HP SCAuto for Operations Manager for Windows

for supported Windows® operating systems

Software Version: 1.50

Installation and Configuration Guide

Document Release Date: October 2010
Software Release Date: March 2010

Legal Notices

Warranty

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

The information contained herein is subject to change without notice.

Restricted Rights Legend

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

Copyright Notices

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com). Smack software copyright © Jive Software, 1998-2004. SVG Viewer, Mozilla JavaScript-C (SpiderMonkey), and Rhino software Copyright © 1998-2004 The Mozilla Organization. This product includes software developed by the OpenSSL Project for use in the OpenSSL toolkit. (http:// www.openssl.org). OpenSSL software copyright 1998-2005 The OpenSSL Project. All rights reserved. This project includes software developed by the MX4J project (http:// mx4j.sourceforge.net). MX4J software copyright © 2001-2004 MX4J Team. All rights reserved. JFreeChart software © 2000-2004, Object Refinery Limited. All rights reserved. JDOM software copyright © 2000 Brett McLaughlin, Jason Hunter. All rights reserved. LDAP, OpenLDAP, and the Netscape Directory SDK Copyright © 1995-2004 Sun Microsystems, Inc. Japanese Morphological Analyzer © 2004 Basis Technology Corp. The Sentry Spelling-Checker Engine Copyright © 2000 Wintertree Software Inc. Spell Checker copyright © 1995-2004 Wintertree Software Inc. CoolMenu software copyright © 2001 Thomas Brattli. All rights reserved. Coroutine Software for Java owned by Neva Object Technology, Inc. and is protected by US and international copyright law. Crystal Reports Pro and Crystal RTE software © 2001 Crystal Decisions, Inc., All rights reserved. Eclipse software © Copyright 2000, 2004 IBM Corporation and others. All rights reserved. Copyright 2001-2004 Kiran Kaja and Robert A. van Engelen, Genivia Inc. All rights reserved. Xtree copyright 2004 Emil A. Eklund. This product includes software developed by the Indiana University Extreme! Lab (<http:// www.extreme.indiana.edu/>). Portions copyright © Daniel G. Hyans, 1998. cbg.editor

Trademark Notices

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

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 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

Contents

1 Overview

Welcome to the *HP SCAuto for Operations Manager for Windows User's Guide*. This guide provides instructions on how to implement the interface between HP Operations Manager for Windows (OMW) and HP Service Manager or HP ServiceCenter by using SCAuto. HP Operations Manager for Windows was formerly known as HP OpenView Operations (OVO) for Windows.

Throughout this guide, we will refer to Service Manager only, however, the instructions work for both. If you are using ServiceCenter just replace the term "Service Manager" with "ServiceCenter".

This chapter provides an overview of this guide.

It covers these main topics:

Overview of SCAuto for OMW

SCAuto for OMW allows you to automate the process of creating, updating, and closing trouble tickets. It has the capability to annotate, own, and acknowledge OMW messages from modifications done on Service Manager incident tickets.

This product is part of the suite of SCAuto interface products that integrate Service Manager with other management tools. The interface is based on event messages sent over TCP connection to the Service Manager server. Additional information about SCAuto can be found in the *SCAuto Applications User's Guide*.

What's New

This version adds i18n support of messages exchanged between HP Service Manager and HP Operations Manager for Windows.

Included in this Book

The *HP SCAuto for Operations Manager for Windows Installation and Configuration Guide* includes the following chapters.

This chapter	Provides
Introduction on page 15	An introduction to SCAuto for OMW.

This chapter	Provides
Installation on page 35	The system requirements and steps you need follow to install SCAuto for OMW.
Configuration on page 43	A description of how to configure SCAuto for OMW, including SCAuto for OMW business logic, OMW business logic and Service Manager business logic.
Maintenance on page 69	Information on how to operate SCAuto for OMW, including starting and stopping it, basic parameter and configuration and troubleshooting.

What You Need to Know

This guide assumes the reader has:

- Working knowledge of Service Manager applications, client/server, and the OMW graphical user interface, as well as a basic understanding of Service Manager applications and Event Services.

  While some procedures for these applications are explained, others are referenced. Refer to the appropriate Service Manager documentation for a more detailed explanation.

- Familiarity with OMW and its components.
- Thorough knowledge of the operating systems where Service Manager, and the SCAuto for OMW product will be installed and implemented.

2 Introduction

This chapter provides an introduction to HP SCAuto for Operations Manager for Windows (OMW) and a high-level view of the operational concepts in this integration, as well as the product architecture.

It covers these main topics:

SCAuto for OMW

SCAuto for OMW offers integration between Service Manager applications and OMW. The interface of these two applications allows systems management functions of OMW to be extended and enhanced by the Incident Management functions of Service Manager.

This integration benefits customers of Service Manager and OMW by providing these features:

- A more robust environment for production IT operations
- Automated ticketing functions supported by the Incident Management processes
- The ability to structure and organize the real-time responses supported by the enterprise systems management functions.

Service Manager

Service Manager contains the following applications:

- Incident Management, which is a incident-tracking tool integrated with knowledge tools to speed resolution.
- Configuration Management, which maintains an operational database of assets used in the enterprise.
- Change Management, which allows you to manage the evolution of the enterprise to meet changing needs and requirements as they arise.
- Problem Management, which a problem-tracking tool integrated with knowledge tools to speed resolution.
- Service Desk provides a call-based front end applications.
- Service Level Management (SLM) offers significant value with service desk automation based on service level agreements (SLAs).
- Request Management maintains catalogs of services and items and organizes the delivery of these.
- Scheduled Maintenance enables you to set up and execute recurring tasks.

Operations Manager for Windows

HP Operations Manager for Windows (OMW) is a distributed, client/server software solution designed to provide service-driven event and performance management of business-critical enterprise systems, applications, and services.

OMW enables management of distributed, heterogeneous e-business infrastructures and includes support for a broad range of Windows and UNIX systems and applications, including e-commerce, web and application servers, conferencing and email, databases, ERP software, and more.

OMW provides console and server functionality to centrally monitor performance and events using agents installed on nodes being managed. To install add-ons and SPIs, you must first install OMW on a management server.

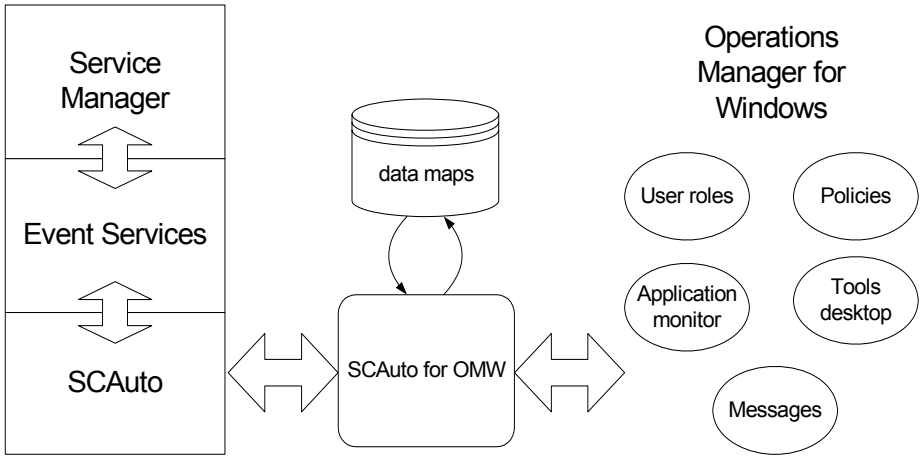
About SCAuto for OMW

SCAuto for OMW performs the following data processing functions:

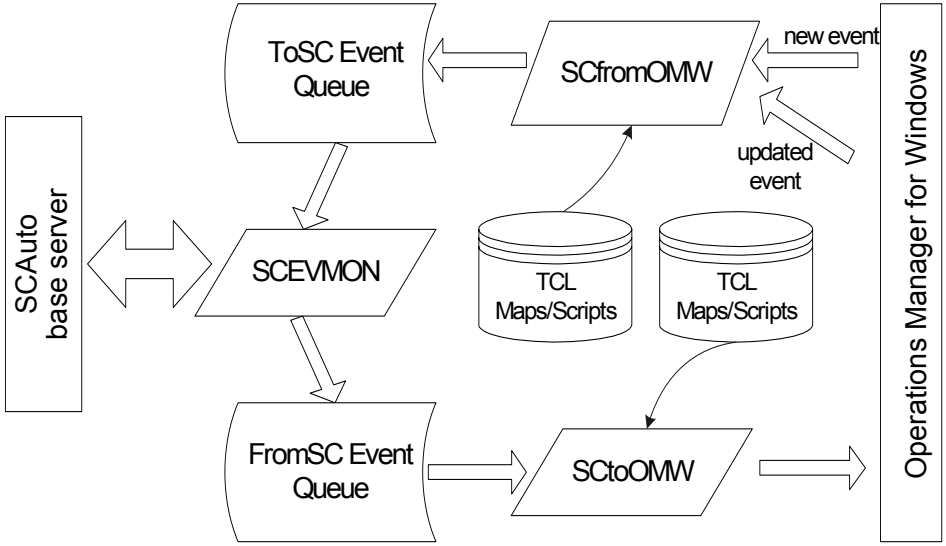
- Delivers notice whenever new events arrive at the event console.
- Delivers messages upon status changes to existing events.
- When event messages match the conditions of policies, incident tickets may be opened automatically. This minimizes the configuration of the SCAuto for OMW adapter and allows the user to configure OMW. It creates an extension to make OMW interact with Service Manager automatically.

New events received by OMW are output to SCAuto for OMW. This allows Service Manager to open new tickets. Subsequent actions against the event message (such as annotation or acknowledgment of the event) are output to SCAuto for OMW.

The following figure shows the architectural block diagram of the integration and the adapter.



The following figure shows the components of SCAuto for OMW in a functional block diagram.



In addition to the programmatic connections, other components within OMW are configured by the installation of SCAuto for OMW.

SCAuto for OMW also uses components that are a part of the standard SCAuto Software Development Kit (SDK). Key modules from the SDK are the event monitor (scevmon), the bi-directional event queues (ToSC and FromSC), and the event maps that formulate the conversion of incoming or outgoing messages.

Operational Concepts

The background of the static components was discussed previously. This section discusses the dynamic nature of the integration, which is a critical part of understanding the overall integration.

OMW is essentially a real-time management tool, while Service Manager delivers a process-oriented framework for operations. The synthesis of these domains is a dynamic environment where events and state changes drive procedure and process, and vice versa. This synthesis offers dramatic benefits for the customers of Service Manager and OMW.

Bi-Directional Integration

The default mode of SCAuto for OMW operation is the bi-directional integration of Service Manager and OMW. In this mode, OMW events trigger Service Manager processes. By default, selected events automatically open incident tickets. Subsequent actions or event messages at OMW may update or close the tickets, or a similar exchange may occur from Service Manager into OMW.

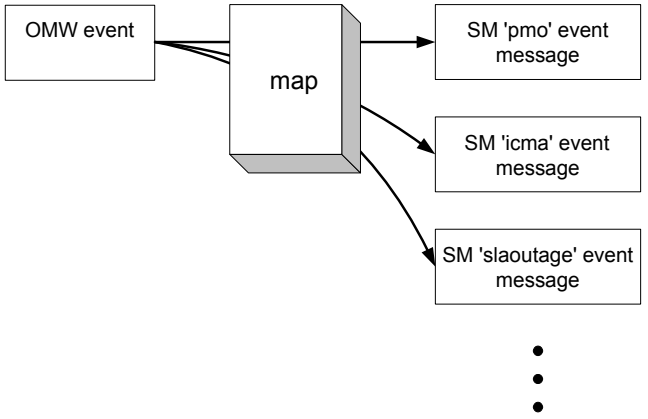
Because most Service Manager applications can accept and generate event messages, this same operational flow can be applied to these Service Manager applications as well. This default mode of operation can be configured to meet specific end user requirements. The section Planning your Service Manager and OMW Integration on page 24 describes the modes of operation that can be derived from the default.

All event messages that are exchanged between Service Manager and OMW must be converted from their native format into a format compatible with the destination. Event messages sent to Service Manager must be formatted in the event message structure defined by the SCAuto SDK. SCAuto for OMW performs most of this formatting, but it relies upon ASCII text map files to specify how to convert specific OMW event message data fields into Service Manager event message data fields. A similar process is used for converting outbound event messages from Service Manager to OMW.

The adapter may create one-to-many relationships of OMW events to incoming Service Manager event messages. This functionality is facilitated by the input maps that use the TCL scripting language. With this feature, for example, an SNMP Node Down event message can do any of the following:

- Open an incident ticket.
- Update a Configuration Management record.
- Start an SLA outage metric against a logical item, such as a database that runs on the node and is reported as down. (This is only available for installations with Service Level Management.)

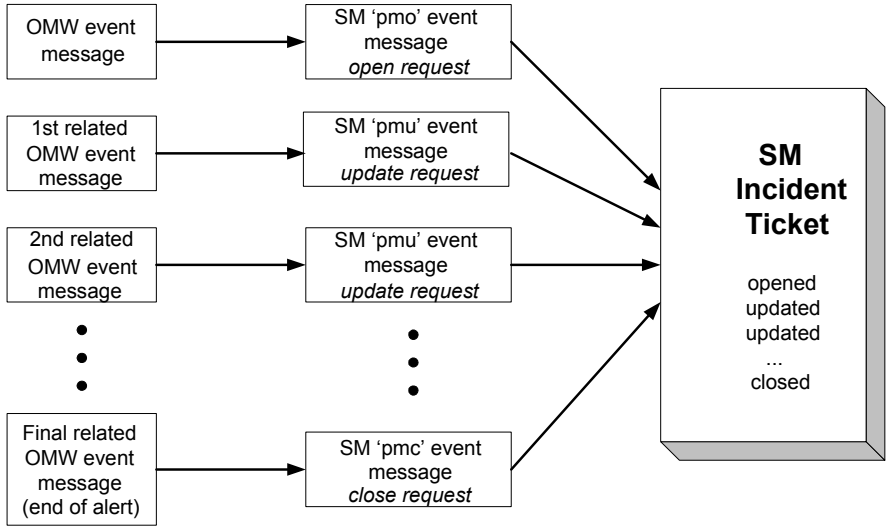
The following figure shows a one OMW Event Message to Many Service Manager (SM) Event Messages.



The adapter can also create one-to-many relationships of outgoing Service Manager events to OMW event messages or other actions. This output functionality is also supported by the use of TCL scripting language in the output maps. Since the maps are really script programs built to massage data fields, they can be edited easily to cause any desired actions.

The typical event message relationship will be one OMW event message to one Service Manager event message. Furthermore, many Service Manager event messages will be used to alter the state and the data of just one Service Manager incident ticket. For example, a OMW event message will create a Service Manager event message that will open an incident ticket. Then another OMW event message creates an update request type of Service Manager event message that adds new information to the original incident ticket. Then a final OMW event message will create a close request that ends the active life of the ticket.

The following figure shows One-to-One Event Message Relationships, with Many Event Messages Linked to Just One Incident Ticket.



Planning your Service Manager and OMW Integration

The following sections describe some of the ways in which SCAuto for OMW can be used. These scenarios help you plan and design your own implementation of OMW integrated with Service Manager using SCAuto for OMW. Any mode, combination of modes, or all modes can be used in a single deployment of SCAuto for OMW. Furthermore, an end user can extend the product to implement unique operational processes not described in any of the following mode sections.

The default mode is bi-directional, but you can also set up a uni-directional mode of operation.

Bi-Directional Exchange (Default Mode of Operation)

In this mode, OMW events drive Service Manager tickets (records), and the Service Manager tickets control OMW events. This creates a partnership of managing the events, where each application has a significant contribution to the event and incident management process.

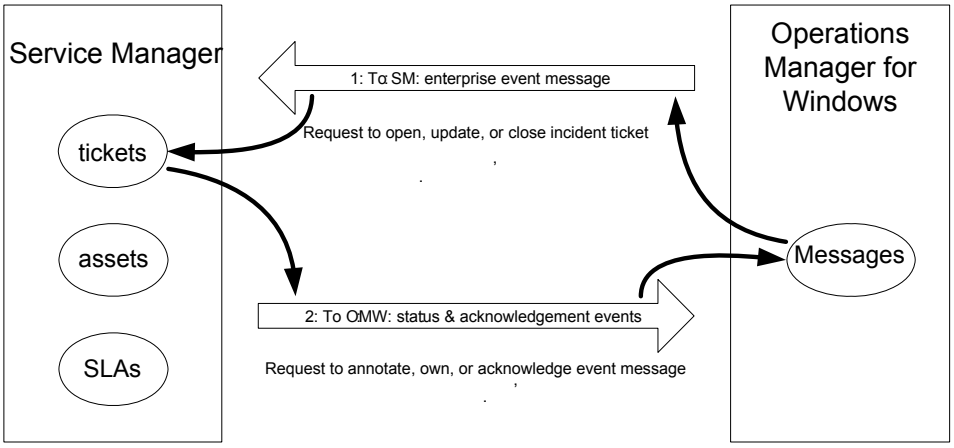
From this bi-directional interface, there are multiple scenarios based upon state transitions and the path of service processing. For example, there are three actions on OMW events that are able to be taken (via specific event messages): annotate, acknowledge, or own. There are many more actions that can be taken on tickets, but the various instances always represent a generalized open, update, or close action on the ticket.

This mode of application partnership is the typical operational mode. It offers service desk functions to extend and enhance the real-time event management of OMW. It moves the service desk into more proactive fault management, allowing service desk analysts to respond to emerging issues, rather than reacting to fully developed faults.

This mode allows a richer protocol of integration. In effect, OMW may request the opening of tickets. In response, Service Manager acknowledges the open ticket and annotates the OMW event. Subsequent exchanges may inform the applications of changes in state and data values. This mode also supports a complete cycle of interaction in which a close ticket closes (acknowledges) an OMW event, and vice versa. Building this type of interaction involves business logic in both Service Manager and OMW.

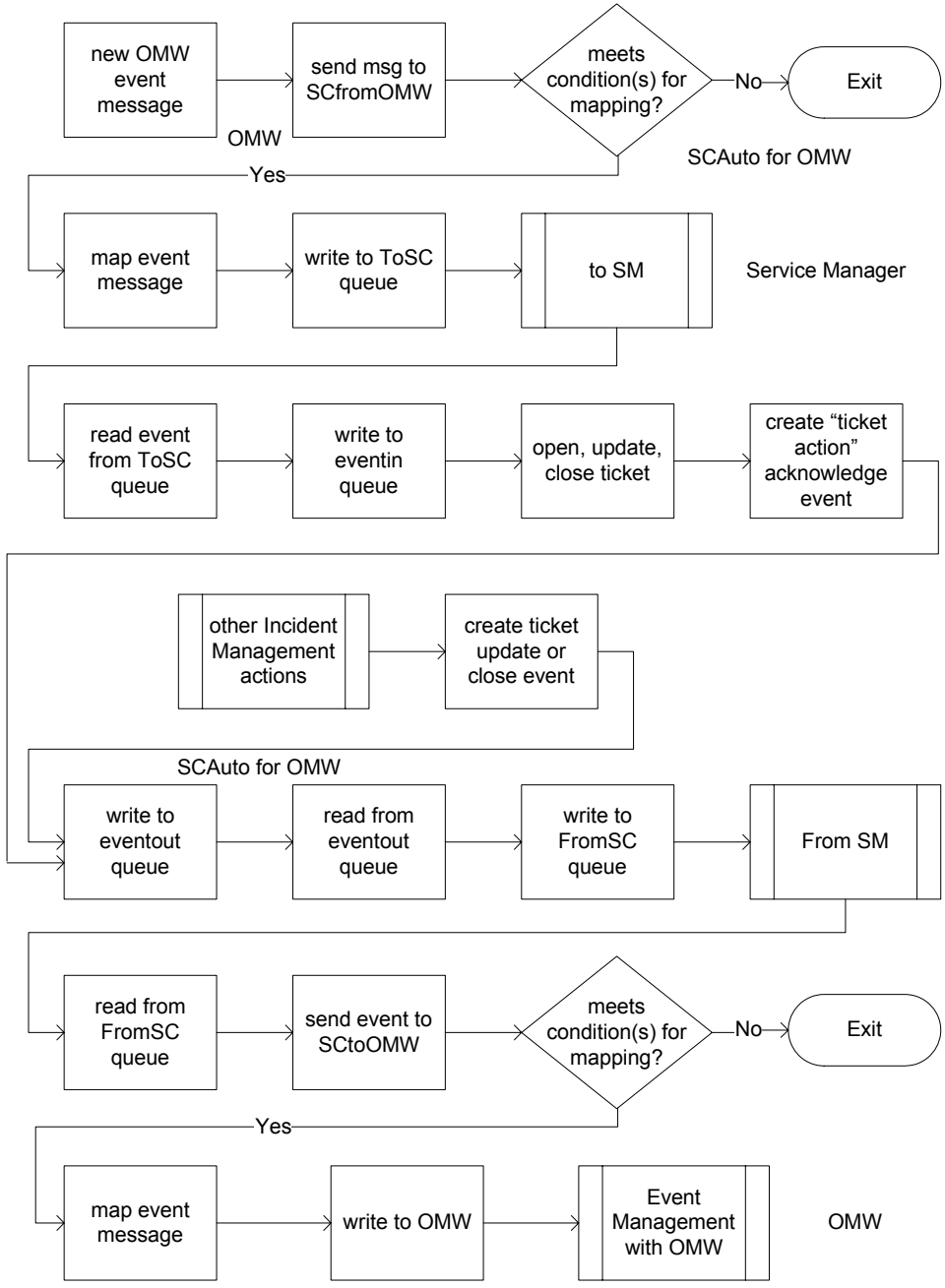
Process and Data Flow

This mode follows a process and data flow as shown in the following two flowchart diagrams.



This mode is constructed through the registered connection from OMW to Service Manager. This mode will invoke the following components of SCAuto for OMW:

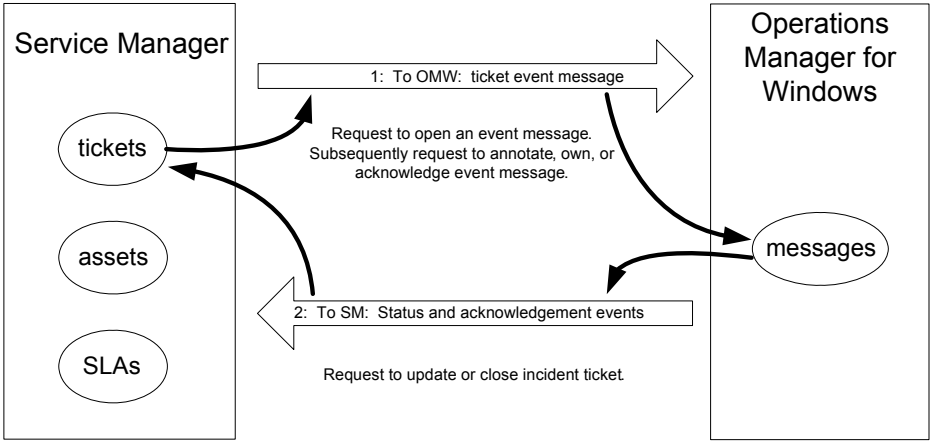
- SCfromOMW, SCtoOMW, and scevmon processes
- ToSC and FromSC Queues
- Event Maps (`event.ini`, and maps referenced in `event.ini`)


new OMW event message
send msg to SCfromOMW
meets condition(s) for mapping?
No
Exit
OMW
SCAuto for OMW
Yes
map event message
write to ToSC queue
to SM
Service Manager
read event from ToSC queue
write to eventin queue
open, update, close ticket
create "ticket action" acknowledge event
other Incident Management actions
create ticket update or close event
SCAuto for OMW
write to eventout queue
read from eventout queue
write to FromSC queue
From SM
read from FromSC queue
send event to SCtoOMW
meets condition(s) for mapping?
No
Exit
Yes
map event message
write to OMW
Event Management with OMW
OMW

New Service Manager Tickets Generate Operations Manager Events

This mode treats Service Manager as an event source or event generator. Through the application of Service Manager business logic, new incident tickets activate logic, which generates event messages that are sent to OMW. OMW Message Source Policies can be configured to treat these events like any other systems or network management events handled at the OMW console.

It allows an OMW-centric model of service desk and enterprise management interaction. If desired, this would allow OMW to track certain Service Manager actions, as if they were SNMP traps, systems administration events, or any other typical OMW managed messages.



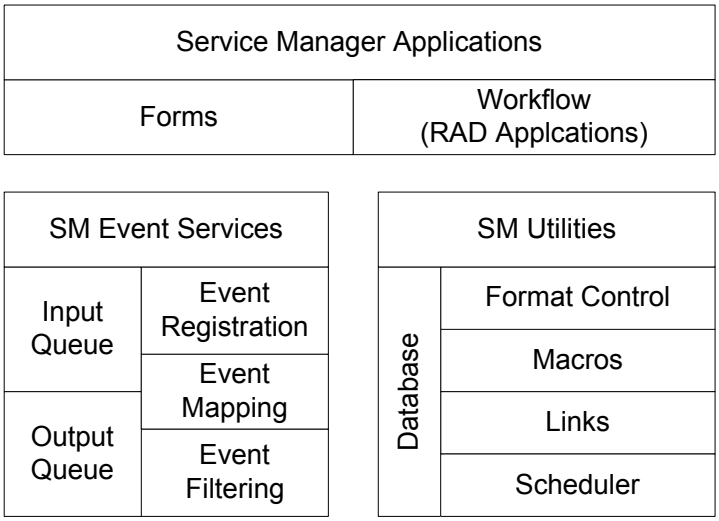
Service Manager Business Logic Topics

Service Manager facilitates and automates the process of supporting business operations. Much like OMW, these benefits can be applied broadly because of Service Manager's flexibility and adaptability to address specific needs. Like OMW, this reflects Service Manager's ability to capture business logic within its applications.

Service Manager provides a significant amount of generic business logic within the service desk applications. The Incident Management application, for example, follows a well-defined help desk model of incident identification, tracking, and resolution. It can be used as is, or it can be customized to address specific organizational requirements. Custom business logic is most often applied to the escalation and notification aspects of the process model, although the process workflow is also frequently altered.

Business logic in Service Manager can be associated with several levels of Service Manager:

- At the bottom level are the Service Manager databases, where policy mechanisms can dictate custom actions.
- Above the database level is a utilities level, which is composed of Schedules, Format Control, Macros, and Links. These tools offer the preferred location for applying custom business logic. All of the tools were designed to support this function. The Service Manager Event Services module essentially connects with Service Manager at the utilities level. This module provides equivalent functions related to processing incoming and outgoing event messages.
- The final level where business logic can be defined is at the Service Manager applications. This level either lends itself to easy customization of the user presentation of application data (via Forms Designer), or it involves more significant alterations that require a programming solution (via Service Manager's fourth generation language, RAD).



SCAuto for OMW utilizes the Service Manager Event Services module for its business logic integration with Service Manager. It leverages default configurations of event messages to open, update, or close an incident ticket. These event message requests are defined as event types of pmo, pmu, or pmc, respectively. The processing logic of these requests can be altered if needed, but the default configuration is satisfactory for integration with OMW.

Comparable to OMW's business logic decision of when events of this type occur, invoke the service desk, the Service Manager Event Services response is when the service desk is invoked, receive and process the data, and then launch the appropriate service desk function. This logic is often combined with Format Control utility logic, which injects programmed queries, tests, and automated actions into the application layer (and therefore into process model of the service desk).

This configuration of utility level business logic offers the Service Manager resident component of synthesizing the service desk with the enterprise systems management console. When combined with the OMW portion, the result is an extremely powerful integration that facilitates more effective and efficient IT operations and the operations of the business as a whole.

SCAuto for OMW Business Logic Topics

The previous two sections discussed capturing your business requirements in the configuration of OMW or Service Manager. As you define how your consolidated service desk functions, you will undoubtedly use the topics of the last two sections. However, when deploying such an enterprise solution, you may also need to put business logic into the middleware represented by SCAuto for OMW.

Other sections of this document discuss the features of this product that allow it to be customized and configured. Any of these features are sufficient for beginning to program business logic into the product. Realistically, any custom logic will be coded into the TCL scripts used for the data maps. Business logic captured in the map scripts will be able to affect many aspects of the overall integration: unique data items, selection of particular event messages, or creating many events from one input event. Regardless, refer to the earlier sections for more details on SCAuto for OMW's abilities of capturing business logic.

In today's business world, companies rely upon their IT resources. HP Operations Manager and Service Manager deliver the IT support that companies count on. The SCAuto for OMW product integrates these major applications, and creates a synthesis of real-time management and process oriented service delivery. Because of the natural dynamic nature of IT, there are constantly new challenges to be addressed in IT management. This new product helps address those challenges directly, and enables enterprise solutions built from the mature concepts of the consolidated service desk.

Architecture

SCAuto for OMW is a collection of tools or components that is highly customizable for integrating event messages from OMW and Service Manager.

Event Integration

Event Message integration provides these capabilities:

- Open Service Manager incident tickets based on OMW new event messages based on event registration conditions.
- Update Service Manager incident tickets based on OMW updated event messages based on event registration conditions.
- Close Service Manager incident tickets based on OMW updated event messages based on event registration conditions.
- Create new OMW event messages from Service Manager eventout output events.
- Modify existing OMW event messages based on modification being done to related incident tickets in Service Manager using the eventout output events.
- Correlate Service Manager incident tickets to one or more OMW event messages.
- Correlate OMW event messages to a Service Manager incident ticket.

Product Flow

When the OMW server gets a message, it validates the message against the conditions in the policy you have loaded. If the conditions specified in the policy are met, SCfromOMW is triggered.

When the SCfromOMW integration component writes an event string to the **`to`** queue file, scevmon picks it up, marks it as processed and forwards it to Service Manager's eventin table.

When an event is created in the eventout table in Service Manager, scevmon picks it up, updates the sync file to point to the last entry in the eventout table, and writes the event string into the **`from`** queue file.

At this point, the SCtoOMW integration component picks it up, marks it as processed, and forwards it through the TCL mapping scripts to OMW.

Integration Components

The following table shows the integration components for SCAuto.

Table 1 Integration Components

File Name	File Type	File Description
scevmon	executable to manipulate queue files to and from Service Manager.	The core technology of the SCAuto product that facilitates bi-directional queueing of messages to and from Service Manager.
SCfromOMW	executable to interface with OMW and write the **`to`** queue from OMW Message Events	Uses event maps and generates the appropriate Service Manager events to the **`to`** queue. It will only notify about changes done to existing events in OMW.
SCtoOMW	executable to read the **`from`** queue file and forward to OMW	Takes an event forwarded from Service Manager eventout and generates or modifies OMW event messages using a series of customizable TCL8.0 scripts.
map files and scripts	configuration text files	VB scripts that are used to get OMW data from the WMI database, and to execute WMI methods to update OMW event messages. TCL scripts that are customizable for creating and modifying OMW events from Service Manager events as well as from OMW to Service Manager. Map files that are used for positionally defining the data being sent to and retrieved from Service Manager.

scevmon

This is an event message processor using **to** (Service Manager) and **from** (Service Manager) queue files to cache events generated by the other components. The basis of this design is to facilitate a fault recovery system that enables OMW event messages to be queued up even while Service Manager is currently not available, as well as queueing up Service Manager events while waiting for OMW to be available. The contents of the queue files include the actual Service Manager events in "^" delimited format preceded by a header. These files should not be modified by hand.

The scevmon component typically runs in the background, makes a TCP connection to the SCAuto server, and waits for events to appear in the queue files.

The config file used by scevmon is `<install dir>\scito.ini`.

The scevmon component reads these files:

- `<install dir>\scevents.to.<host>.<port>` - contains Service Manager eventin events created by SCfromOMW programs. This file is updated to show processed events.
- `<install dir>\syncfile.<host>.<port>` - syncfile for the **from** queue with Service Manager eventout table.

The scevmon component writes these files:

- `<install dir>\scito.log` - log file for informational and error messages.
- `<install dir>\scevents.from.<host>.<port>` - contains Service Manager eventout events to be passed on to OMW. This file is read in by the SCtoOMW program.
- `<install dir>\syncfile.<host>.<port>` - syncfile for the **from** queue with Service Manager eventout table.

SCfromOMW

The SCfromOMW process writes to the **to** queue when registered OMW events appear. You can configure this process in the following file. `<install dir>\EventMap\ToSC\event.ini`.

After you install the policies that have been loaded into OMW, you can tailor these policies to open, update, or close tickets within Service Manager.

When a policy runs, it launches the `SCfromOMW.vbs` script. This script gathers information from OMWs WMI database to populate Environment variables (used by the TCL mapping scripts.) The VB script then launches `SCfromOMW.exe` to process the event and map the OMW event into a Service Manager event. Then it writes the event to a queue file.

For more information, see Configuration on page 43.

SCtoOMW

The SCtoOMW background process monitors the **`from`** queue, and reads in any new events from the Service Manager eventout table. It processes the event by passing it to a configurable TCL script and marks the event as processed.

SCtoOMW is customized by
`<install dir>\EventMap\FromSC\event.ini`.
It runs the TCL scripts pointed to by the event type names in the [EVENT] section, and calls the `SCtoOMW.vbs` script to update the associated OMW event.

For more information, see Configuration on page 43.

3 Installation

This chapter gives the requirements for installing HP SCAuto for Operations Manager for Windows (OMW) and provides step by step installation instructions.

It covers these main topics:

System Requirements

To use SCAuto for OMW, your installation must be on a supported Windows platform and must include the following:

- ServiceCenter version 6.1 or later
- Service Manager version 7 or later
- OMW version 8 or later

The Service Manager server can run on Windows or UNIX.

OMW should be implemented and operational to the extent that meaningful events arrive at the Message browser, and at least one designated individual is familiar with the OMW product as well as defining and editing OMW policies.

By default, SCAuto for OMW is designed to work with the default Service Manager configurations. If Service Manager has been tailored, you may also have to tailor SCAuto for OMW.

Installation Requirements

To install SCAuto for OMW, you must have administrator-level access both to the OMW server and the Service Manager server. Installation takes only a few minutes if you have determined the correct host name and port number values beforehand.

Before you begin installation, you need the following information about your OMW and Service Manager installations:

- Administrator authority at the OMW server
- Administrator-level access to the Service Manager server
- An authorization code for ServiceCenter (SCAuto for OMW is authorized by default in Service Manager.)
- The host name or IP address of the Service Manager server and the TCP port number for use by the SCAuto Base Server.

Installing SCAuto for OMW

Follow these steps to install the SCAuto for OMW:

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

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

   a. Navigate to the installation media folder.

   b. Double-click `setupwin32.exe`.

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

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

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

     The default installation folder is:
     `C:\Program Files\HP\HP SCAuto for OMW 1.5.0`

     Do not install the applications over existing versions of SCAuto. You must uninstall other versions of SCAuto first before installing this one.

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

     You must install SCAuto for OMW in a folder containing only ASCII characters in the folder name.

   The Connection Information screen opens.

6 Fill in the following fields

Table 2 Connection Information

Field	Default Value
ServiceCenter/Service Manager Server Name	localhost
ServiceCenter/Service Manager Server Port	12670

7 Click **Next** to add the folder where OMW is installed.

Do one of the following:

- Click **Next** to accept the default installation folder.
- Click **Browse** to choose a different installation location.

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

The installation information page opens.

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

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

A summary page opens when the installation is complete.

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

11 Reboot the computer to make the configuration of SCAuto for OMW to function.

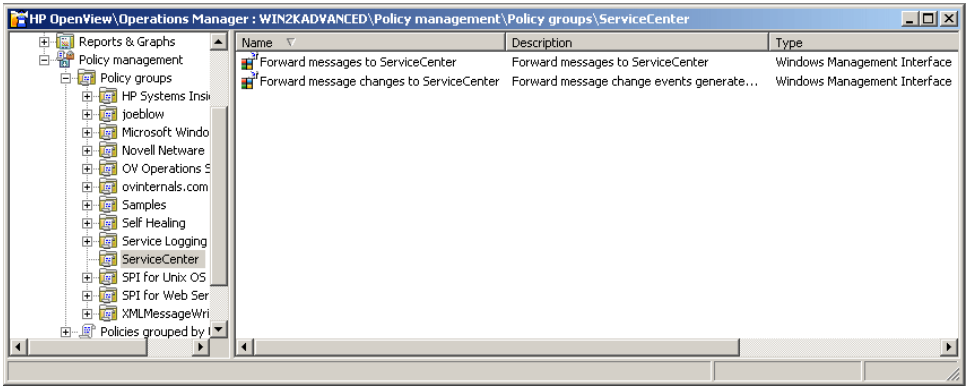
You for integration with Service Manager, you will need to deploy the Service Manager policies after installation. The following section describes how to deploy the out-of-box polices to enable automatic event message integration with Service Manager.

Deploying the Out-of-Box Policies

Follow these steps to deploy the out-of-box Service Manager Policies:

1 Open the OMW console.

2 Expand the **Policy Groups** folder.



3 Select **ServiceCenter**.

The policies should be displayed in the right-hand pane. If there are no policies available, you can upload them. To upload SCAuto for OMW policies, run the OMW tool `ovpmutil.exe` from the command line, as follows:

- For OMW 8 on a 32-bit Windows platform: **`"C:\Program Files\HP\HP BTO Software\bin\ovpmutil.exe" cfg pol upl "C:\Program Files\HP\HP SCAuto for OMW 1.5.0\policies\config.mm"`**
- For OMW 9 on a 64-bit Windows platform: **`"C:\Program Files\HP\HP BTO Software\bin\win64\ovpmutil.exe" cfg pol upl "C:\Program Files (x86)\HP\HP SCAuto for OMW 1.5.0\policies\config.mm"`**

The quotes for the paths are required.

4 Right-Click **ServiceCenter.**

A pop-up menu opens.

5 Select **All Tasks > Deploy on...** from the menu.

The Deploy Policies on dialog box opens.

6 Select the Management Server node only.

7 Click **OK**.

The polices are now deployed on the Management Server, and SCAuto for OMW will now create a new incident in Service Manager for every OMW message with a severity of warning or higher (>=4). You can change this configuration to suit your needs. (For more information on severity in OMW messages, see Table 9 on page 62.)

To further tailor the system to your business needs, refer to Chapter 4, Configuration.

Starting and Stopping SCAuto for OMW

You can start and stop SCAuto for OMW from Windows Services.

Use the following steps to start or stop SCAuto for OMW:

1 From the Windows Start Menu click **Start > Control Panel > Administrative Tools**.

2 Double-click **Services**.

The Services dialog box opens.

3 Double-click **HP SCAuto for OMW**.

The HP SCAuto for OMW propeties window opens.

4 Select the Startup Type from the drop-down menu.

5 Click **Start** to start the program or click **Stop** to stop it.

6 Click **OK** to confirm your selections and exit.

Uninstallation Procedure

Follow these steps to uninstall SCAuto for OMW:

1. Stop SCAuto for OMW if it is running.
2. From the Windows main menu, click **Start** > **Settings** > **Control Panel** > **Add/Remove Programs**.

   The Add/Remove Programs dialog box opens.
3. Select SCAuto for OMW and click **Change/Remove**.

   The SCAuto for OMW installation program opens.
4. Click **Next**.
5. Mark the checkboxes to indicate the features you would like to remove and
6. Click **Next**.
7. Review the uninstallation information.
8. Click **Uninstall** to remove the indicated features.

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

   A summary page opens when the installation is complete.
9. Click **Finish** to exit the Setup wizard.
10. Click **Close** to exit the Control Panel.

4 Configuration

This chapter describes how to configure HP SCAuto for Operations Manager for Windows (OMW).

It covers these main topics:

Refer to the Service Manager Event Services online help for Service Manager event configuration information.

OMW Configuration

Requirements Analysis

To take full advantage of OMW and Service Manager integration, you need a high-level plan for application integration. While the integration can automatically create incident tickets from any events managed by OMW, a defined set of application requirements enables you to focus this general capability into specific measurable results.

For example, you can choose to have all kernel alarms open incident tickets automatically, but this process will create hundreds of incident tickets. If the design is reviewed from a requirements standpoint, where the service desk is pursuing operations in accordance with a service level management (SLM) agreement, perhaps it can be refined to open tickets only on critical events from a small set of critical hosts. In this case, defining the requirements helps meet the specific goals of the SLM agreement by notifying the service desk of potentially huge impacts on meeting the contract.

The process of defining requirements precedes the configuration of any software. It consists of analyzing your business and application and service desk requirements, and organizing these requirements to suit your needs. It is the first step in the Analyze, Design, and Implement process described in this section.

Design

After you have defined the requirements of your consolidated service desk environment, you can determine the specific monitored sources help needed to meet these requirements. This is the Design step of the process. In OMW, this amounts to identifying what Message Sources are relevant. This may be done for individual Message Sources or for groups. Beyond noting which Message Sources are relevant, the next step is to define the parameterization and configuration of each relevant Message Source.

Customization

The out-of-box OMW Policies contain rules that describe the conditions that trigger OMW actions. You can change these rules by editing the policies. The tcl scripts map OMW message attributes to Service Manager event variables and create ServiceCenter incidents and OMW messages using these mappings. You can change these specifications by editing the tcl scripts.

Example 1: Editing an OMW policy

You can specify when to create Service Manager incidents by modifying the rules in the "Forward messages to ServiceCenter" policy, located in the ServiceCenter policy group folder.

Out-of-box the policy for forwarding messages to Service Manager acts on any new OV_Message where TargetInstance.Application is not set to "ServiceCenter" and TargetInstance.Severity is greater or equal to "4" (Warning or greater).

Here is the condition statement from the out-of-box rule:

```
IF Condition (TargetInstance.Application != ServiceCenter [Value]
                & {TargetInstance.Severity >= 4 [Value]})
```

If you want to only open tickets for Critical messages from Insight Manager you would change the condition in the Policy's rule to:

```
IF Condition (TargetInstance.Application == Insight Manager [Value]
                & {TargetInstance.Severity == 32 [Value]})
```

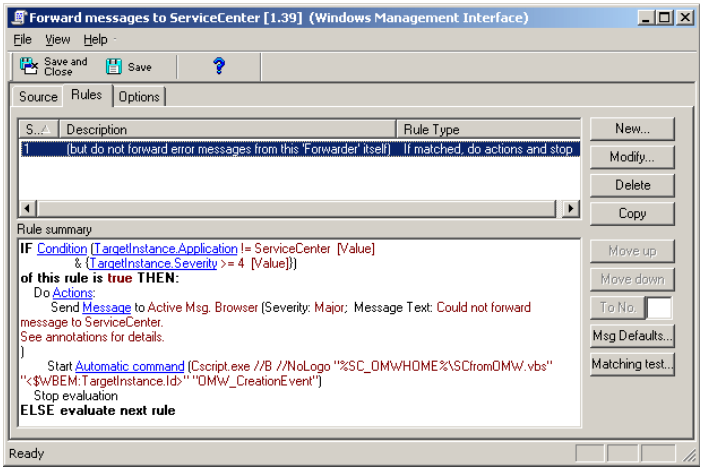
Follow these steps to have incidents created only for Critical messages from Insight Manager:

1. From the OMW console, expand the **Policy Groups** folder.
2. Select **ServiceCenter.**

   The policies are displayed in the right pane.
3. Right-click the **Forward messages to ServiceCenter** policy.
4. From the menu, select **All Tasks > Edit...**

   The Policy dialog box opens.

5 Select the **Rules** tab.



6 Click **Modify.**

7 Select **TargetInstance.Severity** and click **Modify.**

8 In the Specific Value to Compare field, change the value “4” to **`32`**.

9 Select **TargetInstance.Application** and click **Modify.**

10 In the Specific Value to Compare field, change the value “ServiceCenter” to **`Insight Manager`**.

11 Click **OK.**

12 Click **OK**.

13 Click **Save and Close**.

14 Deploy the modified policy.

Right-click the **Forward messages to ServiceCenter policy** and click **OK** to deploy the policy on the management server.

SCAuto for OMW will now create a new incident in Service Manager only for Critical messages from Insight Manager.

Example 2: Editing a tcl Script

By default, the out-of-box tcl scripts for processing open, update, and close events (pmo, pmu, pmc) from Service Manager annotate the original message and send an informational message to the console indicating that a ticket has been opened, updated or closed. This is can be useful when you are first working with the product, to be sure that you are creating the appropriate messages. However, at some point you may want to turn those messages off.

You can suppress these informational messages by commenting out the block of code that calls "opcif_write" from the appropriate tcl script by inserting a "#" character at the beginning of the line.

For example, the following is from the `<install dir>\EventMap\FromSC\pmo.tcl` file:

```
# lets make a new ServiceCenter message.
#
if [catch {opcif_write "$OPC_Message_Text" "$OPC_Application"
"$OPC_Message_Group" "$OPC_Message_Type" "$OPC_Node_Name"
"$OPC_Object" "2"} rc] {
  tcl_logprint "pmo.tcl: Unable to create OMW message for pmo.
$SC_TICKET"
  tcl_logprint "pmo.tcl: $rc"
} else {
  tcl_logprint "pmo.tcl: OMW message created for pmo. $SC_TICKET"
}
```

To suppress the informational message you would change this block of code to:

```
# lets make a new ServiceCenter message.
#
#if [catch {opcif_write "$OPC_Message_Text""$OPC_Application"
#"$OPC_Message_Group" "$OPC_Message_Type" "$OPC_Node_Name"
#"$OPC_Object" "2"} rc] {
#  tcl_logprint "pmo.tcl: Unable to create OMW message for pmo.
#$SC_TICKET"
#  tcl_logprint "pmo.tcl: $rc"
#  else {
#  tcl_logprint "pmo.tcl: OMW message created for pmo.
#$SC_TICKET"
#}
```

SCAuto for OMW no longer sends the informational messages.

Using scito.ini Parameters

You can set the following startup parameters in the `scito.ini` file:

Table 3 scito.ini parameters

Parameter	Description
sessid	By default, this is set to OPCENTER. This is the license string for the SCAuto for OMW product. Do not modify.
scauto	This is the **`hostname.port`** of the SCAuto server for SCAuto for OMW. This parameter is configured by the installation script.
event_map_dir	This is the root folder name (starting from the installation folder) of the event mapping scripts and files. By default, it is configured as “EventMap”. It's not suggested to modify this parameter. Besides to modify the root folder name, all of the other places referring to this parameter also need to be modified.
log	This is the file name for the log file. By default, it is configured as `scito.log`.
debugSCfromOMW	Set =1 turns on tracing for the SCfromOMW process. Set =2 turns on more verbose tracing for the SCfromOMW process
debugSCtoOMW	Set=1 turns on tracing for the SCtoOMW process. Set=2 turns on more verbose tracing for the SCtoOMW process.
debugscevmon	Set=1 turns on tracing for the scevmon process. Set=2 turns on more verbose tracing for the scevmon process.
eventlogmaxlen	This parameter defines the maximum size, in bytes, that the event queue files can reach before being purged of processed events. The default is 5M. The minimum size is 1M.
logmaxlen	This parameter defines the maximum size in bytes that the log file can reach before it will be wrapped down. The default is 5M. It is not recommended to set this value to less than 1M.
usersepchar	Allows user to define a separator character other than “^” (^ is the default value). To specify a different character, enter a decimal value of an ASCII character between 1 and 127.

Table 3 scito.ini parameters

Parameter	Description
scevents	Specifies an optional list of event types which are to be retrieved from Service Manager's EVENTOUT queue. The default is to retrieve all types, however, this can be inefficient, because it can result in bringing over certain types of events, such as outbound page messages or E-mail messages, which have no meaning to OMW. To restrict the types of events that are retrieved, code them in a list separated by commas and enclosed in parentheses. For example, **`scevents:(pmo,pmu,pmc)`**. To disable all retrieval of outbound events from Service Manager, use the "**`noeventsfromsc:`**" option.
noeventsfromsc	Specifies whether or not scevmon should retrieve events from Service Manager's EVENTOUT queue. The default is 0 (which means events will be retrieved). To disable retrieval of all outbound events from Service Manager, code "**`noeventsfromsc:1`**".
scevusers	This parameter restricts the events which are to be retrieved from Service Manager's EVENTOUT queue based on the value in evuser field. The default is to retrieve all events regardless of the value of the evuser field, however, this can be inefficient, because it can result in bringing over certain events which are not intended for OMW. To use this parameter, code the values in a list separated by commas and enclosed in parentheses.For example, **`scevusers:falcon`** only get events that have falcon as the evuser. **`scevusers:(falcon,SCITO)`** only get events that have falcon or SCITO as the evuser.
scevmon_sleep_interval	Specifies the amount of time in seconds that scevmon should sleep. The default is 5 seconds. For example, **`scevmon_sleep_interval:1`** tells scevmon to sleep for one second.

TCL Event Mapping from OMW to Service Manager

This section discusses TCL language event mapping from OMW to Service Manager. This scripted approach to configuring the SCAuto for OMW product results in a flexible method of configuring or customizing it to meet specific business goals and requirements.

The mapping mechanism to map OMW message events into Service Manager incident tickets is made up of an active facility, which is a TCL script, acting on variables populated with OMW message attributes to create a Service Manager TCL event object, using a passive facility, a static map file positionally defining each kind of interested Service Manager event. The send command of this Service Manager TCL object is then invoked to queue the event to be sent to Service Manager.

Configuration Overview

By default, the mapping files reside in the EventMap folder in the product installation folder. The hierarchy and explanation of these files follows:

- `EventMap`
- `EventMap\ToSC\` -- files to configure events from OMW to Service Manager

The following files are located in the `EventMap\ToSC\` folder:

Table 4 Mapping files

File	Description
event.ini	Defines the name of the TCL script to run.
eventmapSCfromOMW.tcl	The event map script for SCfromOMW.
pmo.map, pmu.map, and pmc.map	Static map file templates used in the scripts to positionally define Service Manager event elements in the Service Manager TCL event object.
util.tcl	A utility script that contains commands to convert OMW severity values to Service Manager severity codes.

The `event.ini` file configures which TCL scripts to invoke when there is an OMW message. The TCL scripts will use the static map files as a template to create Service Manager TCL event objects, populate its members, and finally send it off to the queue file to be sent to Service Manager eventin table.

OMW Variables

These variables are set in the `SCfromOMW.vbs` VB script by retrieving the values in the message with the Window Management Instrumentation (WMI).

These variables are made available as TCL_GLOBAL and will be accessible via the $VARIABLE syntax globally from within the called TCL script.

Table 5 OMW Variables

Variable Name	Short Definition
OPCDATA_MSGID	The unique OMW message ID.
OPCDATA_NODENAME	The name of the node producing the message. The message is only handled by the OMW manager if this system is part of the OMW Node Bank.
OPCDATA_CREATION_TIME	The time the message was created in seconds.
OPCDATA_RECEIVE_TIME	The time the message was received by the management server in seconds.
OPCDATA_MSGTYPE	Message type. This attribute is used to group messages into subgroups, e.g., to denote the occurrence of a specific incident. This information may be used by event correlation engines.
OPCDATA_GROUP	Message group.
OPCDATA_OBJECT	Object name to use for the OMW message.
OPCDATA_APPLICATION	Application which produced the message.

Table 5 OMW Variables

Variable Name	Short Definition
OPCDATA_SEVERITY	Severity of the message. Possible values are: 0 - Unchanged or none 1 - Unknown 2 - Normal 4 - Warning 8 - Minor 16 - Major 32 - Critical
OPCDATA_AACTION_NODE	Defines the node on which the automatic action should run.
OPCDATA_AACTION_CALL	Command to use as automatic action for the OMW message.
OPCDATA_AACTION_ ANNOTATE	Defines whether OMW creates start and end annotations for the automatic action. 0 - do not create annotations 1 - create annotations
OPCDATA_AACTION_ACK	Auto Acknowledge after successful execution of the Automatic Action. 0 - do not auto-acknowledge 1 - auto-acknowledge
OPCDATA_OPACTION_ NODE	Defines the node on which the operator initiated action should run.
OPCDATA_OPACTION_CALL	Command to use as operator-initiated action for the OMW message.
OPCDATA_OPACTION_ ANNOTATE	Define whether OMW creates start and end annotations for the operator initiated action. 0 - do not create annotations 1 - create annotations

Table 5 OMW Variables

Variable Name	Short Definition
OPCDATA_OPACTION_ACK	Auto Acknowledge after successful execution of the operator initiated action. 0 - do not auto-acknowledge 1 - auto-acknowledge
OPCDATA_MSG_LOG_ONLY	Message is Server Log Only.
OPCDATA_UNMATCHED	Defines whether or not the message matches a condition. 0 - the message was sent to the server because it matched a match condition 1 - the message did not match a match condition of the assigned templates, but was forwarded nevertheless
OPCDATA_TROUBLETICKET	Forward message to Trouble Ticket System.
OPCDATA_TROUBLETICKET_ACK	Acknowledge message after forwarding it to the Trouble Ticket System
OPCDATA_NOTIFICATION	Notification
OPCDATA_INSTR_IF_TYPE	Instruction type
OPCDATA_INSTR_IF	The instruction text
OPCDATA_INSTR_PAR	Instruction parameters.
OPCDATA_MSGSRC	Message Source.
OPCDATA_MSGTEXT	Message Text.
OPCDATA_ORIGMSGTEXT	Original Message Text.
OPCDATA_ANNOTATIONS	This is not a OMW data element. It is created by the adapter product to contain a text string of all the annotations of the message.
OPCDATA_LAST_ANNOTATION	This is not a OMW data element. It is created by the adapter product to contain a text string of the last annotation of the message.

There are also three non-OPC variables added for programmability.

Table 6 Non-OPC variables

Variable name	Short definition
INSTALLDIR	The home folder where the product was installed.
SC_MSGTYPE	A converted message type string denoting which type of message it is. Valid values are: New message Message Owned by a user Message Disowned by a user Message Acknowledged Message has new annotation(s)
SC_PROBLEM_NUMBER	The Service Manager incident number embedded in the OMW Message annotation text when its a new message, or the Object of message Group Service Manager when its a Service Manager created message.

Service Manager TCL Event Object

Legend

<> required parameter

[] optional parameter

Summary of Service Manager TCL Commands

```
create_sc_event <object name>
<object name> set_evtype <event type>
    [ use_template <mapfile name> ]
<object name> set_evfield <0...80>|<field name> <field value>
<object name> print
<object name> send
tcl_logprint
```

create_sc_event

```
create_sc_event <ObjectName>
```

The resultant object from this command will have methods to populate its data members and a send method to queue the event to Service Manager. The methods are invoked by using the newly created object name itself and therefore will have its own context. There is a limit of 10 objects that can be created in each script.

```
create_sc_event eventObject
```

The object named eventObject is created and can be used to set values and finally send it to Service Manager.

set_evtype

```
<ObjectName> set_evtype <evtype name>
   [use_template <mapfile name>]
```

This command sets the evtype field of the event to <evtype name> (e.g., pmo, pmu pmc etc.). There are two optional parameters that can be specified for the event to use a Static Map File for setting the event values:

- use_template
- <mapfile name>

If these optional parameters are not specified, setting of object fields can still be done using integer indexes.

Examples

```
create_sc_event eventObject
# create the event object
eventObject set_evtype pmo use_template
   "EventMap/ToSC/pmo.map"
# make it of type pmo, and use the template for
# named indexes.
```

The event object eventObject is created. It is then set to type pmo, essentially, the evtype field of the event is set to pmo. The static map file `EventMap\ToSC\pmo.map` is used to define the named indexes that will be used later to set the other fields.

```
create_sc_event eventObject
# create the event object
eventObject set_evtype pmo
# set the evtype to pmo. do not use templates for indexing
```

The event object eventObject is created. It is then set to pmo for evtype. Because a template was not used here, subsequent setting of evfields will have to use integer indexes.

set_evfield

```
<ObjectName> set_evfield <0 ... 80>|<evfield name> <value>
```

This command sets the event fields with values. If the use_template option was used during the set_evtype command, then you may use field names defined in the static map file templates to set the values, otherwise, you have to use positional integer indexes to set the evfield values.

Examples

```
create_sc_event eventObject
# create the event object
eventObject set_evtype pmo
# set the evtype to "pmo". do not use templates for indexing
eventObject set_evfield 1 $OPCDATA_NODENAME
# set evfield position 1 to OMW nodename variable
```

The event object is created, set to evtype “pmo” without using a static map file for a template. The first field of the event object is set to the OMW variable for nodename.

```
create_sc_event eventObject
# create the event object
eventObject set_evtype pmo use_template
   "EventMap/ToSC/pmo.map"
# set the evtype
# use a template for indexing
eventObject set_evfield logical.name $OPCDATA_NODENAME
# set logical.name evfield to OMW nodename variable
```

The event object is created, set to evtype “pmo” using a static map file for a template. The logical.name field of the event object is set to the OMW variable for nodename.

set_evuser

```
<ObjectName> set_evuser <value>
```

This command sets the evuser field of the event to <value>. The maximum length of <value> is 24 characters. The default value for the evuser field is SCITO.

Example

```
# create the event object
create_sc_event eventObject

# set evuser to SCITO-WEST
eventObject set_evuser "SCITO-WEST"
```

print

```
<ObjectName> print
```

This command outputs to the log file all the contents of the object including the map field names if used.

send

```
<objectName> send
```

This command queues the event object to be sent to Service Manager's eventin table.

For more examples on how to use the commands, see the example script file.

tcl_logprint

```
tcl_logprint "Message"
```

This command is used to write messages from the tcl script to the SCAuto for OMW log file (`scito.log`).

Example

```
tcl_logprint "eventmapSCfromOMW.tcl: queueing new ticket to
be opened. ($OPCDATA_MSGID)"
```

Static Map File

The static map file that is usable in a set_evtype command as the template is an ascii text file. Each row denotes a positional field name of the intended Service Manager event map starting at index 1, for example, row 1 = field 1 in event map.

By default, three maps for event types pmo, pmu and pmc are delivered. These maps match the Service Manager event maps.

The map file is not a required piece of the configuration; it is used to make it convenient and manageable to name the event fields in the Service Manager TCL event object. Without this file, however, you can still set the values using integer indexes.

Customization

By default, after installing the product, the To Service Manager TCL scripts are configured to generate pmo events with the category of example. Also, the default behavior for the example category in Service Manager is to match incident tickets based on the logical.name field in the pmo event. If this is the desired effect, you do not have to customize the product.

If you want the pmo to go into a specific Service Manager category, you must take the following steps:

- Make sure that the targeted category contains format control logic to handle a pmo the same way as the example category.
- Customize the TCL script to generate the desired category in the resulting event.

TCL Event Mapping from Service Manager to OMW

The mapping mechanism to map Service Manager events into OMW messages is done by invoking a TCL script. Service Manager event fields are made into TCL_GLOBAL variables that can be accessed in the targeted script with the $VARIABLE syntax. Various OMW WMI Methods are also made into TCL_Command commands callable from within the same script, thus effectively bridging the two domains.

Configuration Overview

By default, the mapping files reside in an EventMap folder within the product installation folder. The hierarchy and function of the EventMap folder structure follows:

- `EventMap`
- `EventMap\FromSC\` - files to configure events from Service Manager to OMW.

The following files are located in the EventMap\FromSC\ folder:

Table 7 EventMap files

Filename	Description
event.ini	The configuration file that specifies the TCL scripts to invoke based on event types.
pmo.tcl	The event map script to invoke when there is a pmo (Incident Opened).
pmu.tcl	The event map script to invoke when there is a pmu (Incident Updated).
pmc.tcl	The event map script to invoke when there is a pmc (Incident Closed).

Table 7 EventMap files

Filename	Description
util.tcl	Utility TCL script that contains command to convert Service Manager severity codes to OMW severity values.
default.tcl	A default TCL script to invoke when an event type is received and there is no equivalent TCL script.
showEvent.tcl	A debug script that will display all event field values from a Service Manager event.

The `event.ini` file configures the TCL scripts that are invoked based on the evtype field of the incoming Service Manager event.

Service Manager TCL variables

The available variables depend on the evtype type field and are in accordance with the event maps from Events Services in Service Manager for that type. These variables are made available as TCL_GLOBAL and will be accessible via the $VARIABLE syntax globally from within the called TCL script. For more information, refer to the Event Services section of the Service Manager online help.

Table 8 TCL variables

Variable name	Short description
INSTALLDIR	The folder path where the product was installed. This is not part of an event field and is added for convenience in locating helper scripts.
SCEVTYPE	Service Manager registration name for the event, for example, pmo.
SCEVTIME	Date and time event occurred.
SCEVSYSSEQ	Service Manager eventout queue sequence number. Used as checkpoint in sync file.
SCEVUSRSEQ	A user-assigned sequence number used to trace an event through Service Manager.

Table 8 TCL variables

Variable name	Short description
SCEVSYSOPT	A code to identify system options.
SCEVUSER	The event user name; if passed, it is used as the operator name.
SCEVPSWD	The event user's password.
SCEVSEPCHAR	The character used to separate fields in the $SCEVFIELDS variable. Default is "^".
SCEVFIELDS	The data describing the event, with fields separated by the $SCEVSEPCHAR character.
SCEVFIELDS_ 1...2...3...*	The event fields from the $SCEVFIELDS string parsed out as individual field values using the $SCEVSEPCHAR separator value.

The specific field definition depends on the evtype coming back and is documented in the <evtype>.tcl scripts themselves. It is up to the user to define types that do not have a corresponding TCL script.

OMW WMI Methods as TCL Commands

LEGEND

```
< > required parameter
[ ] optional parameter
```

These are OMW WMI Methods that have been wrapped in TCL commands and made available during the script invocation.

Summary of OMW TCL Commands

- `opcif_write <Message Text> <Application> <Message Group> <Message Type> <Node Name> <Object> <Severity>`
- `opcmsg_annotation_add <Message Id> <Annotation Text>`
- `opcmsg_ack <Message Id>`

- `opcmsg_unack <Message Id>`
- `opcmsg_own <Message Id>`
- `opcmsg_disown <Message Id>`
- `opcmsg_cma_set <Message Id> <CMA Name> <CMA Value>`
- `tcl_logprint`

opcif_write

```
opcif_write <Message Text> <Application> <Message Group>
<Message Type> <Node Name> <Object> <Severity>
```

The opcif_write command creates a new OMW message. To pass a null value, use " " for an empty string instead. There are seven required arguments to this command:

Table 9 Required Arguments

Argument	Description
Message Text	The message text of the newly created OMW message.
Application	The application that this message belongs to.
Message Group	The group that this message belongs to.
Message Type	Currently not used.
Node Name	The OMW node name that this message is created for.
Object	The object that this message belongs to.
Severity	The OMW severity value for this message. Check the utility script EventMap\FromSC\util.tcl for the conversion utility to convert from Service Manager severity codes. Valid values are: 32 - Critical 16 - Major 8 - Minor 4 - Warning 2 - Normal

Example

```
# sourcing this script to get the ConvertSeverity command
source "$INSTALLDIR./EventMap/FromSC/util.tcl"

# Format message text to be passed to opcif_write
set OPC_Message_Text "Incident ticket $SCEVFIELDS_2 opened
for Message ID $SCEVFIELDS_14 on $SCEVFIELDS_4 by
$SCEVFIELDS_5"

# convert Service Manager severity to OMW severity
set SCSEVERITY [ ConvertSeverity "$SCEVFIELDS_8" ]

# create the new OMW message
opcif_write "$OPC_Message_Text" "\"Service Manager_pmo\""
"\"Service Manager\"" "TroubleTicket" "$SCEVFIELDS_18" "pmo"
"$SCSEVERITY"
```

In this example, a new OMW message is created with a formatted message text from Service Manager event fields of Application="Service Manager_pmo", Group="Service Manager",Type="TroubleTicket", node name from Service Manager event logical.name field, Object="pmo" and a converted Severity that is equivalent to the Service Manager severity code.

opcmsg_annotation_add

```
opcmsg_annotation_add <Message Id> <Annotation Text>
```

This command adds an annotation to an existing OMW message. This command requires two arguments, consisting of a OMW message ID and free-form annotation text.

Example

```
# pre-format an annotation text to use from Service Manager
# field values
set OPC_Annotate_Text "Incident ticket $SCEVFIELDS_2 opened
on $SCEVFIELDS_4 by $SCEVFIELDS_5\n   Category:
$SCEVFIELDS_3\n   Assigned to: $SCEVFIELDS_9\n   Severity:
$SCEVFIELDS_8"

# annotate and existing OMW message
opcmsg_annotation_add "$SCEVFIELDS_14" "$OPC_Annotate_Text"
```

The previous example pre-formats annotation text to be used to annotate an existing OMW message identified by the message ID stored in event field index 14 of the evfields string.

opcmsg_ack

```
opcmsg_ack <Message Id>
```

Given the message ID of an existing OMW message, this command will acknowledge it.

opcmsg_unack

```
opcmsg_unack <Message Id>
```

Given the message ID of an existing OMW message, this command will unacknowledge it.

opcmsg_own

```
opcmsg_own <Message Id>
```

Given the message ID of an existing OMW message, this command will own it.

opcmsg_disown

```
opcmsg_disown <Message Id>
```

Given the message ID of an existing OMW message, this command will disown it.

opcmsg_cma_set

```
opcmsg_cma_set <Message Id> <CMA Name> <CMA Value>
parameters:
Message Id The universally unique message id used by OM.
CMA Name The name assigned to the Custom Message Attribute.
CMA Value The value assigned to the Custom Message Attribute.
```

Example

```
if [catch {opcmsg_cma_set "$OMW_MSGID" "Ticket Number"
"$SC_TICKET"} rc]
{
tcl_logprint "pmo.tcl: Unable to set CMA Ticket Number for OMW
message
($OMW_MSGID)"
tcl_logprint "pmo.tcl: $rc"
} else {
tcl_logprint "pmo.tcl: OMW message CMA Ticket Number set to
$SC_TICKET"
}
```

tcl_logprint

```
tcl_logprint “Message”
```

This is used to write messages from the tcl script to the SCAuto for OMW log file (`scito.log`).

Example

```
tcl_logprint "pmo.tcl: OMW message created for pmo.
$SCEVFIELDS_2"
```

Event Configuration File

The `event.ini` file in the `<install dir>\EventMap\FromSC` folder is a means of configuring the SCtoOMW component of the interface. In here, you may specify the TCL map script to invoke when a certain event type is read from Service Manager.

Sections

Section header names are identified by square brackets "[....]". The brackets group sets of configurable parameters.

EVENT - Section for configuring event types to the TCL map script to execute.

Each section header name has its own set of required parameters.

Table 10 Required parameters

Section name	Required parameters
EVENT	<event type>=<TCL map script to execute> For example: pmo=EventMap\FromSC\pmo.tcl This will tell the interface to execute the script <install dir>\EventMap\FromSC\pmo.tcl when a "pmo" event is received from Service Manager.

Default event.ini file

```
---------------- begin file ----------------
[ EVENT ]
pmo=EventMap\FromSC\pmo.tcl
pmu=EventMap\FromSC\pmu.tcl
pmc=EventMap\FromSC\pmc.tcl
---------------- end file ----------------
```

Default Behavior

When there is a pmo in the queue file (such as when a ticket is newly created), SCtoOMW annotates the originating event. SCtoOMW also creates a new OMW event opened event with the IM number being the Object.

When there is a pmu in the queue file (such as when its created by a pmu in eventin, or when a Service Manager operator modifies the ticket), SCtoOMW annotates the originating event and creates a new OMW message with the IM number being the Object.

Similarly, when the queue file has a pmc, the originating event will be annotated and a new OMW message will be generated.

5 Maintenance

This chapter provides basic instructions on how to maintain HP SCAuto for Operations Manager for Windows (OMW).

It covers these main topics:

Basic Maintenance

In the product installation folder, the SCAuto for OMW adapter contains a log file called `scito.log`, where all informational and error messages from the product are stored, as well as a parameter configuration file called `scito.ini`

The product requires very little maintenance once installed and running. Once the log file reaches 5M bytes, it purges the old information. In addition, the event queues purge processed events to remove them once they reach 5M bytes each, retaining the unprocessed events.

Troubleshooting

To Service Manager

If incident tickets in Service Manager are not being created by OMW event messages, follow these steps to troubleshoot the problem:

1. Verify that the event monitor process, scevmon, is running.
2. Verify that the event monitor process scevmon is communicating with the SCAuto server. If the `scito.log` file contains the message

   `... scevmon: unable to connect..., do the following:`

   - Check the `scito.ini` file for the parameter SCAuto:. Verify that it exists and points to the **`<hostname>.<port number>`** of the SCAuto server you are trying to connect to.
   - Log in to the Service Manager server host and verify that the SCAuto server is running.
   - Check the network by pinging the Service Manager Server host to see if it is reachable.
3. Verify that the SCfromOMW process is running.
4. Verify that OMW event message sources are correctly set up to communicate with SCAuto for OMW. (See Chapter 4, Configuration for more details on configuring OMW message sources.)
5. Verify that the `event.ini` configuration file in the `EventMap\ToSC` folder is properly configured.
6. Check the `scito.log` file for any unusual error messages.

   If you cannot resolve the problem, you will need to give this information to Customer Support, along with the product versions you are using. For more information, see Determining the Current Product Version on page 74.

To OMW

Service Manager incident ticket modifications are not reaching the OMW message browser

Follow these steps to troubleshoot the problem:

1. Verify that a record is generated in the eventout table in Service Manager. Configure format control to output equivalent records accordingly.
2. Verify that the event monitor process, scevmon, is running.
3. Verify that the event monitor process scevmon is communicating with the SCAuto server. If the `scito.log` file contains the message

   `... scevmon: unable to connect...,` do the following:

   - Check the `scito.ini` file for the parameter SCAuto:. Verify that it exists and points to the **`<hostname>.<port number>`** of the SCAuto server you are trying to connect to.
   - Log in to the Service Manager server host and verify that the SCAuto server is running.
   - Check the network by pinging the Service Manager Server host to see if it is reachable.
4. Verify that the SCtoOMW process is running.
5. Verify that the `event.ini` configuration file in the `EventMap\FromSC` folder is properly configured.
6. Check the `scito.log` file for any unusual error messages.

   If you cannot resolve the problem, you will need to give this information to Customer Support, along with the product versions you are using. For more information, see Determining the Current Product Version on page 74.

Errors occur when an event record field contains the character "^"

By default, the character "^" is used as the separator of event record fields. If the content of a field contains this character, errors will occur when this event record is processed in Service Manager.

To solve this problem:

1 Use the `usersepchar` parameter to specify a different separator character in the `scito.ini` configuration file.

Use the decimal value of an ASCII character between 1 and 127. For example, to use "~" as the separator character, add a line like "usersepchar:126" to `scito.ini`, as the decimal value of "~" is 126.

2 Make sure that this new separator character does not appear in any event fields.

Determining the Current Product Version

Before contacting customer support, you will also need to know the current version of your Service Manager and SCAuto for OMW products.

You can determine the current version of Service Manager by using the Service Manager client About menu.

You can determine the current version of SCAuto for OMW by using the command line argument "**-v**" when running SCfromOMW, SCtoOMW, and scevmon as in the following example.

```
SCfromOMW -v
```

Index

P

R

S

T

U

W