b>Routine called</b>	axces.email	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	text	In this case, use ^.

## email

<b>Event type</b>	Input	
<b>Description</b>	This is the standard interface to receive external email and convert to HP Service Manager mail.	
<b>Routine called</b>	axces.email.receive	



epmosmu

<b>Event type</b>	Input	
<b>Description</b>	This event opens an incident from a call.	
<b>Routine called</b>	axces.apm	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use evmap in \$axces.register.
	string1	In this case, use probsummary.
	text	In this case, use open.
	query	In this case, use \$ax.query.passed.
	boolean1	In this case, the value is the statement nullsub(evstatus in \$axces,"")~#"error".
	cond.input	In this case, the value is the statement \$ax.open.flag.

epmosmu

<b>Event type</b>	Output	
<b>Description</b>	This event writes the record after an incident opens from a call.	
<b>Routine called</b>	axces.write	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	prompt	In this case, use nullsub(evuserseq in \$axces, evsysseq in \$axces).
	string1	In this case, use ^.
	query	In this case, evuser in \$axces.
	name	In this case, use pmo.

## esmin

<b>Event type</b>	Input	
<b>Description</b>	This event opens a call in Service Desk.	
<b>Routine called</b>	axces.sm	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, use the value of \$axces.
	prompt	In this case, evmap in \$axces.register.
	string1	In this case, incidents.
	query	In this case, \$ax.query.passed.
	boolean1	In this case, true.
	text	in this case, esmin.

## esmin

<b>Event type</b>	Output	
<b>Description</b>	This event writes a record once a call opens in Service Desk.	
<b>Routine called</b>	axces.write	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	name	In this case, use smout.
	string1	In this case, use ^.
	query	In this case, evuser in \$axces.
	prompt	In this case, use nullsub(evuserseq in \$axces, evsysseq in \$axces).

## gie

<b>Event type</b>	Input
<b>Description</b>	Both AssetCenter and HP Service Manager use the Generic Input Event (GIE).
<b>Routine called</b>	None

## icma

<b>Event type</b>	Input	
<b>Description</b>	This event adds or updates inventory items to the device file if filter criteria are satisfied.	
<b>Routine called</b>	scauto.inventory	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use evmap in \$axces.register.
	string1	In this case, use device.
	text	In this case, use add.
	query	In this case, use network.name=5 in \$axces.parameters.
	boolean1	In this case, the value is true.
	name	In this case, use icma.

## icmd

<b>Event type</b>	Input	
<b>Description</b>	This event marks an inventory item for deletion if filter criteria are satisfied by placing inactive in the status field.	
<b>Routine called</b>	scauto.inventory	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use evmap in \$axces.register.
	string1	In this case, use device.
	text	In this case, use delete.
	query	To select the device using the network name, use nullsub("network.name=\")+1 in \$axces.fields+"\\" or logical.name=\")+1 in \$axces.fields+"\\", "false".
	boolean1	In this case, the value is true.
	name	In this case, use icmd.

## icmswa

<b>Event type</b>	Input	
<b>Description</b>	This event adds or updates inventory items (that ServerView or StationView discovers) to the device file if filter criteria are satisfied.	
<b>Routine called</b>	axces.pfiles	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	text	In this case, use add.
	boolean1	In this case, the value is false.
	name	In this case, use pc.files.

## icmswd

<b>Event type</b>	Input	
<b>Description</b>	This event marks an inventory item (that ServerView or StationView discovers) for deletion in the pfiles file if filter criteria are satisfied.	
<b>Routine called</b>	axces.pfiles	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	text	In this case, use delete.
	boolean1	In this case, the value is false.
	name	In this case, use pc.files.

## icmu

<b>Event type</b>	Input	
<b>Description</b>	This event updates inventory items if filter criteria are satisfied.	
<b>Routine called</b>	scauto.inventory	

Parameter	Name	Description
	record	In this case, the value is \$axces.
	prompt	In this case, evmap in \$axces.register.
	string1	In this case, device.
	text	In this case, update.
	query	In this case, select the device using network name: network.name=5 in \$axces.fields.
	boolean1	In this case, the value is true.
	name	In this case, use icmu.

## opera

<b>Event type</b>	Input	
<b>Description</b>	This event adds or updates a new user to HP Service Manager if filter criteria are satisfied.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use operator.
	string1	In this case, use operator.
	text	In this case, use add.
	query	To select the operator, name=1 in \$axces.fields.
	boolean1	In this case, true.
	cond.input	In this case, false.
	name	In this case, operator.scauto.



**Important:** The default query selects the operator and adds a new user with the minimum privileges to access HP Service Manager: 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 to the name defined for the template operator record.

For example, you can set up an operator record named standarduser 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 changes from name=1 in \$axces.fields to name="standarduser".

## operd

<b>Event type</b>	Input	
<b>Description</b>	This event deletes a user from HP Service Manager if filter criteria are satisfied.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use operator.
	string1	In this case, use operator.
	text	In this case, use de;ete.
	query	To select the operator, name=1 in \$axces.fields.
	boolean1	In this case, true.
	cond.input	In this case, false.
	condition,1	In this case, true.
	name	In this case, operator.scauto.

## operu

<b>Event type</b>	Input
<b>Description</b>	This event updates items specified in a HP Service Manager file if filter criteria are satisfied.
<b>Routine called</b>	axces.database

<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use operator.
	string1	In this case, use operator.
	text	In this case, use update.
	query	To select the operator, name=1 in \$axces.fields.
	boolean1	In this case, true.
	cond.input	In this case, true.
	name	In this case, operator.scauto.

## outageend

<b>Event type</b>	Input	
<b>Description</b>	This event performs updates to outage records, which are part of the SLA application.	
<b>Routine called</b>	axces.outageend	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	This parameter writes the unique record identifier (for example, problem number) to the eventin record. If you use a variable such as \$axces, set it up using Format Control to create the intended result.

## outagestart

<b>Event type</b>	Input
<b>Description</b>	This event performs updates to outage records, which are part of the SLA application.

<b>Routine called</b>	axces.outagestart	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	This parameter writes the unique record identifier (for example, problem number) to the eventin record. If you use a variable such as \$axces, set it up using Format Control to create the intended result.

## page

<b>Event type</b>	Output
<b>Description</b>	This event registration allows HP Service Manager to create eventout records with the evtype=page.
<b>Routine called</b>	axces.write

## pageclose

<b>Event type</b>	Input	
<b>Description</b>	This event uses a condition statement (evfiends in \$axces)#"pm".	
<b>Routine called</b>	axces.apm	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use envmap in \$axces.register.
	string1	In this case, use problem.
	text	In this case, use close.
	query	To select the operator, use "number=\""+substr(1 in \$axces.fields, 3, lng(1 in \$axces.fields) - 2)+"\"".
	boolean1	In this case, use false.



## pageresp

<b>Event type</b>	Input	
<b>Description</b>	This event updates an incident with an acknowledgment or message received as response to a page. It uses a condition statement (evfiends in \$axces)#"pm".	
<b>Routine called</b>	axces.apm	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use envmap in \$axces.register.
	string1	In this case, use problem.
	text	In this case, use update.
	query	To select the operator, use "number=\""+1 in \$axces.fields+"\"".
	boolean1	In this case, use false.

## pcsoftware

<b>Event type</b>	Input	
<b>Description</b>	This event allows desktop inventory products to update HP Service Manager.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use pcsoftware.
	string1	In this case, use pcsoftware.
	text	In this case, use add.
	query	In this case, logical.name=20 in \$axces.fields and license.number=2 in \$axces.fields and application.name=1 in \$axces.fields.
	boolean1	In this case, false.
	cond.input	In this case, true.

<b>Event type</b>	Input	
<b>Description</b>	This event closes an incident if filter criteria are met. It uses the same path as manually closing the operation.	
<b>Routine called</b>	axces.apm	
<b>Initialization expressions</b>	<pre>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("IND", evuser in \$axces)&gt;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</pre>	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use envmap in \$axces.register.
	string1	In this case, use probsummary.
	text	In this case, use close.
	query	In this case, use \$axces.query.passed.
	boolean1	In this case, the value is nullsub(evstatus in \$axces, "close")~#"error".

<b>Event type</b>	Output
<b>Description</b>	This event writes after an incident is closed.
<b>Routine called</b>	axces.write

Parameters	Name	Description
	record	In this case, the value is \$axces.
	name	In this case, pmc.
	string1	The delimiter character is ^.
	prompt	In this case, nullsub(evusrseq in \$axces, evsysseq in \$axces).
	query	In this case, evuser in \$axces.

pmo

<b>Event type</b>	Input	
<b>Description</b>	This event opens an incident if filter criteria are met. It uses the same path as manually opening the incident.	
<b>Routine called</b>	axces.apm	
<b>Initialization expressions</b>	<pre> \$ax.query.passed=nullsub("flag=true and network.name=\""+2 in \$axces.fields+"\", "false") if (index("axmail", evuser in \$axces)&gt;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)&gt;0) then (\$ax.open.flag=true) \$axces.lock.interval='00:00:30' if (index("IND", evuser in \$axces)&gt;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 </pre>	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use envmap in \$axces.register.
	string1	In this case, use probsummary.
	text	In this case, use open.
	query	In this case, use \$axces.query.passed.
	boolean1	In this case, the value is nullsub(evstatus in \$axces, "")~#"error".
	cond.input	In this case, the value is \$ax.open.flag.

pmo

<b>Event type</b>	Output	
<b>Description</b>	This event writes after an incident is opened.	
<b>Routine called</b>	axces.write	
<b>Parameters</b>	Name record	Description In this case, the value is \$axces.
	name	In this case, pmo.
	string1	The delimiter character is ^.
	query	In this case, evuser in \$axces.
	prompt	In this case, nullsub(enusrseq in \$axces, evsysseq in \$axces).

pmu

<b>Event type</b>	Input
<b>Description</b>	This event updates an incident if filter criteria are met. It uses the same path as manually updating the incident.
<b>Routine called</b>	axces.apm
<b>Initialization expressions</b>	<pre>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") \$byypass.failed.validation=true \$axces.bypass.failed.validation=true</pre>

<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use envmap in \$axces.register.
	string1	In this case, use probsummary.
	text	In this case, use update.
	query	In this case, use \$axces.query.passed.
	boolean1	In this case, the value is nullsub(evstatus in \$axces, "update")~#"error".

## pmu

<b>Event type</b>	Output	
<b>Description</b>	This event writes after an incident is updated.	
<b>Routine called</b>	axces.write	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	name	In this case, pmu.
	string1	The delimiter character is ^.
	query	In this case, evuser in \$axces.
	prompt	In this case, nullsub(evusrseq in \$axces, evsysseq in \$axces).

## prgma

<b>Event type</b>	Input
<b>Description</b>	This adds or updates a software inventory item to the pcfiles file that an external agent (other than ServerView or StationView) discovers if filter criteria are satisfied.
<b>Routine called</b>	axces.software

<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use software.
	string1	In this case, use pfiles.
	query	In this case, logical.name=1 in \$axces.fields and description=9 in \$axces.fields.
	name	In this case, pc.files.
	boolean1	In this case, false.

prgmd

<b>Event type</b>	Input	
<b>Description</b>	<p>This deletes a software inventory item from the pfiles file that an external agent (other than ServerView or StationView) discovers if filter criteria are satisfied.</p> <p>The default updates the estatus field as deleted rather than removing the record from the database.</p>	
<b>Routine called</b>	axces.software	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use pfiles.
	string1	In this case, use software.
	query	In this case, logical.name=1 in \$axces.fields and description=9 in \$axces.fields.
	name	In this case, pc.files.
	boolean1	In this case, false.
	cond.input	In this case, true.
	condition,1	In this case, false.

## prgmu

<b>Event type</b>	Input	
<b>Description</b>	This updates an inventory item in the pfiles file that an external agent (other than ServerView or StationView) discovers if filter criteria are satisfied.	
<b>Routine called</b>	axces.software	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, use pfiles.
	string1	In this case, use software.
	query	In this case, logical.name=1 in \$axces.fields and description=9 in \$axces.fields.
	name	In this case, pc.files.
	boolean1	In this case, false.
	cond.input	In this case, false.

## rmlin

<b>Event type</b>	Input	
<b>Description</b>	This provides access to Request Management line items.	
<b>Routine called</b>	axces.rm	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, it passes the value of \$axces.
	prompt	In this case, use evmap in \$axces.register.
	string1	In this case, use ocml.
	text	In this case, use 3 in evlist in \$axces.
	query	In this case, use number=1 in evlist in \$axces.

## rmoappr

<b>Event type</b>	Input	
<b>Description</b>	This provides access to Request Management order approval.	
<b>Routine called</b>	axces.rm	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, it passes the value of \$axces.
	prompt	In this case, use evmap in \$axces.register.
	string1	In this case, use ocmo.
	text	In this case, use 3 in evlist in \$axces.
	query	In this case, use number=1 in evlist in \$axces.

## rmoin

<b>Event type</b>	Input	
<b>Description</b>	This provides access to Request Management order input.	
<b>Routine called</b>	axces.rm	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, it passes the value of \$axces.
	prompt	In this case, use evmap in \$axces.register.
	string1	In this case, use ocmo.
	text	In this case, use 3 in evlist in \$axces.
	query	In this case, use number=1 in evlist in \$axces.

## rmqappr

<b>Event type</b>	Input	
<b>Description</b>	This provides access to Request Management quote approval.	
<b>Routine called</b>	axces.rm	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, it passes the value of \$axces.



	prompt	In this case, use evmap in \$axces.register.
	string1	In this case, use ocql.
	text	In this case, use 3 in evlist in \$axces.
	query	In this case, use \$L.approve.action.

## rmqin

<b>Event type</b>	Input	
<b>Description</b>	This provides access to Request Management quote input.	
<b>Routine called</b>	axces.rm	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, it passes the value of \$axces.
	prompt	In this case, use evmap in \$axces.register.
	string1	In this case, use ocmq.
	text	In this case, use 3 in evlist in \$axces.
	query	In this case, use number=1 in evlist in \$axces.

## slaresponse

<b>Event type</b>	Output	
<b>Description</b>	This is a request from an external application to enter a response time metric against a device with an SLA.	
<b>Routine called</b>	axces.postresponse	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	This parameter writes the unique record identifier (for example, problem number) to the eventin record. If you use a variable such as \$axces, set it up using Format Control to create the intended result.

## smin

<b>Event type</b>	Input	
<b>Description</b>	This event accesses Service Desk incoming service requests or help issues.	
<b>Routine called</b>	axces.sm	
<b>Initialization expressions</b>	<pre>\$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")</pre>	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, evmap in \$axces.register.
	string1	In this case, incidents.
	query	In this case, \$ax.query.passed.
	boolean1	In this case, true.

## smout

<b>Event type</b>	Output	
<b>Description</b>	This event writes once an incoming service request or help issue enters the system.	
<b>Routine called</b>	axces.write	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	record	In this case, the value is \$axces.
	prompt	In this case, nullsub(evusrseq in \$axces, evsysseq in \$axces).
	name	In this case, smout.
	string1	The delimiter character is ^.
	query	In this case, evuser in \$axces.

## submit

<b>Event type</b>	Output
<b>Description</b>	This event submits a job for processing.
<b>Routine called</b>	axces.write

<b>Event type</b>	Input	
<b>Description</b>	This event adds a new System Bulletin to HP Service Manager if filter criteria are satisfied.	
<b>Routine called</b>	axces.database	
<b>Parameters</b>	<b>Name</b>	<b>Description</b>
	prompt	In this case, use bulletin.
	string1	In this case, use bulletin.
	text	In this case, use add.
	query	In this case, use date=date(val(str(1 in \$axces.fields),3)).
	boolean1	In this case, the value is true.
	cond.input	In this case, the value is false.



Note: The system bulletin record is for today's date. For example, if today is New Year's Day, the bulletin is for 01/01/05 00:00, or the one with the default flag set to true.



Warning: Do not modify the application names or parameters unless you are completely familiar with RAD programming.