ServiceCenter®

Version 1

SCAuto for Unicenter TNG

February, 1999

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

This edition applies to version 1 of the licensed program

ServiceCenter®

February, 1999

Table of Contents

Chapter 3 Customizing SCAuto for Unicenter TNG

Preface

Contacting Peregrine Systems

Contact Peregrine Customer Support if you run into any problems.

North and South America Support

For ServiceCenter problems or information that is needed immediately, call Peregrine Customer Support at:

(800) 960-9998 *- or -* **(619) 794-7402**

For ServiceCenter questions or information, send a FAX or email.

FAX: **(619) 794-6028**
Internet email: **support@peregrine.com**.

Send materials that Peregrine Customer Support requests to:

Peregrine Systems, Inc.
attn: Customer Support
3611 Valley Centre Drive
San Diego, CA 92130

Hours: 5:00 a.m. to 5:30 p.m. PST

Europe - Contact the Office Nearest You

Great Britain

Peregrine Systems Ltd.
1st Floor
Ambassador House
Paradide Road
Richmond, Surrey, Great Britain, TW9 1SQ

Telephone: **+44 (0) 181-332-9666**
Fax: **+ 44 (0) 181-334 5890**
Email address: **uksupport@peregrine.com**
Hours: 08:00h to 18:00h GMT, Monday - Friday

France

Peregrine Systems
Tour Franklin-La Défense 8
92042 Paris La Défense Cedex

Telephone: **+331 47 73 11 11**
Fax: **+331 47 73 11 12**
Hours: 08:00h to 17:00h, Monday - Friday

Germany

Peregrine Systems GmbH
Burohaus Atricom
Lyoner Strasse 15,
60528 Frankfurt, Germany

Phone: **+49-(0)69-66-80-260**
Fax: **+49-(0)69-66-80-2626**

Hours: 08:00h to 17:00h, Monday - Friday

Denmark

Peregrine Systems A/S
Naverland 2, 12 SAL
DK-2600 Glostrup
Denmark

Phone: **+45-4-346-7676**
Fax: **+45-4-346-7677**
Hours: 08:30h to 17:00h, Monday - Friday

Holland / Netherlands / Benelux

Peregrine Systems BV
Botnische Golf 9a
3446 CN Woerden
the Netherlands

Phone: **+31-348-437070**
Fax: **+31-348-437080**
Hours: 08:30h to 17:30h, Monday - Friday

Asia-Pacific

Internet email: **support@peregrine.com**. or call

U.S. telephone: **619-794-7402**

Note: Only the European Customer Support staff is multi-lingual and can provide technical support to customers in their native language, i.e., English, French, German, Italian, Spanish, etc.

Chapter 1 Introduction

SCAuto for Unicenter TNG

SCAuto for Unicenter TNG provides a bi-directional, near-real-time event-driven interface between Unicenter TNG and Peregrine Systems' ServiceCenter®. Operating in a Windows NT environment, SCAuto for Unicenter TNG installs on one or more TNG servers.

SCAuto for Unicenter TNG uses Unicenter message action commands to translate Unicenter Event Console messages to ServiceCenter events. Useful for opening, updating, and closing problem tickets within ServiceCenter, it can also integrate with other ServiceCenter applications, including Inventory and Configuration Management, Request Management, and Change Management.

SCAuto for Unicenter TNG employs the TNG Worldview API to receive notifications about TNG Common Object Repository (CORE) updates, then converts these to ServiceCenter events.

SCAuto for Unicenter TNG is bi-directional, receiving output events from ServiceCenter applications, then executing automation scripts on the TNG server. These scripts can issue Windows NT commands or TNG commands such as **cawto** and **cautil**.

SCAuto for Unicenter TNG also includes a user menu, a popup menu, and method objects, which are added to the 2D map. These allow operators to launch various ServiceCenter applications by selecting objects on the 2D map and using the right mouse button.

Finally, SCAuto for Unicenter TNG has the optional capability of transparently interfacing with the Unicenter Enterprise Problem Management applications, including MGPT.

SCAuto for Unicenter TNG uses the TCP/IP protocol to exchange messages (events) with the Event Services component of ServiceCenter. SCAuto for Unicenter TNG can generate any kind of input event to ServiceCenter, including custom events of your own design. When installed with no special customization, SCAuto for Unicenter TNG generates two types of ServiceCenter events:

- Problem Management Events
- Inventory and Configuration Management Events

The rules for generating ServiceCenter events from a given type of Unicenter event are specified in external text files, which can easily be customized to your site's requirements. This allows you to specify the mapping of TNG event variables or CORE properties to the various ServiceCenter application fields.

System Prerequisites

To use SCAuto for Unicenter TNG, your installation must include the following:

- Unicenter TNG product release 2.0 or later
- Windows NT 4.0 or later
- ServiceCenter 1.4 or later

The ServiceCenter server can be on Windows NT, UNIX, or MVS, but must be using the TCP/IP protocol. SCAuto for Unicenter does not presently support the APPC networking option.

Unicenter TNG should be implemented and operational to the extent that meaningful events arrive at the Event Management console, and at least one designated individual is familiar with the UnicenterTNG product as well as defining TNG Event Message records and TNG Message Actions.

SCAuto for Unicenter TNG supports ServiceCenter releases 1.4, 2.0, and 2.1; however you will achieve best results if the test ServiceCenter installation to be used to evaluate SCAuto for Unicenter TNG is a freshly installed one, with unmodified applications as distributed with release 2.0 SP 1 or later, or 1.4 SP 7 or later.

If you have an older ServiceCenter installation, and/or have done extensive tailoring and customization, you ***must*** first perform RAD application maintenance on your system. This is distributed on the *SCAutomate for ServiceCenter 1.4* and *SCAutomate for ServiceCenter 2.0* CDs, in the **\UNLOAD** directory, along with a README file describing the required updates.

If your ServiceCenter system has been customized in significant ways, for example, if you intend to add a variety of new categories, special forms, and extra validation of data entered by help desk operators, you should carefully review the material in the publication entitled **Event Services Utility**.

Problem ticket events generated by automation products such as SCAuto for Unicenter TNG may require new categories to be established, with different validation rules. In particular, interactive script processing must not be attempted for automated trouble tickets.

Customers with heavily customized or older systems should give consideration to use of Peregrine Systems Professional Services to implement SCAutomate products.

Installation Prerequisites

Basic installation of SCAuto for Unicenter TNG is simple; however, you must have administrator level access both to the Unicenter TNG server and the ServiceCenter server.

> **Important:** The SCAuto for Unicenter TNG product installs on one or more TNG servers, depending on the design of your Unicenter environment. If all TNG events of any significance are forwarded to a single TNG machine, then you need only install the product on that machine. SCAuto for Unicenter TNG is *not* normally installed on the ServiceCenter server, unless this also happens to be a Unicenter TNG server machine.

You must obtain several key pieces of information about your TNG and ServiceCenter installations before you begin. Installation takes only a few minutes if you have determined the correct host name and port number values beforehand.

The following is a checklist of requirements:

- Administrator authority at the TNG Server
- A userid and password with access to the TNG object repository
- Administrator level access to the ServiceCenter Server
- An authorization string for ServiceCenter which permits the use of the SCAuto TNG feature of ServiceCenter
- The TCP hostname or IP address of the ServiceCenter server
- The ability to assign (or obtain) a new TCP port number for use by ServiceCenter, if required
- The name of the TNG repository

Installation

ServiceCenter Configuration

Peregrine Systems recommends installing the ServiceCenter portion of SCAuto for Unicenter TNG prior to installing SCAuto for Unicenter on the TNG server. Installation steps are described in the following section.

Verify ServiceCenter Binary Version

Verify the release and compatibility of your current version of ServiceCenter by doing the following:

1. For Windows NT and UNIX installations of ServiceCenter, navigate to the directory containing the executables, e.g. the **RUN** directory, to issue the commands shown below.
2. For MVS, issue a **modify** command to the ServiceCenter started task containing the commands shown.
3. Check the version and build information:

   scenter -version

```
D:\Program Files\ServiceCenter\RUN>scenter -version
05/08/98 14:19:57   pid (658) ServiceCenter diagnostic report follows:
----ServiceCenter Version/Environment Details-----
  Executable name: scenter
          Version: 2.0 SP1e
        Timestamp: 9804060949
      System name: 12670
       System key: 0x61e13c00
    Hardware type: PC
 Network hostname: unknown
  Network address: unknown
 Operating system: Windows NT
       OS version: 4.0
-------------------------------------------------

D:\Program Files\ServiceCenter\RUN>
```

The version shown in the report must be 1.4 SP7 or later, 2.0 SP1 or later, or 2.1 SP1 or later.

If the version displayed is not in the above list, you should order the latest version and upgrade the binary executables. This is a simple and safe procedure, and much less time-consuming than an application (RAD code) upgrade.

Verify Event Services Applications

There are many RAD applications (**axces.read**, **axces.write**, etc.) which are critical to the correct functioning of SCAutomate adapter products. Even if you are running very current ServiceCenter binaries, you may have back-level versions of one or more of these applications, unless yours is a very new ServiceCenter installation, or you have recently performed an application upgrade. If you have a ServiceCenter system which has been customized to a significant degree and/or has not received a RAD application upgrade recently, you probably need to perform an upgrade. For 1.4 systems, either upgrade to 1.4 SP 7 or later, or upgrade to ServiceCenter 2.1. For 2.0 systems, either upgrade to 2.0 SP 1, or to Service Center 2.1.

Verify ServiceCenter License

1. Verify that ServiceCenter is licensed for use with SCAuto for Unicenter TNG
2. From the same command line, issue the following command:

   scenter -reportlic

The output should be similar to the following:

```
D:\Program Files\ServiceCenter\RUN>scenter -reportlic
--- ServiceCenter License Report ---
This is a Trial License
Expiration date:  09/22/1998
Licensed Client Platforms:  All
                 Licensed     Active    Inactive
Active Users:       50           0          0
Product/Features
   Problem Management
   Inventory Management
   SCAuto/CA TNG
```

3. Make sure that *SCAuto/ CA TNG* appears in the list of licensed options

Determine if SCAUTO is in Use

ServiceCenter has an optional server component known as **SCAUTOD**. The purpose of this component is to monitor a designated TCP port for incoming transactions from external applications such as **SCAuto for Unicenter**. This component may already be running in your ServiceCenter installation if you use certain optional licensed components. To determine if **SCAUTOD** is already in use, following the procedure:

1. Log in to the ServiceCenter server.
2. Click the **Status** button from the main menu. This action displays a list of all logged-in users and running background processes. Figure 1-1 below shows a sample status panel.
3. If you are using ServiceCenter version 1.4, look for an entry in the list of processes containing *S_C_AUTO*. If you are using version 2.x, look for an entry containing *SCAuto Server*.

If such an entry is present, SCAUTOD is already running. If not, you will need to customize ServiceCenter so that the SCAUTOD background process is running whenever ServiceCenter is up.

To do this, continue with the next few steps.

4. If you do not need to startup SCAUTOD, examine the *sc.ini* file to make a note of the SCAUTO port number, then go to *Verify that Event Services is Running* on page 1-9.

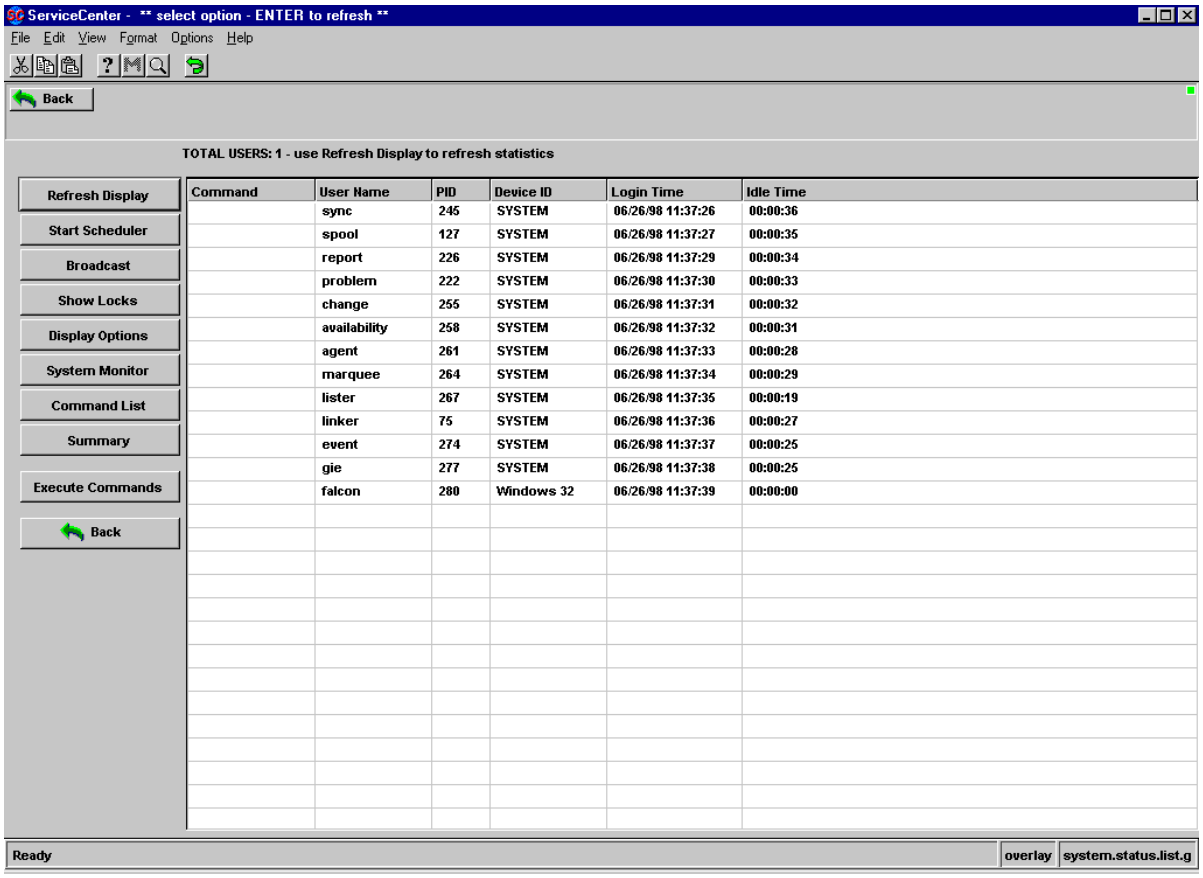

Figure 1-1 Scheduled Event Status List

Determine the SCAUTO Port

1. Check the *sc.ini* file in the **RUN** directory (for MVS, check the *PARMS* member in the **SAMPLIB**). It should contain a line beginning with *scauto*:, such as:

   scauto:12690

There might be a TCP port name (service name) coded following the *scauto*:scauto instead of a TCP port number.

2. If the *scauto*: parameter is not present, you will need to add it. This requires you to assign a new TCP port number not already in use by another application, for example *12690*. You should consult with your IP networking staff before arbitrarily choosing a TCP port value, since many installations administer these.

3. If the *scauto*: parameter is present, make a note of the value coded for it. This will be needed for installation of SCAuto for Unicenter.

Ensure that SCAUTOD Exists

There should be an executable called **SCAUTOD** among the executables on the ServiceCenter server. This ships with all ServiceCenter tapes and CDs. The **SCAUTOD** executable must be in the **RUN** directory for UNIX or Windows NT, and in the **LOADLIB** for MVS. Contact customer support if you cannot locate this component.

Ensure that SCAUTOD is Current (Windows NT and MVS only)

If you are running ServiceCenter 2.0 SP 1 on Windows NT or MVS, you may need to upgrade your binaries. If the SCAUTOD process fails to start, you should upgrade your binaries to 2.0 SP 1e on Windows NT, or 2.0 SP 1e+ on MVS.

If you are running ServiceCenter 1.4 SP 7 on MVS, you may need to upgrade your binaries. If the SCAUOTD process fails to start, you should upgrade your binaries to 1.4 SP 7b.

Upgrading binaries on NT is done via the setup program. Upgrading binaries on MVS is done by replacing the **loadlib** with a new one uploaded from a current distribution tape.

Start the SCAUTOD Process

Once you have:

1. Assigned a TCP port number to SCAUTO.
2. Coded it in the *sc.ini* file using the **scauto:nnnn** syntax described above.
3. Verified the existence of the SCAUTOD executable.
4. Made sure you have an authorization code which permits operation of SCAUTO.

you are ready to start SCAUTO.

5. Start an **express** client session. For Windows clients, this requires a shortcut (icon properties) of:

   scguiw32.exe *-express:*<hostname>.<servicename>

   instead of the usual

   scguiw32.exe *-system:*<hostname>.<servicename>

   ServiceCenter must also be configured to support **express** clients. Consult your ServiceCenter documentation regarding how to run *express-mode* clients if you are not familiar with this process. You must use an *express-mode* session when starting background processes such as SCAUTO.

6. Navigate to the **status** panel as before.
7. Press the **start sche** button at the bottom of the panel.

   The panel shown in Figure 1-2 below appears.

8. Choose the **scauto.startup** button.

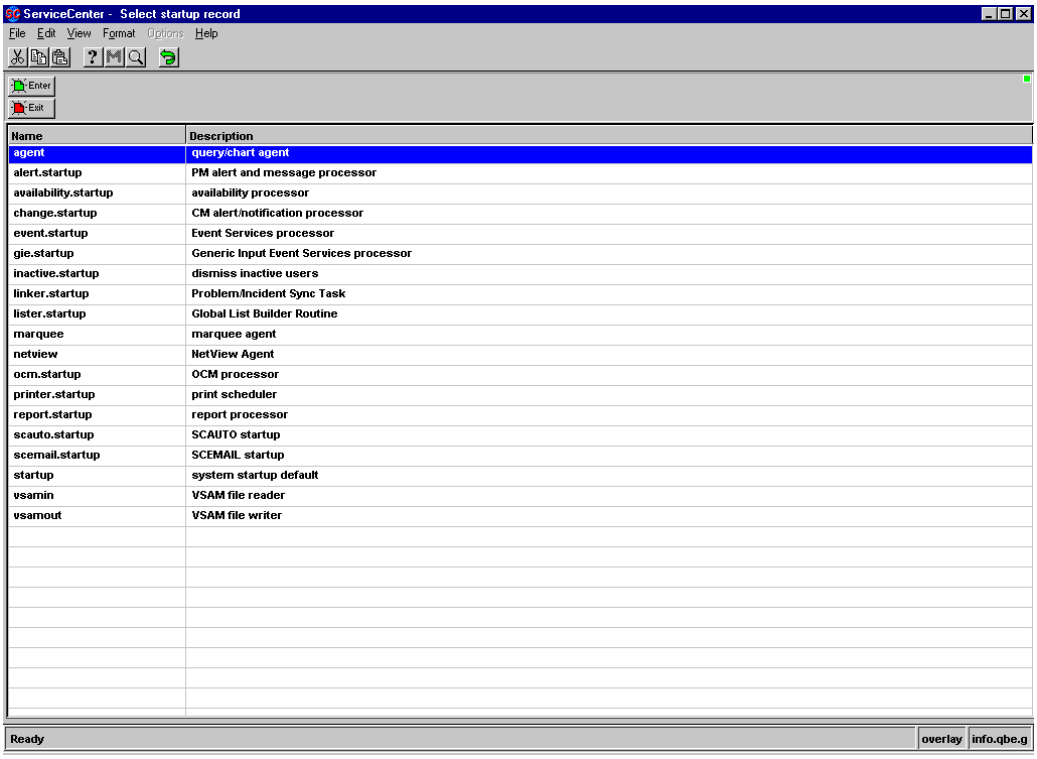

Figure 1-2 Start Scheduler List

9. Press the **Exit** button to return to the previous panel.
10. Verify that an entry labelled *S_C_AUTO* (for ServiceCenter version 1.4) or *SCAuto Server* (for version 2.x) is now listed among the running processes. If it is not listed, contact customer support for assistance.

Check the SC.LOG for Errors

1. Verify that no licensing issues or other problems exist.
2. Check the end of the *sc.log* file (*SYTERM DD* dataset on MVS) to make sure that no error messages were issued by SCAUTOD at startup.

Note: If an authorization problem exists, **S_C_AUTO** or **SCAuto Server** appears in the processes list, but error messages will be written to *sc.log*, and SCAUTO will ignore events from external applications such as SCAuto for Unicenter.

If SCAUTO started correctly, an entry similar to the following appears in the *sc.log* file:

```
665 05/08/98 14:24:57 SCAuto Base 2.0.7
665 05/08/98 14:24:57 SCAuto service name = 12690
665 05/08/98 14:24:57 main I:Initiating
```

Verify that Event Services is Running

1. Examine the status panel in ServiceCenter again, this time looking for an entry labelled *event*. The presence of this entry indicates that the background scheduler for Event Services is running.
2. If no such entry is present, start the *Event Services* processor in the same way as described earlier for SCAUTO, this time choosing the menu entry labelled *event.startup* (see Figure 1-2 above).
3. Perform this action from an *express-mode* client, or the background process will not be started correctly.

The purpose of Event Services is to process incoming events from external applications such as SCAuto for Unicenter, and give them to the appropriate applications. For example, a *pmo* event from SCAuto for Unicenter will open a problem ticket in ServiceCenter Problem Management. An *icma* transaction will add or update an inventory item.

Check the Event Scheduler Expiration Time

If Event Services has not previously been used in your installation (i.e you just started it for the first time as described above), one additional value must be checked for proper Event Services functionality.

1. Navigate back to the main menu of ServiceCenter by pressing the **Exit** button.
2. Select **Event Services** from the *Utilities* group of buttons on the main menu.

For ServiceCenter version 1.4:

3. Select **Event Scheduler** from the *Administration* group of buttons. This brings you to a form called *Event Scheduler*.

For ServiceCenter version 2.x:

4. Select **Agent Status** from the Services group of buttons. This brings you to a form called *SCAuto Check Status*.

For both ServiceCenter version 1.4 and 2.x:

5. Check the **Expiration** field in the top right corner. It should contain a valid date and time value.

   In some application releases of ServiceCenter, it may contain a date ending in **00**, e.g. 01/01/00. This is incorrect.
6. Change the date/time value to a valid date/time in the near future. This ensures that Event Services begins processing newly arrived events when the appointed time arrives.
7. If the Expiration date/time value is already a valid date/time in the near future, leave it unchanged.

Add SCAUTOD to SC.CFG (Windows NT)

For the Windows NT version of ServiceCenter, you should add an entry to the *sc.cfg* file so that SCAUTOD is automatically started each time ServiceCenter is brought up.

1. From the **Start** Menu, choose **Programs** / **ServiceCenter Server** / **Config File**. For Windows NT 3.51, double click the ***Config file*** notepad icon in the *ServiceCenter Serv*er program group.

   This starts a notepad session to edit the sc.cfg file, which is kept in the *\System32* sub-directory of the Windows NT server directory.

2. Add one line at the end of the file, containing the single word *scautod*. When done, your *sc.cfg* file should like something like this:

```
##   sc.cfg
#
#   Start ServiceCenter servers (regular and express)
scserver
scenter -express:12672
#   Start ServiceCenter background processes
scenter system.start -bg
scenter -sync
#   Start SCAuto transaction server
scautod
```

Important: Be sure to save the sc.cfg file when you have added the line for scautod. Note the *d* at the end of *scautod*.

This completes the ServiceCenter server portion of the installation of SCAuto for Unicenter.

Install SCAuto for Unicenter

Now you are ready to install the TNG server resident portion of SCAuto for Unicenter. Make sure you have the following items of information handy:

- TCP/IP hostname or IP address of the ServiceCenter server
- SCAUTO port number from the ServiceCenter *sc.ini* file
- Name of the TNG repository database which SCAuto for Unicenter should connect to
- Userid and password for SCAuto for Unicenter to use in connecting to the TNG repository database

1. Insert the SCAuto for Unicenter CD-ROM or the first diskette
2. Run the SETUP.EXE file.
3. During the installation, answer the prompts regarding name, company, directory/folder location, etc.
4. The final step of the install brings up a configuration dialog asking you for the items of information listed above.

   There is detailed help available explaining each item.
5. The setting for the *Monitor Enterprise Management Problem API* check box may not be initially obvious. Leave this box unchecked at this time. This feature is only intended for sites which have been making significant use of the TNG Problem Management application and associated features such as MGPT, and who wish to continue to use these in conjunction with ServiceCenter. If this applies to your installation, please consult the prerequisite customization instructions provided later in this manual. You can rerun this configuration dialog at a later time to select this option if desired.
6. When all the required items have been entered into the dialog, click the **Apply** button.

   This updates a file called *scuni.ini* in the **\Program Files\Peregrine Systems\SCAuto for Unicenter TNG** directory, and optionally replaces or restores the CA Enterprise Problem Management API DLL, depending on the options chosen. Details are provided if you press **Help**. If anything goes wrong, detailed error messages appear in the multi-line text window of the dialog.
7. Go into the Control Panel **Services** applet.
8. Update the entry for the SCAuto for Unicenter TNG service so that it runs under an appropriate privileged account, such as *administrator* or *caunint*, instead of defaulting to run under the Local System account. This is important as some components of SCAuto for Unicenter TNG may encounter an **Access Denied** security error (Error 5) under some conditions.

   To perform this update use the following procedure:

   a. Select **Start->Settings->Control Panel**.

b. Click on Services.

c. Find SCAuto for Unicenter TNG in the list.

d. Click startup.

e. Under **Log on As** click the radio button for *This account*.

f. Supply a valid userid, e.g., *administrator* or *caunint* and specify the correct password.

If you encounter a problem which you are unable to resolve, please contact Peregrine Customer Support.

Chapter 2 Operation of SCAuto for Unicenter TNG

Overview

This chapter describes how SCAuto for Unicenter TNG operates, and how to start and stop it.

SCAuto for Unicenter TNG is a Windows NT service, called **SCAUTOTNG**, and can be started and stopped from the Control Panel services applet, or by using the **net start** and **net stop** commands, or by invoking a specific startup application supplied for that purpose.

Starting SCAuto for Unicenter TNG

There are three ways to start SCAuto for Unicenter TNG:

1. Select **Start->Programs->SCAuto for Unicenter TNG->Start SCAuto for Unicenter TNG Service** from the menu.

   This executes the command **scunisrv -start** from the **\SCAuto for Unicenter TNG\Bin** directory.

2. You can also issue the following command from a Command window (the current directory in the command window is unimportant):

   net start scautotng

3. You can also execute the Control panel services applet to start SCAuto for Unicenter TNG.

All three of these methods cause the same code, and thus are completely identical in function.

Starting the **SCAUTOTNG** service normally takes from two to five seconds. If the service is unable to start for any reason, a message will be logged in the Windows NT system log, and detailed error messages will appear in the *scuni.log* file.

What happens when SCAuto for Unicenter TNG is Started

When SCAuto for Unicenter TNG is started, the main service executable ***scunisrv.exe*** is invoked. It then locates the file *scuni.cfg* which is found in the **\SCAuto for Unicenter TNG** directory. If this file cannot be found, the service will not start.

Next, the *scuni.cfg* file is processed. Each line in the file which is not blank or commented is interpreted as a process to run. Processes are typically either transient commands, i.e., invocations of *cmd.exe*, or

one of the following background processes installed with SCAuto for Unicenter TNG:

scevmon.exe

This process makes a TCP connection with the SCAUTOD component of ServiceCenter, using the hostname and port number information supplied to the configuration dialog program, *scunicfg.exe*. Whenever a connection is established, scevmon attempts to exchange events with ServiceCenter. **scevmon.exe** manages two queues of events on the TNG server: an inbound events queue and an outbound events queue. These queues are contained in the SCAuto for Unicenter TNG installation directory.

The name of the outbound queue is **scevents.to<host>.<port number>**. Outgoing events are logged to the *scevents.to.* file by processes such as the transient message action command **scevent.exe** or the long-running background process **sccoremon.exe** (discussed below) and consumed by scevmon.exe, which sends them to the ServiceCenter SCAUTOD process indicated in the file name of the **scevents.to** queue. When SCAUTOD accepts the events, they are written to the ServiceCenter EVENTIN file.

The name of the inbound events queue is **scevents.from<host>.<port number>**. This queue is created by **scevmon.exe** from events retrieved from the ServiceCenter EVENTOUT file via SCAUTOD. **scevmon.exe** determines its starting position in the EVENTOUT file of ServiceCenter by consulting a file called *syncfile.*<*host*>.<*service*> where <*host*> and <*service*> are the hostname and TCP port or service name of the SCAUTOD server. It then requests from SCAUTOD all events in EVENTOUT with sequence numbers greater than the value contained in the *syncfile*. The *syncfile* is updated as each event is received from SCAUTOD and successfully written to the *scevents.from* queue.

scunievent.exe

This process is responsible for processing the events logged in the *scevents.from*. file by scevmon.exe. **scunievent** reads these events, consults the appropriate script file(s) designated by the \EventMap\FromSC\event.ini file, and carries out the actions indicated in the script file(s). These actions invoke **cawto** to write a message to the Unicenter Event console. For example, when a problem ticket has been successfully opened, a **cawto** is issued containing the number of the problem that was automatically opened.

sccoremon.exe

This process is responsible for registering with the Worldview API to receive notifcations about updates to the TNG Common Object Repository. When it receives a notification, **sccoremon** retrieves information regarding the affected object from the repository, consults the \EventMap\ToSC\event.ini file to see what map files apply, then consults those map files to generate one or more events to ServiceCenter. The generated event(s) are logged to the scevents.to... file to be picked up and sent by scevmon. If for any reason you do not wish to generate ServiceCenter events based on CORE notifications, it is a simple matter to comment out the sccoremon line in the *scuni.cfg* file, using a # character as the first character on the line, or delete the line entirely.

Stopping SCAuto for Unicenter TNG

There are three ways to stop SCAuto for Unicenter TNG:

1. Select ***Start/Programs/SCAuto for Unicenter TNG/Stop SCAuto for Unicenter TNG Service*** from the menu.

   This executes the command **scunisrv -stop** from the **\SCAuto for Unicenter TNG\Bin** directory.

2. You can also issue the following command from a Command window. The current directory in the command window is unimportant:

   net stop scautotng

3. You can also execute the **Control Panel Services** applet to stop SCAuto for Unicenter TNG.

All three of these methods execute the same code, and thus are completely identical in function.

Stopping the SCAUTOTNG service normally takes five to ten seconds, except when CPU monitoring is in effect. If the service was instructed to monitor the CPU utilization via the optional **monitorcpu:1** parameter in the *scuni.ini* file, stopping the scautotng service may take up to twenty seconds.

What happens when SCAuto for Unicenter TNG is Stopped

Stopping the SCAUTOTNG service signals a Windows NT event called **SCAutoTNG.StopEvent**. This causes the various processes that were started by the service to immediately go into termination logic, where they release whatever resources they acquired from Windows NT and then exit. Should any process fail to exit voluntarily for any reason, the main service executable **scunisrv.exe** will terminate it using the Windows NT *Terminate Process* command. For this reason, the service should not normally fail to stop. In this unlikely event, detailed error messages should appear in the *scuni.log* file.

Determining if SCAuto for Unicenter TNG is Running Normally

You can use the Windows NT Resource Kit command **SC** (Service Control) to query the SCAUTOTNG service as follows:

sc query scautotng

This results in output similar to the following:

```
N:\tmp>sc query scautotng
SERVICE_NAME: scautotng
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 4  RUNNING
```

An alternative way to determine if SCAuto for Unicenter TNG is running is to use the Task Manager. If the service is running, the **scunisrv.exe** process is visible in the task list displayed by Task Manager, as will processes with the names ***scunievent.exe*** and ***sccoremon.exe***

Determining if SCAuto for Unicenter TNG is Processing Events

Examine the SCUNI.LOG file by choosing **Start/Programs/SCAuto for Unicenter TNG/ SCAuto for Unicenter TNG Log**

If SCAuto for Unicenter TNG is processing TNG events and/or CORE notifications successfully, messages appear similar to the following:

```
08/18/97  13:31:50      pid  (272)(324)  The  SCAUTO  server  to  be  used  is
204.33.93.69.12673
08/18/97 13:31:50  pid (272)(324) SCAUTO.DLL version is SCAuto SDK 2.0.4, build
Aug 17 1997
08/18/97 13:35:08   pid (272)(260) Successfully transmitted an icma event to
204.33.93.69.12673
08/18/97 13:35:13   pid (272)(242) Successfully transmitted an icma event to
204.33.93.69.12673
08/18/97 13:35:17    pid (272)(260) Successfully transmitted an pmo event to
204.33.93.69.12673
08/18/97 13:35:22   pid (286)(247) Successfully transmitted an icma event to
204.33.93.69.12673
```

Day to Day Administration

SCAuto for Unicenter TNG is designed for minimal administration overhead and maximum availability and operational flexibility.

SCAuto for Unicenter TNG can tolerate indefinite outages of Unicenter TNG and ServiceCenter. The operation of the latter products does NOT have to be synchronized with SCAuto for Unicenter TNG in any way; i.e., SCAuto for Unicenter TNG does NOT have to be stopped if ServiceCenter is being stopped or paused, nor is it necessary for those products to be running in order to start SCAuto for Unicenter TNG.

SCAuto for Unicenter TNG does not maintain a database. It produces a log file, but it automatically wraps this log when it reaches a defined maximum size.The default maximum log file size is 5 MB. This value is controlled by a setting in the *scuni.ini* file.

You should never need, therefore, to stop SCAuto for Unicenter TNG.

SCAuto for Unicenter TNG can be started and stopped at any time. However, long periods of Unicenter TNG operation without running SCAuto for Unicenter TNG should be avoided, as this will result in no automatically generated problem tickets and/or out-of-date information in the ServiceCenter ICM repository.

Similarly, long periods of Unicenter TNG and SCAuto for Unicenter TNG operation without running ServiceCenter should be avoided, as large numbers of outbound ServiceCenter events may accumulate in the **scevents.to...** file awaiting transmission to ServiceCenter.

Initial Loading of ServiceCenter Inventory

Following initial installation of the product, the ServiceCenter Inventory database can be *primed* by loading some or all of the data accumulated in the TNG object repository about autodiscovered machines and other objects of interest to ServiceCenter ICM.

Use of this feature requires licensing of the ServiceCenter Inventory and Configuration Management (ICM) component. ServiceCenter ICM contains an extensible object repository similar in concept to the TNG CORE component.

After initial loading of ServiceCenter inventory, automatic propagation of data from CORE to ICM will keep it up to date. Propagation of data about your network or business applications into ICM is extremely beneficial to the problem management and resolution process.

Initial loading is done by issuing commands on the TNG server where SCAuto for Unicenter TNG is installed, to copy objects from the TNG repository over to ServiceCenter ICM, by class of object, for example, all objects of class **IP_Network**, or all objects of class **WindowsNT_Server**.

After this initial loading, the arrival of occasional TNG objects adds due to autodiscovery, or other repository notifications due to status changes will cause SCAuto for Unicenter TNG to automatically generate inventory adds/updates and send them to ServiceCenter. Therefore, the initial loading procedure described below should only need to be done once.

Special Considerations

There are several important issues to consider before trying to process a large quantity of TNG objects using SCAuto for Unicenter TNG.

Although initial loading is something that only needs to be done once, it could have significant impact in terms of CPU (as much as 25% to 45% of a Pentium 233Mhz uniprocessor) and run for 5 to 15 minutes or more, depending on repository size and number of objects in each class. For this reason, you should perform the loading during off hours.

Also, the mapping between Unicenter CORE object classes and ServiceCenter ICM device types is controlled by a series of files contained in the \EventMap\ToSC\FromTNGObjectRepository directory. These mappings, as delivered, cover a small subset of the many types of objects which may be found in the Unicenter repository:

- IP_Network objects (mapped to ICM *Network* type)
- IP_Subnet objects (mapped to ICM *Segment* type)
- IP_Interface objects (mapped to ICM *interface* type)
- CISCO objects (mapped to ICM *router* type)
- IPX_Host objects (mapped to ICM *server* type)

- WindowsNT_Server objects (mapped to ICM *server* type)

These mappings are controlled by entries in the **\EventMap\ToSC\event.ini** file and by expressions at the top of each of the .map files listed in the event.ini file, as discussed in *Chapter 3 Customizing SCAuto for Unicenter TNG* on page 3-1.

The initial mappings as delivered with the product are based on default ICM device types delivered with ServiceCenter ICM. You may wish to modify these to conform to ICM customization performed at your site. You may also wish to add additional mappings to get more Unicenter objects into ICM. You can easily do this by "cloning" one of the existing .map files and adding entries to the event.ini file for the new .map files. The existing .map files are largely identical, except for the expression at the top, which tests for the class of CORE object being processed, and the corresponding ICM device type, such as "server" coded on one of the lines of the .map file.

It is also possible to revise all of the ICM device types to precisely match the object classes contained in the Unicenter CORE. This requires substantial ServiceCenter ICM customization, however.

It is recommended that you test the mapping between Unicenter CORE properties and ServiceCenter ICM device attributes with one or two objects of each class, before loading the whole class.

Test with One or Two Objects First

Before attempting to process all of the TNG data at once, be sure to test with one or two objects selected from the **Unicenter 2D Map.** This verifies that the connectivity between the TNG server and ServiceCenter is working, and that the mapping of TNG object properties to ServiceCenter inventory fields is set up as you wish.

1. Start the SCAuto for Unicenter TNG service.
2. Make sure ServiceCenter is started, and start the Unicenter 2D map.
3. Locate an object on the 2D map of the type whose mapping you wish to test.
4. Highlight the object and use the right mouse button to choose **Open Details**
5. Make a trivial change to the object to force an object repository update, for example, change the label of the object slightly.
6. Click **OK**.

   This causes the Unicenter object repository to be updated, and the instance of sccoremon.exe which is running in the background as part of the SCAuto for Unicenter TNG service is notified of the update.

   sccoremon.exe retrieves the object properties, along with parent and child object properties, and consults the **[CORE]** section of the **\EventMap\ToSC\event.ini** file to see what type(s) of ServiceCenter events should be constructed from the object data.

As delivered, the \EventMap\ToSC\event.ini file lists the following .map files in the [CORE] section:

icma = FromTNGObjectRepository\icma_ntserver.map

icma = FromTNGObjectRepository\icma_cisco.map

icma = FromTNGObjectRepository\icma_novell.map

icma = FromTNGObjectRepository\icma_ipnet.map

icma = FromTNGObjectRepository\icma_ipsubnet.map

icma = FromTNGObjectRepository\icma_unclasstcp.map

icma = FromTNGObjectRepository\icma_ipiface.map

Each of these .map files is consulted in turn, to see if it applies to the object in question.

The event generation expression(s) at the top of each .map file are evaluated to see if any of them is TRUE. If so, the map file is processed, and a ServiceCenter event is constructed in accordance with the map expressions. If not, processing of the .map file is terminated.

- If any ServiceCenter **icma** events are generated, they are appended to the scevents.to.<hostname>.<service> file for transmission to the ServiceCenter host and scauto port, which was configured during installation or when **scunicfg.exe** was last run.
- If scevmon.exe is running, it will attempt to transmit the **icma** events to the ServiceCenter scautod server/listener task. Otherwise the events will wait for later transmission.
- If ServiceCenter's scautod component is running, it will accept the **icma** event(s) and write them to the ServiceCenter EVENTIN queue. Otherwise, the events will await later retransmission.
- Once the event(s) have arrived in the ServiceCenter EVENTIN queue, if the ServiceCenter Event Services background scheduler is running, they will be processed at the next wakeup interval, which by default is 60 seconds. You may wish to shorten this interval to speed up event processing. If Event Services is not running, the events will wait for the next time Event Services is started.

7. When the icma event(s) are processed by Event Services, they will be added to (or updated in) the device file and related files. The EVENTIN queue should contain one or more **icma** events showing a userid of **Unicenter** and a status of *added* or *updated*. If the event is selected, and the **Messages** tab is clicked, messages similar to the following should be seen:

   Record updated in deviceparent file.

   Record updated in server file.

   Record updated in device file.

8. If you see error messages in the Messages tab and are unable to resolve them, contact Customer Service.

Loading ICM devices by TNG Object Class

After any connectivity, mapping, or other problems are resolved, you should copy over groups of objects, one class at a time.

To do this, instead of forcing object updates as discussed in the previous section, you can execute the sccoremon.exe directly from the command line, using the special **-discover:** parm to specify the object class. For example:

```
sccoremon -discover:CISCO
```

would retrieve all objects of class “CISCO” from the object repository, and generate a stream of **icma** events for devices of type “router” (assuming the delivered .map files are used unchanged).

```
sccoremon -discover:WindowsNT_Server
```

would retrieve all objects of class “WindowsNT_Server and generate a stream of **icma** events for devices of type *server*.

Before issuing a particular **sccoremon -discover** command, make sure that you have a .map file in the **\EventMap\ToSC\FromTNG ObjectRepository** directory which is set up to handle that kind of object, and make sure there is an entry in the \EventMap\ToSC\event.ini file, in the [CORE] section, which points to it. Otherwise, nothing will happen.

Chapter 3 Customizing SCAuto for Unicenter TNG

Overview

This chapter explains how to customize SCAuto for Unicenter TNG, using the event mapping files and script files contained in the **\Program Files\Peregrine\SCAuto for Unicenter TNG\EventMap** directory.

Note: You may not have to perform any customization of these files if you are testing SCAuto for Unicenter TNG in conjunction with an unmodified (*out-of-the-box)* ServiceCenter system. This is the best way to perform an initial evaluation of the product, as it permits you to observe successful bi-directional communication between ServiceCenter and Unicenter TNG with the least amount of effort.

However, if you have made significant changes to ServiceCenter Problem Management, it is possible *pmo* events sent to your ServiceCenter system by SCAuto for Unicenter TNG may fail some of these validation checks unless you customize it to send in problem events which meet these additional rules. If this happens, you may wish to consider having the ServiceCenter administrator at your site set up a new category of problem tickets for use by SCAuto for Unicenter TNG, using a standard Peregrine Systems, Inc.-supplied category such as *example* or *default* or *datanetwork* as a model.

Likewise, if you are working with a customized ServiceCenter rather than a new, *out-of-box* ServiceCenter, you may find that even if inbound events such as *pmo* are successfully processed, you do not see any outbound events being sent back to Unicenter in response. This commonly happens because the Format Control expressions in the Event Registration records for the event types in question may not be correct. Refer to the **Event Services Utility Guide** for more detail.

Concepts

SCAuto for Unicenter TNG uses two kinds of external event mapping control files:

.map files — These files in the \EventMap\ToSC directory tree define how one or more ServiceCenter events going *TO* ServiceCenter should be constructed from TNG event data, TNG object repository data, or TNG problem data.

This type of .map file has a special syntax which is oriented toward the creation of ServiceCenter event strings.

.scr files — These files in the \EventMap\FromSC directory tree define what actions should be taken with outbound events *FROM* ServiceCenter, for example, responses to *pmo* problem ticket open request events previously sent to ServiceCenter.

.scr files have a syntax which is more procedural in nature, because there is a broader spectrum of actions which might need to be taken in response to events from ServiceCenter.

This chapter explains how these files work and how to customize them to meet your site's requirements.

Any type of ServiceCenter event which is defined in your ServiceCenter server can easily be supported by SCAuto for Unicenter TNG, by ensuring there is at least one correctly coded .map file or .scr file pointed to by the *event.ini* file in the **\EventMap\FromSC** and/or **\EventMap\ToSC** subdirectory of SCAuto for Unicenter TNG.

Note: SCAuto for Unicenter TNG's **event mapping** function discussed here is not the quite the same thing as the ServiceCenter Event Services event mapping function.

Both types of event mapping definitions are used together. The event mapping functions within SCAuto for Unicenter TNG documented here are used to control the generation and emission of the event strings which are sent to ServiceCenter, and to specify the actions to be taken with the response events which are received from ServiceCenter. The event mapping functions defined within ServiceCenter Event Services, which are documented in the **Event Services Guide**, are used to interpret the event strings once they arrive in ServiceCenter's event input queue, and to specify what processing should occur within ServiceCenter in response to those events. For any given type of ServiceCenter event, the mappings defined in ServiceCenter must match those defined in SCAuto for Unicenter TNG.

How Event Mapping Works

All of the files which pertain to event mapping are kept in subdirectories of the **\SCAuto for Unicenter TNG\EventMap** directory.

The \EventMap\ToSC mappings are concerned with events sent TO ServiceCenter, and the \EventMap\FromSC mappings are concerned with events sent FROM ServiceCenter.

The *ToSC* map files are discussed next.

Events TO ServiceCenter

The events sent TO ServiceCenter are generated either by **scevent.exe**, which is a Unicenter message action command, or by **sccoremon.exe**, which receives TNG Object Repository update notifications.

TNG Event Console Integration

If you wish to generate input (for example, problem tickets) to ServiceCenter based on TNG Event Console messages, then you must create a TNG message action of the COMMAND type which specifies the Peregrine Systems, Inc.-supplied executable called **scevent.exe** as the command to be executed. This executable is kept in the \Bin subdirectory of the directory where SCAuto for Unicenter TNG is installed.

You should not copy scevent.exe to any other directory. It should be executable by TNG Event Console by virtue of the PATH environment variable which was updated when SCAuto for Unicenter TNG was installed. However, you may need to log off and log back on and restart the TNG Event Console Log application before the new path setting takes effect.

Invoking scevent.exe

When **scevent.exe** is specified in a Unicenter message action, the action type must be COMMAND, and the TEXT field must contain "scevent" followed by the type of ServiceCenter event desired for this message action. To open a problem, this will usually be *pmo*. To update a problem, it will *pmu*, and to close a problem, *pmc*. For example:

```
scevent pmo
```

Note: It is not necessary to specify conventional message action parameters, such as &1 or &TEXT. This is because all of the relevant variables for the Unicenter event which is being processed are automatically made available directly to scevent.exe by Unicenter, as environment variables beginning with the characters EVENT_.

For example, the value of &TEXT appears to scevent as the environment variable ***EVENT_TEXT***.

It is possible to specify additional variables of your own, or override existing Unicenter environment variables, by coding environment variable assignment statements on the command line of scevent.

For example, define a new variable called ***EVENT_PAGE*** containing a brief message to be sent to a pager by ServiceCenter when the problem is opened. You could code:

```
scevent pmo EVENT_PAGE="Application xyz crashed again"
```

You can also override the values of any Unicenter-defined ***EVENT*** variables if you wish. You may wish to force a variable such as ***EVENT_CATEGORY*** to contain a value which is understood by ServiceCenter, for example:

```
scevent pmo EVENT_CATEGORY=datanetwork
```

Notice the double-quote characters (“) to enclose the string for ***EVENT_PAGE***. This is because there were embedded blanks in the string. This was not necessary for the second example, because EVENT_CATEGORY was given a simple value without any blanks.

The following sections details what occurs when scevent.exe runs, and how a new variable such as EVENT_PAGE gets into a ServiceCenter *pmo* event.

The \EventMap\ToSC Directory

When a command such as

```
scevent pmo EVENT_CATEGORY=datanetwork
```

runs, it will consult the \EventMap\ToSC event mapping files as follows:

- The desired type of ServiceCenter event for this message action is examined. This must always be the first argument to scevent. In the example above, it is *pmo*.
- The *event.ini* file in the \EventMap\ToSC directory is consulted. A sample *event.ini* file appears as follows:

```
[ EVENT ]

pmo = FromTNGEventMgmt\pmo.map
pmu = FromTNGEventMgmt\pmu.map
pmc = FromTNGEventMgmt\pmc.map

[ CORE ]

icma = FromTNGObjectRepository\icma.map
icmu = FromTNGObjectRepository\icmu.map
icmd = FromTNGObjectRepository\icmd.map

[ EMPRB ]

pmo = FromTNGProblemMgmt\pmo.map
pmu = FromTNGProblemMgmt\pmu.map
pmc = FromTNGProblemMgmt\pmc.map
```

1. The section of the *event.ini* file which corresponds to the source of the event data is examined. For **scevent.exe**, this is always *EVENT*, meaning the source is TNG Event Management. Other

possible sources are *CORE*, meaning the TNG Object Repository, and *EMPRB*, if you use the optional MGPT integration feature. The latter is not recommended for ServiceCenter customers unless your site has a desire to continue using the Enterprise Problem Management applications delivered with Unicenter TNG, and ServiceCenter, both at the same time.

2. If there is no *event.ini* section with the value *EVENT*, no ServiceCenter events will be generated for this TNG event. For this reason, you should not change this value. If you ever want to suspend generation of ServiceCenter events from TNG event management, either disable the message action(s) which specify scevent, or shut down the SCAUTOTNG service momentarily.
3. If there is a section in the *event.ini* file for the type of data being processed, (*EVENT* in this example) each line in that section is consulted to see if the ServiceCenter event type to the left of the equals sign matches the value specified by the calling program, (*pmo* in this example). If a match is found, the string to the right of the equals sign is interpreted as a relative path to the event mapping file which describes how the event is to be constructed. For example, with the *event.ini* file shown above, the line:

   ```
   pmo = FromTNGEventMgmt\pmo.map
   ```

   will cause scevent.exe to try to open a file called "EventMap\ToSC\FromTNGEventMgmt\pmo.map".
4. The indicated *pmo.map* file is interpreted. The pmo.map file may contain one or more rules (expressions) governing the generation of the event transaction, and other expressions which generate the values for each of the sub-fields, or slots, in the **pmo** event.

Note: You may code more than one entry with the same ServiceCenter event type left of the equals sign in a particular section of an event.ini file. This permits you to generate multiple problem tickets from a single TNG event, or multiple ICM events from a single CORE notification. It also permits you to construct multiple different mappings, and use the event generation expressions in each map file to control which one is used for a particular event or class of object.

Note: Although the example event.ini file given above shows only problem events in the [EVENT] section and only icm events in the [CORE] section, there is no restriction as to the number or type of events which can be specified. It is perfectly valid to specify a mixture of event types in any section of the event.ini file.

Note: Although the scevent.exe message action command is usually specified with a single event type as the first parameter, it is also valid to specify a list of event types (in double quotes) as the first argument to scevent.exe. For example,

```
scevent "pmo pmc"
```

might be coded if for some reason you wanted to generate both a pmo and a pmc from the same TNG event. It is also valid to

code an asterisk instead of a specify event code or list of event codes:

```
scevent *
```

which means: *for this TNG event, process all of the .map files listed in the [EVENT] section of the event.ini file, without regard to event type.*

Some Sample ToSC .map Files

Here are two different sample ToSC.map files. Notice how the TNG data specifications in each .map file depends on the nature of the TNG data being processed, i.e., TNG Event data in the *pmo.map* file, and TNG Object Repository data in the icma.map file.

For TNG Event data, (i.e., in .map files which are specified in the [EVENT] section of the \ToSC\event.ini file) the TNG data specifications are special environment variables automatically populated by TNG when it launches message action commands.

For TNG Object Repository data, (i.e., in .map files which are specified in the [CORE] section of the \ToSC\event.ini file) the TNG data specifications are instance-level object properties.

\EventMap\ToSC\FromTNGEventMgmt\pmo.map

Below is a sample .map file, called *pmo.map*, which is used to generate ServiceCenter *pmo* events from TNG Event console messages, using the TNG Event Management environment variables:

```
# PMO.MAP
#
# This is for PMO events going TO ServiceCenter Problem Management
# being created from Unicenter Event Management environment variables

#---------------------- Start of Event Generation expressions ---------------
---------
#------------- One of these must evaluate TRUE, or no event is generated ----
---------

EVENT_JOBNAME != ""
EVENT_DEVICE != ""
EVENT_NODEID != ""
EVENT_LOGRECID != ""

#---------------------- End of Event Generation expressions -----------------
---------

#---------------------- Start of Event field specifications -----------------
---------

# Look for first non-blank value in { jobname, device, nodeid } for logical.name
# The "AFTER" business handles Windows NT domain name prefixes (which we dont
want)

EVENT_JOBNAME, EVENT_DEVICE, AFTER(EVENT_NODEID,"\\"), EVENT_NODEID^
AFTER(EVENT_NODEID,"\\"), EVENT_NODEID^
EVENT_LOGRECID^
EVENT_MSGNUM^
EVENT_DATEGEN + EVENT_TIMEGEN + EVENT_TEXT^
^
^
AFTER(EVENT_NODEID,"\\"), EVENT_NODEID^
EVENT_TYPE^
```

```
EVENT_CATEGORY^
^
^
^
^
EVENT_JOBNO^
^
^
EVENT_USERID^
^
^
^
TRANSLATE(EVENT_SEVERITY,"FSEW","1234")^
EVENT_SOURCE^
EVENT_TAG^
```

EventMap\ToSC\FromTNGObjectRepository\icma.map

Below is another sample .map file, called *icma.map*, which might be used to generate *icma* events from TNG object repository changes. Note that in this example, the TNG object class name is used directly for the **device type** field of the icma event, which would only make sense if your ICM application had been customized to use TNG-style object class names such as `WindowsNT_Server` instead of the more general types delivered with ICM, such as `server`:

```
# ICMA.MAP
#------------------ Start of Event Generation Section -----------------------
--
name != ""
#------------------ End of Event Generation Section ----------------------
name^
^
parent.name^
class_name^
address^
uuid^
location^
^
class_name^
^
^
uuid^
date_modify^
^
description,label^
^
containerhost^
^
^
^
^
^
^
address^
sysobjid,uuid^
^
^
^
contact^
^
^
```

How ToSC.map Files relate to ServiceCenter Events

ServiceCenter event strings consist of a header portion followed by a series of data fields separated by a separator character, which by default is the caret symbol, ^. Currently, SCAuto for Unicenter TNG requires the use of this symbol as the separator character.

SCAuto for Unicenter TNG .map files are concerned solely with defining the data fields portion of event transactions. This portion is called *EVFIELDS* in the SCAutomate documentation. These data fields correspond positionally to fields defined within the ServiceCenter event map for the event type in question.

ServiceCenter defines both an input and an output map for each type of transaction. For example, there is both an input *pmo* and an output *pmo*.

To see these formats, and to understand what the name and meaning of each field is, use the following steps:

1. Login to ServiceCenter as an administrator.
2. Go into **Event Services**.
3. Shoose **Maps**.
4. Select a particular input or output map to examine.

The header portion of the event is generated without reference to anything in the .map file. The most important part of the header is **evtype**, the event type, and this value has already been determined by the time the .map file is consulted. Other fields in the header include **evuser**, which is always *Unicenter* for events generated by SCAuto for Unicenter TNG, and the sequence number, also referred to as *checkpoint* value, which is an automatically generated value.

Most of the lines in an SCAuto for Unicenter TNG .map file are expressions which cause TNG event environment variables (or TNG object data in the case of map files for the [CORE] section of the \ToSC\event.ini file) to be emitted to the EVFIELDS portion of the ServiceCenter event.

EVFIELDS is constructed field by field as the file is sequentially processed. White space lines and comment lines are ignored.

Event generation criteria, which take the form of relational expressions, are evaluated when encountered, but do not cause EVFIELDS data to be generated.

How ToSC.map files are Processed

This section is intended to give you a basic idea of the structure of .map files which specify construction of ServiceCenter events. The rather different *FromSC*.map files, which specify actions to be taken in response to events from ServiceCenter, are discussed later.

Types of Lines Found in ToSC.map Files

There are several types of lines in *ToSC*.map files, including blank lines, comments, expressions, and event field definitions.

Blank Lines and # Comments

In processing a .map file, SCAuto for Unicenter TNG ignores any blank lines or lines starting with the **#** character, which indicate a comment.

Event generation expressions

Apart from blank lines and comments which are ignored, lines in a .map file must either be event generation expressions or definitions of event fields. Event generation expressions are evaluated but do not generate event field data, regardless of where they appear in the file.

If a line contains a relational operator such as =, !=, <, >, <=, >= etc., then it is an event generation expression which evaluates to TRUE or FALSE. At least one such expression must evaluate to TRUE for the event to be generated, but a given rule evaluating to FALSE does NOT prevent the event from being generated. In other words, the results of each expression are logically tied together with and *OR* statement.

Event generation expressions can be present anywhere in the file but it makes sense to place them at the top, as is shown in the example. Note that the lines in the sample .map files containing `# Start of Event Generation Section` and `# End of Event Generation Section` are just comments -- they do not have any special significance.

Event field specifications

Any line in a .map file which is not an expression and is not a comment or white space is interpreted as an event field specification. These cause TNG attribute data to be emitted into the current field of the buffer which is used for the construction of the event. Encountering a ^ character causes SCAuto for Unicenter TNG to terminate the current field with the ^ character it encountered and begin the next field of the evfields string.

Event field specifications are defined in terms of TNG Event environment variables, TNG Object properties, TNG Enterprise problem fields, and/or literal data, and are terminated with the separator character **^**. Event fields can also be empty, i.e., consist only of the separator character.

Positional nature of Event field definitions

Each line of the .map file which is not either an event generation expression or a blank line or comment corresponds positionally to the successive fields of the ServiceCenter event specification. A *pmo* or *icma* or other event has certain fields, in a particular order, each having a particular meaning, as defined in the ServiceCenter Event Map for the event in question. SCAuto for Unicenter TNG has no intrinsic knowledge of these event formats or their meaning. It simply constructs a buffer in accordance with the specifications found in a particular .map file and sends it. Therefore you must refer carefully to the ServiceCenter Event Map specification for a particular event while customizing a .map file. The scheme is totally positional.

Starting at the top of the .map file, each line in the .map file that defines event data to be emitted must correspond to the next sequential field in the ServiceCenter event to be generated.

Empty fields

All event field specifications in a .map file must be terminated with the ^ separator character. Event field specifications which contain only ^ cause empty event fields. This is used when there is no TNG attribute which corresponds to a particular ServiceCenter event field, or when that field is not used in your installation.

Specifying TNG Data in “ToSC” map files

TNG data specifications in a .map file can take several forms, depending on the source of the TNG data. The three sources of TNG data currently supported are:

1. TNG Event data
2. TNG Object data
3. TNG Problem data

> **Important:** Pay close attention to spelling and case when coding an TNG data specification in a .map file. You must code the TNG name exactly or the expected substitution will not occur.

TNG Event Data

For TNG Event data, the TNG data specifications are environment variables, usually having names that begin with **EVENT_**, for example **EVENT_CATEGORY** or **EVENT_NODEID**. However, as discussed previously, these variables can include environment variables of your own, provided you defined them on the command line which invoked the **scevent.exe** message action.

SCAuto for Unicenter TNG recognizes the environment variables and replaces the name of each variable with its value. If a variable has no

value, nothing is generated for that data specification, and processing continues.

The complete list of TNG Event environment variables which are automatically defined on entry to .map file processing is given in your Unicenter TNG reference guide. Some of these variables are not active in TNG 2.0, but are given values in TNG 2.1:

EVENT_CATEGORY	The CATEGORY field from the message.
EVENT_DEVICE	The DEVICE field from the message.
EVENT_JOBNAME	The JOB NAME field from the message.
EVENT_JOBNO	The JOB NUMBER field from the message.
EVENT_JOBQUAL	The JOB QUALIFIER field from the message.
EVENT_JOBSET	The JOB SET field from the message.
EVENT_LOGRECID	The record ID of the message in the console log.
EVENT_MSGNUM	The MESSAGE NUMBER of the message.
EVENT_PID	The pid of the process that caused the message.
EVENT_PROGRAM	The PROGRAM field from the message.
EVENT_REPLID	The reply ID for responding to the message.
EVENT_SEQNO	The sequence number of the current action.
EVENT_SEVERITY	The SEVERITY field from the message.
EVENT_SOURCE	The SOURCE field from the message.
EVENT_STATION	The STATION field from the message.
EVENT_TAG	The PLATFORM TAG field from the message.
EVENT_TOKEN	The ID of the message record in the database.
EVENT_TYPE	Type of message (MSG, CMD, WTOR, REPLY).
EVENT_UDATA	The USER DATA field from the message.

Object Repository Data

For TNG Object Repository data, (i.e., in .map files which are specified in the [CORE] section of the \ToSC\event.ini file) the TNG data specifications must be the names of class-level and instance-level properties of the object for which sccoremon.exe received a notification.

When sccoremon.exe receives a notification via the WorldView API, it retrieves the values of all of the object's properties. Then, while processing the .map file, it replaces the property names with the values.

Object Properties

Typical object properties have lower-case names such as:

- name
- address
- contact
- location

The complete list of TNG Object properties which are available to be coded in a .map file depends on what you have in your repository, and the class of object that is being added or updated. To see what sort of properties are associated with specific objects, use the Unicenter TNG Class Browser application. To see example values of properties for actual objects in the repository, use the Unicenter TNG Object Browser. You can also examine a Unicenter Repository Import / Export (TRIX) report.

Parent and Child Notation

When sccoremon.exe receives a notification about an object, it retrieves all of the properties for that object, and also retrieves all of the immediate parents and immediate children for that object. To refer to a property of a child or parent object, you can use the notation *child.xxx* or *parent.xxx* where *xxx* is one of the properties that the child or parent object has. You can use the **ANY( ), ALL( ), and NTH( )** built-in functions to refer to a particular child or parent instance, or any and all instances. If you have used event generation expressions to limit the processing of a particular map file to a particular class of object, or even to a particular object such as a specific BusinessView, then you are more likely to be able to predict what children or parent objects will exist for the object being processed. Currently, SCAuto for Unicenter TNG does not recursively retrieve all levels of children and all levels of parents, because this could potentially cause it to traverse the entire repository for a given notification. Instead, one level of children and one level of parents are retrieved.

Operators

There are two operators used in event field definitions; the choice operator and the concatenation operator. These are distinct from the relational operators found in event generation expressions.

Choice (,)

The , operator allows you to list several TNG data specifications separated by commas for a single ServiceCenter event field specification. This means that the first one of these TNG data specifications which evaluates to a non-null value is chosen and used in the construction of the ServiceCenter event.

Concatenation (+)

The **+** operator functions as a concatenation operator, allowing you to paste multiple TNG data specifications and/or literal values together.

In one of the sample given above, the **+** operator is used to paste together `EVENT_DATEGEN + EVENT_TIMEGEN + EVENT_TEXT^`

You can use this explicit form of concatenation to concatenate literal strings in double quotes with TNG data values. Note that the + concatentation operator inserts a blank between the concatenated values. There is another way to accomplish concatenation, called "implicit" concatenation, which does not insert any spaces. Implicit concatenation consists of deliberately deferring the specification of the ^ separation character for one or more lines of the .map file. This causes SCAuto for Unicenter TNG to evaluate each line and accumulate the result until the ^ character is seen and the current EVFIELDS subfield is terminated.

Built-in Functions

TNG data specifications may be enclosed in a function specification, such as **AFTER( )**, **ANY( ), ALL( ), BEFORE( ), NTH( ), SUBSTR(), TOKEN( ), TOUPPER( ), TOLOWER( )**, or **TRANSLATE( )**. These functions are discussed next.

Note: Only very limited nesting of functions is currently supported. You cannot nest two functions which have multiple arguments separated by commas. Also, if one of the functions has multiple arguments separated by commas, it must be the innermost function. For example, `TOLOWER( SUBSTR( foo, 0, 3 ) )` is valid, but `SUBSTR( TOLOWER( foo), 0, 3 )` is not. Another restriction on nesting is, `if ANY( )` or `ALL( )` is used, it must be outermost. Finally, you cannot use both `ANY( )` and `ALL( )` together, or `TOLOWER( )` and `TOUPPER( )` together, because it doesnt make sense to do so.

Despite all these restrictions, it is still possible to create strings in the following manner:

```
ALL( TOLOWER( SUBSTR( .....
```

This restrictions will be removed in a future release, when this specification language will be discarded and replaced with Tcl, Perl, Basic, Rexx, or some other common scripting language.

Important: Whenever any of these functions is used, the opening parenthesis must be specified immediately following the function name, for example, `ALL( ....` is valid, but `ALL (` is not.

AFTER( )

AFTER(*variable*, "substring") where *variable* is either a TNG event environment variable, or a CORE property name, etc. and "substring" is a literal string value in double quotes. AFTER returns the string text in *variable* which follows the substring given as the second argument.

So if you were to code `AFTER(EVENT_NODEID, "\\")`, and EVENT_NODEID contained *PS_SD\hp800*, then the expression would resolve to *hp800*. The backslash is coded twice because backslash is the escape character. Only backslash has to be doubled up in this fashion.

ALL( )

The ALL( ) built-in function allows TNG data values which may occur multiple times to be concatenated into a ServiceCenter array. This function concatenates the TNG data with | symbols, which denote an *array* to ServiceCenter event services. Be sure when using this function to make arrays, that the TNG data items being concatenated do not themselves contain | symbols. Use the TRANSLATE( ) function to convert any possible | characters in the data to some other value, such as blank or - or _. For example:

```
ALL(TRANSLATE(foo, “|”, “-”) )
```

ANY( )

The ANY( ) built-in function can be used in rule expressions to test TNG properties which may occur multiple times, for example, when testing all child instances of an object property for an object with multiple children.

```
ANY(child.status_no) > 0
```

BEFORE( )

BEFORE(*variable*, "stringvalue"). It works the same as AFTER(), only it returns the text in front of the indicated substring.

NTH( )

NTH( ) causes selection of a specific instance of an TNG property value which may occur multiple times.

For example, `NTH(child.name, 3)` would select the name of the third child object of the object for which we received a notification, provided such an instance exists. If the instance doesn't exist, no data is emitted by the specification.

SUBSTR( )

The SUBSTR operator allows you to take a substring of an TNG data value. Results of application of SUBSTR to other than string attributes are not well-defined. The first argument following the attribute name is the starting offset and the second argument is the desired length. SUBSTR() offsets are **zero-relative**. The length specification may be the special value * which means *the rest of the field*.

TOKEN( )

The TOKEN built-in function allows you to select a particular token from a string such as EVENT_TEXT. For example, TOKEN(EVENT_TEXT, 0) would be the first token. The special value *last* may specified as the second argument to permit selection of the final token without having to know how many tokens are in the string.

TOUPPER( )

The TOUPPER operator forces the data for the enclosed TNG attribute to upper case. For example, you could code:

```
TOUPPER(EVENT_NODEID)
```

TOLOWER( )

The TOLOWER operator forces the data for the enclosed TNG attribute to lower case. In the sample .map file shown above:

```
TOLOWER(EVENT_NODEID)
```

TRANSLATE( )

The TRANSLATE( ) function is used to convert all occurrences of a specific character within an TNG attribute to a different value. For example, TRANSLATE(Identification|Location, |, -) would convert all "|" symbols found in the Location attribute value to hyphens. Translation of | symbols to another value is done to avoid confusing ServiceCenter event services, which uses the "|" symbol to delimit array values. TRANSLATE( ) an also be used to accomplish a limited form of table lookup, provided single-character codes are involved. For example:

```
TRANSLATE(EVENT_SEVERITY, "FSEW", "1234")
```

converts EVENT_SEVERITY of *F* to 1, *S* to 2, and so on.

Special Variables

There is one special variable, $EVENT_TRIGGER, which can be coded in .map files. This is discussed below.

Built-in functions such as SUBSTR, TOLOWER, TOUPPER etc. currently cannot be used together with special variables.

$EVENT_TRIGGER

A special variable called ***$EVENT_TRIGGER*** is provided which is replaced by a text string containing a copy of all rule expressions which evaluated to TRUE, thereby triggering the event.

This can be used to include in the event somewhere (such as a problem description) the rule(s) which fired to cause the event to be generated.

Use of ***$PREV*** is dependent upon the successful generation at installation time of the history views supplied with SCAuto for Unicenter TNG. You may need to create additional history views yourself if you wish to reference the previous value of TNG attributes other than those referenced in the views created by SCAuto for Unicenter TNG.

Restrictions

- Expressions cannot be arbitrarily complex, they must be very simple relational tests using ==, !=, >, <, >=, <=, etc. against a particular TNG data value. For example, `EVENT_NODEID != ""` requires a non-null value for the network node name of the machine. Expressions can contain only relational operators used to test a given TNG attribute value. Expressions including parentheses and logical operators such as && and || are not currently supported.
- Built-in functions such as SUBSTR, TOLOWER, TOUPPER etc. currently cannot be used together with special variables.
- Only limited nesting of built-in functions is currently supported.
- The separator character must be ^.
- Expressions cannot be split between multiple lines. Continue the line to the right as far as is necessary to code the expression. There is no continuation character defined.
- If a literal string is coded as part of an event field definition, it must be enclosed in double quotes, and any embedded double quote characters within the literal string must be escaped with \.

Format of TOSC.MAP files

This section contains a more detailed discussion of the format of .map files.

The format of a .map file is as follows:

<item> <separator> newline

Separator character

The separator character is always ^. The current event sub-field is terminated when the separator character appears.

Implicit concatenation

Multiple .map file lines can contribute to a single EVFIELDS sub-field if desired, by not coding the ^ character until the end of the last .map file line for that EVFIELDS sub-field. This allows multiple items to be concatenated into a single EVFIELDS sub-field. This form of concatenation is implicit. There is an explicit concatenation operator, +, also provided, see *List of TNG Data specifications and/or literals* on page 3-22 for more details on the + character.

Null items

An item may be a NULL item, in which case the line is either completely blank, or only the separator ^ character appears. Completely blank lines are ignored. Lines with only the separator character generate an empty EVFIELDS subfield.

Comments

An item may be a comment starting with **#**, in which case the line is ignored, just as though it were completely white space. Any separator character at the end of a comment is ignored, since comments do not cause EVFIELDS data to be emitted.

Event Generation Expressions

An item can be a relational expression, in which case it functions as a rule controlling whether or not the event should be generated. If the expression evaluates to *true*, the event will be generated. Multiple such expressions may be coded. If any single one evaluates to *true*, the event will be generated.

Expressions exist only to control event generation. They do not cause any EVFIELDS data to be emitted, regardless of where they may occur in the .map file. As with comments, any separator character at the end of an expression is ignored.

Event generation expressions should be coded prior to any event field specifications, because when the first ^ character is encountered, if no true expression has yet been encountered, processing of the .map file is abandoned. This is to improve the efficiency of processing potentially many .map files for a single TNG event.

An expression has the form:

<TNG data specification> <relop> <value>

- The value must be a legal TNG data value as it would appear in the object repository, or as part of an event console message, or in a field of a Unicenter problem ticket.
- Expressions cannot be split between multiple lines. Continue the line to the right as far as is necessary to code the expression. There is no continuation character defined.

Literal strings

An item can be a literal string enclosed within double-quotes. Any double-quote characters within the string must be escaped with a backslash. The characters between the double quotes are emitted into the current EVFIELDS sub-field.

TNG data specification

An item can be a TNG data specification. The form of a TNG data specification depends on the source of the TNG data.

For event console data, it is simply the name of an environment variable. No *$* or other special character should be coded in front of the variable name. Coding a variable name always implies substitution of the variable's value for the name.

For TNG object data, a TNG data specification consists of an instance-level property name. The literal value *child.* or *parent.* may be prepended to the property name. Examples of instance-level property names follow. This is not an exhaustive list, since new properties can be added to the object repository at any time. Any property name found in your object database can be specified. However, if the object for which the notification is being processed does not have one of the properties listed in the map file, the value emitted for that data specification will be null.

```
uuid
label
status_no
hidden
propagate_status
severity
posted
acknowledge
autoarrange_type
date_ins
date_modify
class_name
address
address_type
```

```
ip_address_hex
subnet_mask
alarmset_name
pollset_name
contact
location
sysobjid
description
comment
interface_type
latitude
longitude
get_auth_name
set_auth_name
mac_address
```

For TNG problem data, a TNG data specification consists of the name of a Unicenter Enterprise Problem Management API variable name, as listed in the TNGSDK\INCLUDE\CAUPRB.H file, with the literal value *EmPrb|* or *EmPrbEnt|* prepended to it. *EmPrb|* indicates that the field is part of the problem ticket header, and *EmPrbEnt|* indicates that the field is part of a problem ticket log entry.

The following is the complete list of permitted problem-related names:

```
EmPrb|acID
EmPrb|acStatus
EmPrb|acCntrlNo
EmPrb|acLogName
EmPrb|acPriority
EmPrb|acLocation
EmPrb|acSerNum
EmPrb|acAssignedUser
EmPrb|acCategory
EmPrb|acComponent
EmPrb|acActBy
EmPrb|acRespArea
EmPrb|acEscTable
EmPrb|acContactName
EmPrb|acContactPhone
EmPrb|acOccurredName
```

```
EmPrb|acOccurredDate
EmPrb|acOccurredTime
EmPrb|acVendorNotifier
EmPrb|acVendorDate
EmPrb|acVendorTime
EmPrb|acEngineerContact
EmPrb|acEngineerDate
EmPrb|acEngineerTime
EmPrb|acResolvedContact
EmPrb|acResolvedDate
EmPrb|acResolvedTime
EmPrb|acSymptom
EmPrb|acSolution
EmPrb|acOccNode
EmPrb|acMGPTTrkID
EmPrb|acActByTime
EmPrb|acCreateDate
EmPrb|acCreateTime
EmPrb|acCreateUser
EmPrb|acUpdateDate
EmPrb|acUpdateTime
EmPrb|acUpdateUser

EmPrbEnt|acID
EmPrbEnt|acEntryNo
EmPrbEnt|acCreateDate
EmPrbEnt|acCreateTime
EmPrbEnt|acCreateUser
EmPrbEnt|acUpdateDate
EmPrbEnt|acUpdateTime
EmPrbEnt|acUpdateUser
EmPrbEnt|acNotes
```

List of TNG Data specifications and/or literals

An item can be a list of TNG data specifications and/or literal strings, separated by commas and/or concatenation operators.

A list separated by commas indicates that SCAuto for Unicenter TNG should select the first item which evaluates to non-null. For example,

```
EVENT_JOBNAME, EVENT_NODEID^
```

will result either in the value of `EVENT_JOBNAME` or `EVENT_NODEID` if `EVENT_JOBNAME` is empty, or nothing if both variables are empty.

TNG data specifications and literals can be mixed. For example:

```
EVENT_CATEGORY,"software"^
```

The above expression will result in either the value of EVENT_CATEGORY or the literal value *software*, if EVENT_CATEGORY is null.

Items in a list can themselves be concatenations of other items:

<data item or literal> + <data item or literal> < , data item or literal...>

- **+** indicates that the items on either side of the "+" should be concatenated with an intervening blank to form the field. If no blank is wanted in between the two items, use implicit concatenation instead by coding each *itemon* a line by itself, with a separator char after the last one.
- **,** indicates that if the a given item is null, blank, or otherwise not present, that the immediately following specification should be used instead.

Special variables

An item may be the special variable ***$EVENT_TRIGGER***, which is replaced by the literal relational expression(s) (if any) that evaluated to TRUE. If more than one relational expression was coded and evaluated to TRUE, they will be concatenated together with an intervening blank.

Events FROM ServiceCenter

SCAuto for Unicenter TNG is a bi-directional interface. Ordinarily there will be ServiceCenter event messages flowing from ServiceCenter back to Unicenter. This will be true unless this feature is disabled using the **-noeventsfromsc** parameter in the **scuni.ini** file, or unless ServiceCenter is customized in such a way that outbound events are not being generated.

Most of the outbound events generated by ServiceCenter will typically be email events, which are ignored by default. Ordinarily, the only outbound events from ServiceCenter which are of interest to SCAuto for Unicenter TNG are outbound problem events (*pmo*, *pmu*, *pmc*) which are generated in response to inbound problem events sent in by SCAuto for Unicenter TNG.

For example, if a message action causes an inbound *pmo* request to be sent to ServiceCenter, then an outbound *pmo* response should be

generated a short time later. By default, all that is done with these responses is to issue a **cawto** command to notify the operator that a problem ticket was opened in ServiceCenter. It is possible, however, to perform arbitrarily complex actions in response to outbound ServiceCenter events. This section documents the simple scripting facility which is used to accomplish this.

How Events are Obtained from ServiceCenter

Whenever the SCAuto for Unicenter TNG service is running, a process named *scevmon* is executing. This process maintains a TCP connection with ServiceCenter, and uses it to retrieve ServiceCenter event messages from the ServiceCenter EVENTOUT queue. These are appended to the local **scevents.from....** inbound events queue by **scevmon.exe** and later processed by **scunievent.exe** as described in *What happens when SCAuto for Unicenter TNG is Started* on page 2-1.

scunievent.exe is the component which determines what to do with each event from ServiceCenter, based on the configuration files found in the \EventMap\FromSC\ directory.

The \EventMap\FromSC Directory

Each outbound ServiceCenter event is processed by scunievent.exe as follows:

- The \EventMap\FromSC\event.ini file is consulted. As initially delivered this file contains the following:

```
[ Unicenter ]
pmo = ProblemMgmt\pmo.scr
pmu = ProblemMgmt\pmu.scr
pmc = ProblemMgmt\pmc.scr

[ Problem ]
pmo = ProblemMgmt\pmo.scr
pmu = ProblemMgmt\pmu.scr
pmc = ProblemMgmt\pmc.scr
```

  The sections of the .ini file correspond to ServiceCenter event ***evuser*** values found in the outbound event being processed. In other words, they represent the name of the application that created the outbound event. Events created by SCAuto for Unicenter TNG always have an evuser value of *Unicenter*. Events created by Problem Management on its own (for example, when escalation occurs) have an evuser value of *Problem*. If you wish outbound events containing other values in the evuser field to be processed, add a corresponding section to this file.

- The appropriate section of the \EventMap\FromSC\event.ini file is determined based on the evuser value of the event being processed. If there is no section with the appropriate name in the event.ini file, processing of the current event is terminated. This is not considered an error.
- If the section exists, but has no entries, processing of the current event is terminated. This is not considered an error.

- Each entry in the event.ini section is examined in turn. These take the form of assignment statements, with the left-hand side being the name of a ServiceCenter event type, and the right-hand side being a relative path within the \FromSC directory to a script file to be processed. Entries with ServiceCenter event types which do not match the event type of the outbound ServiceCenter event being processed are skipped. There may be more than one matching entry if desired, i.e. it is not an error to specify multiple entries with the same event type. This just means that multiple scripts will be run for that single ServiceCenter event.
- For each entry in the event.ini section whose event type matches, the script file indicated by the right-hand side is opened and interpreted.

Sample FromSC.scr Files

Since neither Windows NT nor Unicenter TNG included a standard scripting language such as Bourne shell, Perl or REXX at the time this product was developed, a simple ad-hoc language was developed for this purpose. It is intended to be replaced in the future with one of the afore-mentioned languages, or some other more suitable one.

Script for *pmo* events From ServiceCenter

Here is the fragment of script which is executed by default for each outbound ServiceCenter event of type *pmo*, which represent newly opened problems.

```
# if opened.by is not "Unicenter" this is not one of our problems

 if ( $5 != "Unicenter" )
  {
    msg( "Problem $2 not opened by Unicenter, ignored" )
    exit 0
  }

  exec cawto "Problem $2 opened by $5 in ServiceCenter"
  exit 0
```

This script checks the value of slot 5 in the outbound ServiceCenter event to see if it contains *Unicenter*, and if so, executes the CA utility program cawto to write a message to the Unicenter event console. The message written to the CA event log will contain the value of slot 2 (problem ticket number assigned by ServiceCenter) and the value of slot 5 (which will always contain *Unicenter* given the way this script is written).

If you wanted to always see a message in the TNG event log for all ServiceCenter problem open activity, not just for those problems opened by SCAuto for Unicenter TNG, you could comment out or delete the *if ($5 != "Unicenter")* clause in its entirety.

Script for *pmu* events from ServiceCenter

Here is the fragment of script which is executed by default for each outbound ServiceCenter event of type *pmu*, which represent newly updated problems:

```
# if opened.by is not "Unicenter" this is not one of our problems

 if ( $5 != "Unicenter" )
  {
    msg( "Problem $2 not opened by Unicenter, ignored" )
    exit 0
  }

  exec cawto "Problem $2 has been updated by $6 in ServiceCenter"
  exit 0
```

As before, the “opened.by” slot (slot number 5) of the event is checked to see if the newly updated problem was originally created by Unicenter. However the text of the message written to the TNG event console is slightly different, indicating that the problem was updated (rather than updated) and also gives the name of the operator or background process which performed the update (slot number 6).

Syntax of FromSC.scr files

The scripting facility is presently quite limited and is not suitable for complex applications. As mentioned above, it is planned for replacement with a standard language in the next release. The present syntax is very simple syntax and is intended only to permit the following:

- Control flow decisions to be made based on ServiceCenter event slot contents
- Command line programs to be executed
- Messages to be written to the *scuni.log* file

Some additional functions were developed to permit the CA Enterprise Management Problem API to be invoked. These are only useful if you wish to use the CA Problem Management system included with Unicenter TNG in conjunction with ServiceCenter Problem Management:

- Assignment of values to CA problem management variables defined in CAUPRB.H
- Invocation of specific functions within CAUPRBM.DLL
- Processing of return codes ($LASTCC) and feedback messages ($FEEDBACK) returned by the CAUPRBM.DLL functions

The following syntax elements are defined:

- *if* statements. *if* statements as implemented here have a strictly limited syntax for the *if* expression. Only simple relational expressions with a single operator and no parentheses are supported. The body of the *if* statement must be enclosed within a pair of curly braces, i.e., { } specified one to a line. *if* statements may be nested, i.e. a line contained between the { and } lines may be another *if*.

- Comments. Lines starting with # are treated as comments and are ignored.
- *exit* statements. The *exit* keyword may be followed by a numeric value. The *exit* statement causes interpretation of the current script to terminate.
- *exec* statements. The *exec* keyword must be followed by the path to an .exe file and a string in double quotes which contains the arguments to be passed on the command line. The *exec* statement causes a process to be started for the named executable.
- *msg* statements. These cause messages to be written to the scuni.log file. The *msg* keyword must be followed by a string in double quotes, enclosed in parentheses, as in the examples shown above.
- ServiceCenter event slot variables. These take the form ***$nn*** where *nn* is the slot number. For example, ***$2*** or ***$5***. These may appear in strings and will be replaced by the value of the specified slot.
- The special variable ***$EVENT*** is defined which is replaced by the event fields string.

Appendix A SCUNI.INI File Parameters

Overview

This appendix documents special parameters that can be coded either in *scuni.ini* or on the command line of a given executable launched via the *scuni.cfg* file. A setting on the command line of a given executable in the *scuni.cfg* file overrides any global setting specified in the *scuni.ini* file.

The appropriate values are automatically set when the product is installed. In normal operation, you should not have to change these parameters.

Important:	In general, parameter settings should not be changed unless suggested by customer support. Experimentation with these values without consultation with customer support could lead to unexpected results, or impair the operation of SCAuto for Unicenter TNG.

Logging Parameters

-log:

Controls where the log file is created. -log: is followed by a path specification, normally `scuni.log`, which causes the log file to be called *scuni.log* and written in the directory where the product is installed, which is normally the \Program Files\Peregrine Systems\SCAuto for Unicenter TNG\ directory.

-logappend:1

A required parameter when multiple processes and threads share a common log, which is standard with SCAuto for Unicenter TNG. This setting causes each process to append its messages to a common log file, rather than start a new log file.

-logmaxlen

Allows you to specify, in bytes, the maximum log file size. When the log reaches this size, SCAuto for Unicenter TNG will begin writing log information at the beginning of the file again.

-logpreservelen

Allows you to specify, in bytes, the amount of log file text which is to be preserved when the log wraps. This feature permits the log to wrap without losing all immediately preceding messages. The last logpreservelen bytes in the log are saved when the log wraps, and are written to the beginning of the new log.

Process and Thread related Parameters

-mt:

-mt:**1** enables multi-threading, and -mt:**0** disables multi-threading. This parameter currently has no effect, since this version of SCAuto for Unicenter TNG does not use multi-threading.

-maxthreads

Allows you to specify the maximum number of threads that any given process should create. This parameter currently has no effect, since this version of SCAuto for Unicenter TNG does not use multi-threading.

-retrymax

Allows you to specify the maximum number of times that SCAuto for Unicenter TNG will restart a failed process before deciding to shut itself down. The default setting is **3**.

CPU Monitoring Parameters

SCAuto for Unicenter TNG has the capability of monitoring the CPU utilization of its processes, and shutting itself down if specified values are exceeded. This feature is disabled by default. If enabled, care should be taken not to set the maximum CPU utilization values too low, or SCAuto for Unicenter TNG will needlessly terminate one or more of its sub-processes. If this happens more than *retrymax* times, the service will shut itself down. Use of this feature is only advised if a problem is experienced with a runaway (looping) SCAuto for Unicenter TNG process.

-monitorcpu:1

Enables CPU monitoring. Every 10 seconds, a snapshot of Windows NT performance counters is taken, and the percentage utilization of each SCAuto for Unicenter TNG process is examined.

-maxprocesscpu

Allows you to specify a value from 0.00 to 1.00 which is the maximum allowable CPU utilization for a single SCAuto process within the monitoring interval 0.99 would be 99%.

-maxtotalcpu

Allows you to specify a value from 0.00 to 1.00 which is the maximum allowable CPU utilization for all SCAuto processes within the monitoring interval. 0.99 would be 99%.

Event Mapping and Event Logging Parameters

-event_ini_path

Allows you to specify the location of the event.ini file which controls the event mapping process.

-sceventserver

Set by the configuration utility, and consists of the hostname or IP address of the ServiceCenter server and the SCAUTO port number or service name. These two values are separated by a period, for example *-sceventserver:helpdesk.12673*, and are used by the SCAUTO.DLL to connect to the SCAUTOD process running on the ServiceCenter server.

scevmon_sleep_interval

Specifies the length of time, in seconds, that the scevmon.exe process should sleep between polls of the ServiceCenter EVENTOUT queue to look for outbound ServiceCenter events. The default value if not specified is 5 seconds. The delivered value in the scuni.ini file is 1 second.

scunievent_sleep_interval

Specifies the length of time, in seconds, that the scunievent.exe process should examinations of the scevents.from... log to look for newly arrived outbound events from ServiceCenter. The default value if not specified is 5 seconds. The delivered value in the scuni.ini file is 1 second.

scevents

Specifies an optional list of event types which are to be retrieved from ServiceCenter'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 email messages, which have no meaning to Unicenter. 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 totally disable retrieval of outbound events from ServiceCenter, use the “noeventsfromsc:” option.

noeventsfromsc

Specifies whether or not scevmon.exe should retrieve events from ServiceCenter’s EVENTOUT queue. The default is 0 (which means events will be retrieved). To disable retrieval of all outbound events from ServiceCenter, code “noeventsfromsc:1”

-eventlogmaxlen

Specifies the maximum length of the scevents.to.... or scevents.from... event logs, before **scauto.dll** will attempt to purge them. The purge process is automatic, and involves rebuilding the affected file, while removing processed events. For example, purging the scevents.to... file involves removing events which have been successfully transmitted to ServiceCenter. Purging the scevents.from... file involves removing events from ServiceCenter which have been successfully processed by scunivent.exe. It is possible for a purge attempt to cause little or no reduction in the size of the event log; for example if few or none of the events qualifies for purging because ServiceCenter has been unavailable and the “to” log is full of unsent events. If a purge attempt fails, the new event that was to have been appended to the log is discarded, and a message to that effect is written to the scuni.log file. If this happens, investigate why events have not been being processed successfully. You can shut down the service and delete the scevents.from and scevents.to log files, along with the syncfile.. file without harming SCAuto for Unicenter TNG, if necessary.

Unicenter TNG Parameters

-tng_repository

This parameter is set by the configuration utility, and is the name of the TNG repository to be monitored by sccoremon.exe.

SCAuto for Unicenter TNG uses this value to connect to the repository.

-tng_userid

This parameter is set by the configuration utility, and is the userID for SCAuto for Unicenter TNG to use to connect to the TNG repository.

-tng_password

This parameter is set by the configuration utility, and is the encrypted value of the password to be used by SCAuto for Unicenter TNG with the tng_userid parameter to connect to the TNG repository.

Miscellaneous Parameters

-monitor_EmPrb_API:

-monitor_EmPrb_API:1 is set by the configuration utility, scunicfg.exe if SCAuto for Unicenter TNG is monitoring the Unicenter Enterprise Management Problem API. This means that the CA provided DLL called CAUPRBM.DLL has been renamed to CAUPRBM_ORIG.DLL and replaced with a Peregrine-provided DLL which supplies the same entry points. The purpose of this scheme is to allow Unicenter applications written to the CA API to work unchanged with ServiceCenter. To change the value of this setting, use the configuration utility. Do not manually update scuni.ini. If the value is changed manually, the configuration utility will not know it should replace or restore the CA DLL.

-killableprocesses:

-killableprocesses:**1** causes the processes which are started by SCAuto for Unicenter TNG to be killable by anyone using Task Manager or a similar utility. By default, the processes started by SCAuto for Unicenter TNG are created with NULL Windows NT security parameters, such that they are not killable via the Task Manager or *pview.exe*, or similar utilities. They can only be killed by shutting down the service, or by a program which acquires the special Windows NT debug privilege.

Debugging Parameters

Debugging parameters all begin with *-debug* and are given a value of **1** or **0** to turn them on or off. All debugging parameters are **0** by default. Debugging parameters are not intended for everyday use, but only for investigating why something is or is not happening as expected. For example, if events are not being generated as expected , **debugscautoevents:1** might be coded.

These settings should generally only be used if requested by customer support. The output of these commands is not designed to be intellible to the customer, but rather to the developers.

-debug:1

Causes the process to intercept **CTRL-C** and interpret this as a shutdown event. Used in conjunction with -event:0. Not for production use.

-debugall:1

The equivalent of coding :**1** for all other available -debug options. When **-debugall:1** is present in *scuni.ini*, enormous numbers of messages are written to scuni.log by all processes. This setting should not be used in normal operation, as it will slow down the product and cause the log to wrap frequently.

-debugcpu:1

Causes logging of CPU utilization monitoring.

-debugscautoevents:1

Causes information regarding construction and transmittal of events to be logged.

Index

M

N

O

P

R

S

T

W


Peregrine
SYSTEMS®