ServiceCenter®

Version 3

SCAuto for NetView OS/390

Version 1.0

August 2000

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

Peregrine SYSTEMS®
The Infrastructure Management Company™

This edition applies to version 3 of the licensed program for **ServiceCenter**®
and version 1.0 of **SCAuto for NetView OS/390**

Contents

Chapter 6 Generic Problem Services

Chapter 7 SCAuto API

Chapter 8 Inventory Automation

Chapter 9 Problem Determination

Index

Chapter 1 Introduction

Overview

Welcome to Peregrine Systems *SCAuto for NetView OS/390*. This product allows problem tickets in ServiceCenter to be automatically opened, updated, or closed for network alerts, abends, or other problems in the OS/390 system. Support is also provided for automatic collection and maintenance of inventory records in the ServiceCenter database, for network as well as locally attached devices.

Throughout the remainder of this guide, the product is referred to as SCAuto for NetView.

Prerequisite knowledge

It is recommended that implementers of SCAuto for NetView have specific knowledge in the following areas:

- Experience installing and customizing TME 10 NetView for OS/390.
- Knowledge of OS/390 MVS and UNIX System Services.
- Knowledge of RACF, or equivalent security software.
- Knowledge of ServiceCenter and the operating system platform on which it is installed.
- Experience customizing ServiceCenter, or the equivalent training.
- Knowledge of ServiceCenter Event Services.

Software requirements

- OS/390 version 2 (and above) with UNIX System Services, and HFS
- A system security product, such as RACF
- TME 10 NetView for OS/390 version 1.0, and above
- TCPIP
- ServiceCenter (may be on a remote system)

SCAuto for NetView OS390

The enterprise systems of today support vast, complex, multi-vendor networks. Management of these networks necessitate the use of high quality tools to automate and simplify routine tasks, capture information, and identify problems as soon as they occur. TME 10 NetView for OS/390, from IBM, is one such tool. SCAuto for NetView extends the capabilities of NetView by collecting problem data and inventory information, for forwarding to Peregrine Systems' ServiceCenter product.

ScAuto for NetView exploits NetView's Application Programming Interface (API) to provide integrated extensions to that environment. Various SCAuto components run as NetView command processor tasks, auto task commands, or as installation exit routines, and make use of the Message Queueing Services and the Program to Program Interface. An external component enables the subsystem to interface with ServiceCenter running on any supported platform, via TCPIP.

The following subcategories describe SCAuto for NetView's component features.

Basic Services

SCAuto for NetView functions and facilities are founded on a collection of base routines that provide common services to the other components. Data handling services provide for routing, filtering, formatting, and logging of information collected by the other SCAuto components. As data is received, it is passed through a series of user modifiable filters. Records that pass filtering are routed to the logging task, where they are added to the event log.

Basic Services also provides operator facilities for controlling the SCAuto for NetView system. Commands are provided for stopping and restarting SCAuto components, without recycling NetView. Other commands allow for refreshing or displaying filters.

SCAuto Basic Services, along with the Service Center Interface must be installed and properly configured, for supporting any of the other features. Other components may be deployed based on your installation's requirements.

ServiceCenter interface

The SCAuto Event Monitor provides a direct interface to Event Services within ServiceCenter. The Event Monitor, running as an MVS started task, establishes and maintains a ServiceCenter session. Events previously captured by SCAuto from within NetView, are read from the event log and

forwarded to ServiceCenter over TCPIP. ServiceCenter may be running on any supported platform. The Event Monitor is a required component for communicating data collected by SCAuto for NetView, to Service Center. No other software is required to interface between the systems, even if they reside on dissimilar platforms.

Network Alert Handling

SCAuto for Netview provides an interface for opening problems reported by NPDA and VTAM. As alerts are generated in the network, they are processed against user-defined filtering criteria to determine whether or not to open a problem record. In addition, an optional user-defined CLIST filter can be used to implement environmentally unique filtering rules.

Abend Reporting

Support for opening problems on abnormal job terminations, is collected dynamically, via an SMF installation exit routine. Information about the abend is formatted and passed to Central Services, where user-defined filtering criteria determine whether a problem should be opened. No modifications to existing JCL are required to take advantage of this feature.

Generic Problem Services

SCAuto for NetView provides facilities for collecting and reporting information for virtually any application or subsystem that produces system console messages. Messages are trapped through NetView's message automation facilities, and the data is used to populate the fields of problem records. The use can easily open, update, and close problem tickets based on existing system messages or new messages from applications. SCAuto comes pre-configured to open/close problems on JES, CICS, RACF, and other common messages.

Users of a Console Operations package also have the capability of generating problem records for failed attempts at automatic device recovery.

Application Programming Interface

SCAuto for NetView provides an Application Programming Interface (API) that supports user written REXX or NetView CLISTs for customized event reporting. Applications may be written to collect data, format it into a predefined event record structure, and issue an SCAuto command to route the event directly to the logging task.

Inventory Automation

ServiceCenter's configuration database can be built and automatically maintained by SCAuto for NetView. Information for locally attached devices, and network resources is used to create, and subsequently update the ServiceCenter inventory file with device names, parent devices, device types, and other data obtained from the system or network.

Contacting Peregrine Systems

For further information and assistance with ServiceCenter Mobilize.It!, contact Peregrine Systems' Customer Support. Current details of local support offices are available through these main contacts.

North America, South America, Asia/Pacific

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

Europe, Africa

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

Documentation Web Site

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

http://support.peregrine.com

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

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

Chapter 2 Installation

Overview

This chapter describes the installation of SCAuto for NetView, which consists of the following general tasks:

- Installation of the distribution files
- OS/390 system setup
- NetView configuration
- HFS setup
- Configuration and customization of optional components
- Installation verification

Installation should be performed by the systems programmer most familiar with, and responsible for installing and configuring NetView.

Upgrading from NAPA

SCAuto for NetView is the replacement product for Network Automated Problem Applications (NAPA), formerly available from Peregrine Systems. Current NAPA customers are encouraged to read this section carefully for information on the differences between the products, and how to plan and carry out a successful migration.

Why upgrade?

SCAuto for NetView implements new technology and features for managing problem and inventory information, while maintaining backward compatibility with existing NAPA data structures.

Improved interface to ServiceCenter

SCAuto for NetView supports logging of data to the standard event log, flat file, supported by other SCAutomate applications. By running the event monitor program, *scevmon*, OS/390 users may now easily interface directly to ServiceCenter, regardless of platform. The VSAM file and SC3270 interface required by NAPA has been replaced with a direct TCPIP interface to Event Services.

New Application Programming Interface

SCAuto for NetView provides a new Application Programming Interface (API) for submitting event data from NetView. This API supports the extended event record format, with a maximum of nearly 32K of data per record. Any High Level Language (HLL) for which a NetView services interface is provided, can be used to develop custom applications. Currently, these services are available for Assembler, PL/I, C, and REXX programs.

Gateway to future releases

This version of SCAuto for NetView implements an architectural foundation for a series of planned evolutionary enhancements to the product. Subsequent releases will extend this base to provide increased flexibility, ease of use, and processing improvements.

Migration considerations

SCAuto for NetView was designed to provide an evolutionary upgrade from its predecssor, NAPA, incorporating base functionality, and supporting data structure compatibility. There are, however, significant differences between the two products that must be taken into consideration for migration planning, as described in the sections below.

Operating system services

SCAuto for NetView exploits various system services not used by its predecessor. Each of these services must be fully configured and enabled. They include:

- OS/390 UNIX System Services
- Hierarchical File System
- OS/390 Security Server (for example, RACF)
- eNetwork Communication Server - TCP/IP services
- OS/390 Language Environment

NetView facilities

In addition to NetView's Commuinication Network Management interface, SCAuto also uses the Program to Program Interface for communication between some components.

SCAuto for NetView vs NAPA

Differences between the two products include:

- **New and/or modified component names:** This includes the names of all tasks, modules, CLISTs, messages, and operator commands. Message Automation Table entries, or filter CLISTs may have to updated to refelect these changes.
- **Elimination of the VSAM log file:** SCAuto for NetView uses a flat file, located on the Hierarchical File System, for logging event data to be sent to ServiceCenter.
- **Introduction of the Event Monitor:** The Event Monitor, SCEVMON, runs as an MVS started task. Note that although the execution of this task can be controlled through normal MVS system commands, SCEVMON is automatically started and stopped through SCAuto for NetView commands.

- **Discontinued APR support:** Automatic Problem Routing between distributed systems, via VTAM, is not supported in SCAuto for NetView. Problems are routed directly to the ServiceCenter system, over TCPIP.
- **Modified program constants:** Some program constants, used within problem open data records to reflect the source of the data, are different from those encoded by NAPA. Any ServiceCenter RAD applications that check for such constants explicitly, may have to be modified.

Required Components Installation

Step 1: Unload the distribution tape

SCAuto for NetView is distributed on a standard label tape. The volser is on the external label attached to the tape. All files are IEBCOPY unloaded PDSs. Contents of the tape are as follows:

File 1: Installation library

Dataset name: SCANV390.V1R0.INSTALL

Dataset attributes:

DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)

UNIT=3390,SPACE=(TRK,(7,1,20))

File 2: Load module library

Dataset name: SCANV390.V1R0.LOAD

Dataset attributes:

DCB=(RECFM=U,BLKSIZE=6144)

UNIT=3390,SPACE=(CYL,(10,2,50))

File 3: CLIST library

Dataset name: SCANV390.V1R0.CLIST

Dataset attributes:

DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)

UNIT=3390,SPACE=(TRK,(20,1,20))

Step 2: Install the SCAuto load modules

The dataset containing the SCAuto load modules, from file 2 of the distribution tape, must be concatenated on the steplib DD statement in the NetView start-up procedure.

Since NetView requires that all load libraries must be APF authorized, the SCAuto load library name must be added to the APF list in the PROGxx or IEAAPFxx member of the system PARMLIB.

Copy the Event Monitor procedure, SCEVMON, from the INSTALL dataset, to the PROCLIB containing the NetView start-up procedure.

Step 3: Install the SCAuto CLISTs

The SCAuto CLISTs, from file 3 of the distribution tape, must be made accessible to NetView. The dataset should be concatenated to DSICLD DD statement int the NetView start-up procedure. Alternatively, the members could be copied to an existing CLIST library.

Step 4: Update the NetView DSIPARM dataset

The following DSIPARM members are included in the SCAuto INSTALL dataset, from file 1 of the distribution tape. They should be copied to the NetView DSIPARM dataset:

FILTER

SCANPARM

SCANPRMT

The DSIPARM members described below must be updated, to define the SCAuto components to NetView:

DSIDMN	Add the SCAuto task definitions found in the DSIDMNX member of the INSTALL library.
DSICMD	Add the SCAuto command module statements from the DSICMDX member of the INSTALL library.
DSIOPF	Add the auto operator ID definition statements from the DSIOPFX member of the INSTALL library.

Step 5: Enable the Program to Program Interface

SCAuto for NetVeiw requires uses the Program to Program Interface (PPI). Ensure that the option is enable, by specifying the following in the start-up procedure for the Sub-System Interface:

```
PPIOPT='PPI'
```

Step 6: Install the autotask profile

Copy the SCAuto autotask profile, SCANPROF, from the INSTALL dataset to the NetView profile dataset, indicated on the DSIPRF DD statement in the NetView start-up procedure.

Edit the NetView initialization CLIST to add the following statement:

```
AUTOTASK OPID=SCANAUTO
```

The initialization CLIST is defined by the NCCFIC statement in the DSIDMN member of DSIPARM.

Step 7: Setup the HFS files

The Hierarchical File System (HFS) files are distributed in an archive file in the SCANVHFS member of the LOAD library. Copy this file to an HFS working directory with the TSO OPUT command, specifying the BINARY option. For example:

```
oput 'scanv390.v1r0.load(scanvhfs)' '/u/test/scanvhfs.pax' binary
```

Extract the archived files and directory structure with the **pax** command:

```
pax -rvf scanvhfs.pax
```

To identify the ServiceCenter system to SCAuto, edit the *ini* file:

```
/etc/scauto/scanv390/v1.0/run/scanv390.ini
```

In the line starting with *scauto:*, replace the word, *hostid* with the IP address (or host name, if using DNS), and the port designation for ServiceCenter. For example:

```
scauto:123.456.78.9.12690
```

or:

```
scauto:schost.12690
```

Step 8: Configure the Security System

SCAuto for NetView and the Event Monitor, SCEVMON, make use of OS/390 UNIX System Services, and must be permitted access to these services within the system's security product. Additionally, both must be defined with the same user ID and group ID. The following example assumes you are using RACF for security.

Define a group (NETVGRP in the example), and user (NETVSER), with access to UNIX System Services:

```
ADDGROUP NETVGRP OMVS(GID(1))
ADDUSER  NETVUSER DFLTGRP(NETVGRP),
         OMVS(UID(10) HOME('/') PROGRAM('/bin/sh'))
         PASSWORK(ABC123)
```

Add the NetView start-up procedure (NETVPROC in this example) to the RACF started class:

```
SETROPTS GENERIC(STARTED)
RDEFINE  STARTED NETVPROC.*
         STDATA(USER(NETVUSER) GROUP(NETVGRP))
SETROPTS CLASSACT(STARTED) RACLIST STARTED
```

Add the Event monitor procedure to the RACF started class:

```
SETROPTS GENERIC(STARTED)
RDEFINE  STARTED SCEVMON.*
         STDATA(USER(NETVUSER) GROUP(NETVGRP))
SETROPTS CLASSACT(STARTED) RACLIST STARTED
```

Refer to the documentation for RACF, or other security product installed at your location, for detailed descriptions of the commands.

Step 9: Verify the installation

Included in the CLIST dataset, file 3 of the distribution tape, is a REXX program that can be used for installation verification. Before running this command, however, NetView must be stopped and restarted to activate the new settings. On the ServiceCenter system, ensure that ServiceCenter is running, that SCAUTOD has been started, and that a valid license key for SCAuto for Netview OS/390 is in effect. Then, from a NetView operator console, enter:

```
TESTGENR
```

Sample problem data will be sent to the ServiceCenter system. In case of problems, check console messages, and the SCAuto log:

```
/etc/scauto/scanv390/v1.0/run/scanv390.log
```

Network Alert Handling (Optional)

Step 1: Modify members of the NetView DSIPARM dataset

DSIDMN:

Ensure that the following NetView TASK statement is present:

```
TASK MOD=DSIZDST,TSKID=DSIELTSK,MEM=DSIELMEM,PRI=2,INIT=N
```

DSIELMEM:

Two external log routines are provided with SCAuto for NetView. If you want to bypass NetView external logging, add the following statement:

```
DSTINIT XITXL=SCANVXLT
```

However, if other NetView external log exits are to be run, add the following statement:

```
DSTINIT XITXL=SCANVXLP
```

BNJMBDST:

Ensure that the following statement is present (starting in column 2):

```
REPORTS ON
```

FILTER:

FILTER is a new DSIPARM member, distributed with SCAuto for NetView. This member must be present and contain at least one record, either a valid filter record or a comment, for alert handling to be activated. This is a sample file, and should be reviewed and modified to reflect the specific network management objectives of your organization. See the section *The Alert Filter* in Chapter 4 of this document.

Step 2: Update the NetView Message Automation Table

DSIMSGXX:

Add the following statements to the Net View Message Automation Table to open and close problem records for INOPERATIVE RESOURCES, using VTAM messages:

```
IF MSGID = 'IST259I' & TEXT=TEMPTXT THEN
  EXEC(CMD('NETPROB ' TEMPTXT) ROUTE(ONE SCANAUTO));
IF MSGID = 'IST093I' & TEXT=TEMPTXT THEN
  EXEC(CMD('NETPROB ' TEMPTXT) ROUTE(ONE SCANAUTO));
IF MSGID = 'IST621I' & TEXT=TEMPTXT THEN
  EXEC(CMD('NETPROB ' TEMPTXT) ROUTE(ONE SCANAUTO));
IF MSGID = 'IST590I' & TEXT=TEMPTXT THEN
  EXEC(CMD('NETPROB ' TEMPTXT) ROUTE(ONE SCANAUTO));
```

These sample message automation statements are included in the DSITBLXX member of the INSTALL dataset, from file 1 of the SCAuto distribution tape.

Note: Be sure to check whether any other message automation statements are processing these message Ids. If so, merge these statements with the existing ones.

Step 3: Verify setup

From a NetView operator console, enter:

TESTALRT

This command invokes a REXX CLIST to generate a test alert, that will result in the opening of a problem record in the ServiceCenter database.

To test the VTAM message interface, enter:

TESTINOP resname

where *resname* is a valid COMC (NCP), LINE, or CTRL name in your network. This CLIST issues an INOP message, and a RECOVERY message, that will test the problem open and close processes.

If problems occur, check the NetView log, or the ServiceCenter message log.

Abend Reporting (Optional)

Step 1: Install the SMF Exit

The SCAuto for NetView LOAD library includes module, SCAACTRT, that implements the standard SMF exit, IEFACTRT. Copy this module to SYS1.LPALIB. Ensure that the IEFACTRT exit is activated, by checking the SMF parmlib member, SMFPRMxx. IEFACTRT must be specified on the EXITS option of the SYS and/or SUBSYS parameters. To define SCAACTRT to the Dynamic Exits Facility, add it to the PROGxx member of parmlib:

```
EXIT ADD EXITNAME(SYS.IEFACTRT) MODNAME(SCAACTRT)
```

Step 2: Activate the SMF Exit

The SMF exit routine may be activated without an IPL with the **setprog** system command. To enable the exit for batch jobs enter:

```
SETPROG EXIT,ADD,EXITNAME=SYS.IEFACTRT,MODNAME=SCAACTRT,
STATE=ACTIVE,DSNAME=SCANV390.V1R0.LOAD
```

To enable the exit for started tasks enter:

```
SETPROG EXIT,ADD,EXITNAME=SYSSTC.IEFACTRT,MODNAME=SCAACTRT,
STATE=ACTIVE,DSNAME=SCANV390.V1R0.LOAD
```

Step 3: Verify the installation

Force the abnormal termination of a test job. For example, submit a job or start a procedure, and cancel it before it completes normally to produce a 222 abend, or submit a job with an invalid program name specified on the EXEC statement, to produce an 806 abend. A problem open record will be submitted to ServiceCenter reporting the failure.

Generic Problem Services (Optional)

Step 1: Modify the DSIPARM PDS

Sample message automation statements, distributed in the DSITBLxx member of the INSTALL file, may be added to your current message automation table. These are start-up samples that should be reviewed for your installation's requirements. Verify that the MPFLSTxx member of PARMLIB allows the messages need to be automated via the NetView Message Automation Table.

The statements invoke the following CLISTs:

```
@HASP050       JES2 resource shortages.
@HASP310       Job terminated at end of memory.
@HASP406       Job was executing.
@ICH408I       RACF security violations.
@IEA404A       WTO Buffer shortage.
@IEA406I       WTO Buffer shortage relieved.
@IEC032I       DASD storage space problems.
@IEC150I       Dataset security violations.
@IEF453I       Job JCL errors.
@IFB081I       SYS1.LOGREC full messages.
@IKJ605I       Excessive invalid TSO logon attempts.
@IRA400E       Pageable storage shortage.
@IRA402I       Pageable storage shortage relieved.
@DFHSM         CICS under stress - short on storage
```

Note: Be sure to check whether any other message automation statements are processing these message Ids. If so, merge these statements with the existing ones.

Step 2: Verify the installation

From a NetView operator console, enter the command:

```
TESTGENR
```

This command invokes a REXX CLIST that will result in the opening of a problem record in the ServiceCenter database.

Inventory Automation (Optional)

Step 1: Allocate the inventory dataset

Sample JCL for allocating the inventory BSAM dataset is distributed in the INVDATA member of the INSTALL dataset.

Step 2: Modify the NetView start-up procedure

Add an INVDATA data definition statement to the NetView start-up procedure to identify the inventory dataset. For example, if the data set is named as defined in the sample allocation JCL, the following line should be added:

```
//INVDATA  DD  DSN=SCANV390.V1R0.INVDATA,DISP=SHR
```

Step 3: Modify the inventory configuration CLIST

Edit the INVCNFG member of the CLIST dataset:

- Ensure that inventory support is enabled:

  ```
  SCANVINV='YES'
  ```

- For testing purposes, indicate that only a sample of the full inventory configuration should be taken:

  ```
  INV_SAMPLE='YES'
  ```

Step 4: Verify the installation

From a Netview operator console, enter:

```
RFRESH NEW DASD
```

This command refreshes the ServiceCenter database with new DASD units discovered on MVS. (With the SAMPLE option in effect, the discovery is limited to 20 records.) Ensure that the inventory records are being populated in a manner consistent with your requirements. See Chapter 8 for more information on the RFRESH command and building the inventory database.

Important: Remember to turn off the sample option before attempting to build the inventory database:

```
INV_SAMPLE='NO'
```

Chapter 3 Operation

Overview

SCAuto for NetView is comprised of several tasks, that provide basic support for:

- Data collection
- Routing
- Filtering
- Event logging
- Communication with ServiceCenter
- Operator commands

Data collection

SCAuto for NetView acts as a single collection point for various types of data, arriving from multiple sources and interfaces. Network Alerts are trapped within NetView. Abend data are collected via the standard SMF exit routine, IEFACTRT. Information relating to a wide variety of products can be collected via NetView's Message Automation Services, and passed to SCAuto. Inventory data is generated by SCAuto's Inventory Automation Task, for locally attached and network devices.

Routing

Once data has been collected, routing services handle moving the data between the various components for command processing, filtering, and formatting. SCAuto uses both NetView's Message Queueing Services, and the Program to Program Interface for asynchronous communication of event data between components, and across applications.

User applications, written to use the CLIST API, can use the SCAuto routing services to send pre-formatted event data records directly to the routing task, to open, close, or update events in ServiceCenter.

Filtering

All data, except for customer defined events using the CLIST API (see below), are passed through the CLIST filter, SCANFLTR. This filter, which is user modifiable, may be used to select or reject alerts, inventory data, or other event related information. Subroutines are included in the CLIST for threshold and interval processing as well. Records that pass filtering are forwarded, via the command SCANOPEN, or otherwise, are discarded. Record layouts and information on using SCANFLTR, are contained later in this chapter.

If support for Network Alert handling is enabled, those data are pre-filtered, by the Alert filter, which is also user customizable. See Chapter 4 for more information on that filter.

Event logging

Data that pass filtering, or are provided through a custom application that implements the SCAuto API, are formatted into the standard scevent format, and written to the event log. The event log is a sequential DASD file, defined during the installation process, and identified in the NetView start-up procedure.

Communication with ServiceCenter

The SCAuto Event Monitor, SCEVMON, runs as an external started task. On start-up, the Event Monitor establishes a session with the ServiceCenter system, over TCPIP. As event records are written to the log, they are read in, and sent to ServiceCenter's Event Services. If the connection is lost, the Event Monitor will continue attempts to re-establish the connection. Data transfer resumes at the first record after the last one that was successfully transmitted.

Operator commands

SCAuto for NetView provides operator commands that may be used, in concert with standard NetView commands, to monitor and control the environment.

Running SCAuto for NetView

SCAuto for NetView provides commands that will start, stop, and display the status of the SCAuto tasks running in NetView. After installation, as described in the previous chapter, SCAuto will start automatically during NetView start-up. It is recommended that the component tasks be controlled using the supplied commands, and that they not be stopped and started individually with standard NetView commands.

SCANSTRT Start all SCAuto for NetView tasks, and the Event Monitor, SCEVMON, which runs as an external started task.

SCANSTOP Stop all SCAuto for NetView tasks, and signals SCEVMON to stop. Use this command to stop all SCAUto tasks before shutting down NetView.

SCANVDIS Display the status of SCAuto tasks in NetView.

INITFLTR Load the Alert filter into storage. Any accumulated threshold information is reset.

LOADCL SCANFLTR (REPLACE)
This standard NetView command replaces the SCAuto filter CLIST, SCANFLTR, in memory.

Note: The status of the Event Monitor, SCEVMON, can be displayed using standard MVS system commands.

Filter Processing

SCAuto for NetView provides the facility to control what data will create problem and inventory records in ServiceCenter, through filter processing. Since network environments differ in their requirements and size, the filtering function makes it possible to open problem records for events that warrant tracking. All problem and inventory data collected by the SCAuto components, are passed through filter processing.

Note that network alerts are pre-filtered the FILTER member of the DSIPARM dataset. Although the user has the capability to also filter alert data in SCANFLTR, it is recommended that SCANFLTR generally pass all Alerts. See Chapter 4 for more information on that filter.

Filtering is implemented as a user defined, optional, CLIST, SCANFLTR. Before a problem record is opened, or the inventory database is updated in ServiceCenter, the information must pass through SCANFLTR. The data may be:

- NPDA alerts that have passed through the Alert filter
- Abend records from SMF
- Generic problem data from CLISTs, invoked through NetView's Message Automation facilities
- Inventory configuration data for locally attached, and SNA devices collected by the SCAuto Automated Inventory task

Data is accepted, based on IF-THEN conditions defined in the CLIST. To accept a record, the command SCANOPEN is issued from within the CLIST, causing the data to be routed through to event logging. If SCANOPEN is not issued, the data are discarded.

Data passed to the SCANFLTR filter is uniquely identified by a five character token in the first message variable. The general format of the data is as follows:

MSGVAR(1) Data type identifier, five bytes.

MSGVAR(2) Domain name, 8 bytes.

MSGVAR(3) Resource name, 8 bytes.

MSGVAR(4) Resource type, 4 bytes

MSGVAR(5) Error descriptions, 48 bytes.

MSGVAR(6) Resource data, 72 bytes.

Refer to later chapters on using the optional features, for detailed field descriptions used by each.

The CLIST then, may be customized to accommodate installation unique characteristics, or situations that are transient in nature. Potential candidates for customization include:

- Preventing the opening of a problem for test jobs that abend, identified by a job naming convention.
- Preventing opening of multiple problems for a failing resource that generates repetitive NetView Alerts.
- Delaying problem open until a specified number of failures have been recognized, through Message Automation.
- Excluding certain resources from Inventory Automation, such as terminals or applications, for example.

Threshold and Interval Processing

The SCANFLTR CLIST provides subroutines that may be used to perform threshold and interval processing on the data received. Problems may then be opened only when a certain number of events occur, optionally, over a certain interval.

Interval and threshold support is implemented in the THRESHOLD function. This function is called for every occurrence of an alert. The number of occurrences, and the time interval are kept in task global variables.

The syntax of the function call is:

```
THRESHOLD ( ARG(1), COUNT {,'INTERVAL'}{,'FIRST'} )
```

THRESHOLD	Function name.
ARG(1)	As is (required).
COUNT	Number of occurrences (required).
INTERVAL	Time interval, in the format HH:MM:SS (optional).
FIRST	Literal. Allow to pass filtering on first occurrence (optional).

Return value: The function return TRUE if the number of occurrences has been met, and (optionally) within the time frame specified, and/or (optionally) this is the first occurrence.

Otherwise, the function returns FALSE.

Sample invocations:

Open a problem after every 5 occurrences:

```
IF THRESHOLD(ARG(1),5) = TRUE THEN SCANOPEN
```

Open a problem only if there are 2 occurrences in 1 hour:

```
IF THRESHOLD(ARG(1),2,'01:00:00') = TRUE THEN SCANOPEN
```

Open a problem at the first, and after every 3rd occurrence:

```
IF THRESHOLD(ARG(1),3,'FIRST') = TRUE THEN SCANOPEN
```

Chapter 4 Network Alert Handling

Overview

SCAuto for NetView provides a dynamic interface between NPDA, VTAM, and ServiceCenter. As alerts are generated in the network, they are processed against user-defined criteria to determine whether or not to open a problem record. In addition, an optional user-defined CLIST filter can be used to implement environmentally unique filtering rules.

Alert information is collected from the External Log Task, DSIELTSK, using the XITXL exit, and passed to SCAuto. Note that NetView drives the External Log Task for local alerts only, so SCAuto must be present on each NetView image in order to capture distributed system alerts.

The Alert Filter

The Alert Filter has been designed to enable a user to easily control which NetView alerts will open problem records. Only NetView alert data is passed through this filter. Output that passes filtering is routed to the optional SCAuto general filter, SCANFLTR.

NetView alerts are specified by product type, (NetView Block ID or Product ID), and type of incident (NetView Action Code or Alert ID). For example, the Block ID x048, which represents a 3174 controller, and the Action Code x16, uniquely identifies a specific type of alert on 3174 controllers:

```
CONTROL UNIT ERROR:TIMER
```

The filtering criteria are specified in a member named FILTER in NetView's DSIPARM dataset. This is the filtering criteria which controls problem record generation and may be easily modified using a facility such as ISPF. After modification, the filter can be made immediately active in an operational system by invoking the **INITFLTR** command within NetView. This command will cause the updated filter to be loaded into storage and will reset any threshold information that has been accumulating. Due to a NetView restriction, the FILTER member must contain at least one record, which may be a comment. However, in normal practice a significant number of records should be present.

A default set of filtering criteria is provided. To customize the filter, refer to ***Resource Alerts Reference*** (IBM publication SC31-6024), which lists action codes and descriptions for the various IBM product Block IDs.

The filter file should be reviewed and modified to reflect the specific network management objectives of your organization. Your network configuration, size, and complexity are reflected on your NetView Alert History screen, and this is a good place to start your analysis.

A simple way to identify the Block ID/Product ID and Action Code/Alert ID generated by NetView is to log onto NetView and access the NPDA Alerts History screen. Enter the alert selection number followed by *C* on the command line. For example, if the alert number was 3, enter *3 C* on the command line. NetView responds with one of the following messages indicating the Block ID/Product ID and the Action Code/Alert ID.

If the NetView alert was generated using the Block ID/Action Code format, the following message would be displayed:

```
BNJ962I AL/EV DESCRIPTION CODE FOR SELECTION 3 IS FFD3F
```

In this example, the Block ID value is FFD and the Action Code value is **3F**.

If the NetView alert was generated using the Product ID/AlertID format, the following message would be displayed:

```
BNJ378I SELECTION 3 FILTER CODE:  PRODUCT ID 5665362 ALERT ID 8263AE3D
```

In this example, the Product ID value is 5665362 and the Alert ID value is 8263AE3D.

Modifying the Alert Filter

The following keywords are used to define the contents of the FILTER member:

BKID NetView Block ID. The value associated with the BKID keyword must be a 3 byte hexadecimal value representing a valid Block ID. (e.g. BKID=04C).

ACT NetView Action Code. The value associated with the ACT keyword must be a 2 byte hexadecimal value representing a valid Action Code. An asterisk may be used as a special wildcard value for the ACT keyword. This is used to identify all alert types for a specific Block ID. (e.g. ACT=06).

PRID NetView Product ID. The value associated with the PRID keyword must be a 1-9 byte alphanumeric value representing a valid Product ID. (e.g. PRID=IBMPRID).

ALID NetView Alert ID. The value associated with the ALID keyword must be a 1-8 byte hexadecimal value representing a valid Alert ID. The value will be right justified in an 8 byte field and will be front end filled with binary zeroes if necessary. This process is consistent with the NetView GENALERT command. An asterisk may be used as a special wildcard value for the ALID keyword. This is used to identify alert types for a specific Product ID. (e.g. ALID=01BC22C3).

COUNT A threshold to be reached before opening a problem. The value associated with the COUNT keyword specifies the number of alerts of a given type for a specific resource (unique network name) that must be encountered before a problem will be opened. A new problem will be opened each time this value has been reached. COUNT can be used alone or in combination with the INT time interval keyword. The COUNT value must be an integer between 1 and 999. A value of 1 will open a problem on each occurrence. (e.g. COUNT=03).

INT Elapsed time interval. The value associated with the INT keyword specifies the elapsed time interval during which a pre-determined number (COUNT=n) of alerts of a given type for a specific resource must be encountered before a problem is opened. The INT keyword is invalid without the COUNT keyword. The format of the INT value is: hh:mm:ss; where hh (hours) must be 0-23, mm (minutes) must be between 0-59, and ss (seconds) must be between 0-59.(e.g. INT=00:05:30).

OPEN Problem open indicator. The value associated with the OPEN keyword indicates the action to be taken for this filter entry regarding opening a problem. A YES (or Y) value will open a problem and a NO (or N) value will not open a problem. The default is YES. (e.g. OPEN=YES)

General specifications

- An asterisk in column 1 will identify the entire line as a comment in FILTER.
- A keyword must be immediately followed by a value (e.g. COUNT=2).
- No comments are permitted in a filter entry in positions 1-50, unless an asterisk is placed in column 1 indicating that the entire line is a comment. Comments are permitted after column 50 provided they do not abut filter parameters and do not contain an equal symbol.
- If the INT keyword is specified, the COUNT keyword must also be specified and contain a numeric value greater than 0.
- If COUNT=1 is specified, OPEN=YES is assumed.
- If INT=00:00:00 is specified, it is disregarded.
- The keyword pairs BKID/ACT and PRID/ALID must always be used in a pair and must be consistent (e.g. the combination of BKID=value and ALID=value is invalid).
- If record SCA OPTS=1ST is in the SCANPARM initialization member, the COUNT keyword will open a problem on the FIRST occurrence of an alert and the Nth subsequent occurrences.
- If multiple filter entries are encountered with the same BKID/ACT values or PRID/ALID values, a message will be written to the NetView log and the last entry will be used for filtering.
- If the BKID/ACT or PRID/ALID is not specified for an alert type, the alert will unconditionally be filtered and a problem record will not be opened.

The filtering mechanism uses the combination of resource name, Block ID/ Product ID and Action Code/Alert ID to distinguish unique network failures. The combination of thresholds, time intervals and the OPEN keywords are used to determine when to open problem records.

Sample filter criteria entries

Example: 1

```
BKID=017     ACT=*      OPEN=NO
BKID=017     ACT=0B     OPEN=YES
BKID=017     ACT=11     OPEN=YES
```

These records indicate that for NetView alerts with a Block ID of **017** and an Action Code of **0B** or **11**, open problems. All other Action Codes for this Block ID will be rejected.

Example: 2

```
BKID=022     ACT=*      OPEN=YES
BKID=022     ACT=54     OPEN=NO
BKID=022     ACT=56     OPEN=NO
```

These records indicate that all NetView alerts with a Block ID of **022** will open a problem record except those with an Action Code of **54** or **56**.

Example: 3

```
PRID=5665362  ALID=8263AE3D   OPEN=YES
PRID=5665362  ALID=828AE21E   OPEN=YES
PRID=5665362  ALID=AB12CDEF   OPEN=YES
```

These records indicate that all NetView alerts with a Product ID of **5665362** and an Alert ID of **8263AE3D**, **828AE21E**, or **AB12CDEF** will open a problem record.

Example: 4

```
BKID=FF2   ACT=01  COUNT=3
BKID=FF2   ACT=0A  COUNT=2  INT=00:02:00  OPEN=Y
```

The first record indicates that any NetView alert with a Block ID of **FF2** and Action Code of **01** will open a problem record for every 3 NetView alerts received. Since the OPEN keyword was not specified in this filter statement, the default value of OPEN=YES is used. The second record indicates that any NetView alert with a Block ID of **FF2** and Action Code of **0A** will open one problem record for every 2 alerts received within 2 minutes. The counts are accumulated for a specific resource name for NetView alerts with the appropriate Block ID and Action Code. Each resource name will be qualified by its owning domain.

Example: 5 (used in conjunction with SCA OPTS=1ST in SCANPARM)

```
PRID=SCANALERT ALID=1  OPEN=Y  COUNT=99  INT=01:00:00
```

This record indicates that any consolidated alert from VTAM/ANO messages is to open a problem record on the first occurrence, but not again until 1 hour passes or 99 other alerts of the same type have occurred.

Alert Filter Initialization

The member containing filtering criteria is read during SCAuto for NetView initialization, and then it is freed. This enables the individual responsible for maintaining the filter member to make modifications without having to bring NetView down. At initialization, when an invalid filter record is detected, a message indicating which filter record is invalid and why, will be written to the NetView log and processing will continue.

All alerts processed that do not have a corresponding filter entry will be bypassed.

The filtering criteria can be reloaded at will by issuing the following command from within NetView:

```
INITFLTR
```

This command will refresh the area in storage where the interval and threshold statistics were maintained for the alerts processed so far. Any new alerts generated will be processed against the new filtering criteria.

Alert Consolidation

This feature of SCAuto for NetView, eliminates the need to modify and customize the Alert Filter for usability. Problem records can be opened on critical failures immediately after installation. The Alerts Consolidation function performs *intelligent filtering* with no user configuration needed. Customization of the Alert Filter and SCANFLTR are done for fine tuning purposes, not for product usability.

The Alerts Consolidation function processes VTAM inoperative messages to open a problem record on the failed resource. The problem record opened includes the complete resource hierarchy as well as the number of attached resources.

If NetView's ANO or any other automated network operation tool is being used, a problem record may be opened only after recovery attempts have failed for a specific resource. This can be accomplished by adding an entry in the NetView message automation table that invokes a command processor in NetView.

The alert generated for the inoperative resource will flow through the Alert Filter, and the user-defined CLIST filter, SCANFLTR. The Product ID/Alert ID for this new alert is SCANALERT/00000001, this PRID/ALID is included in the Alert Filter to accept these alert types. Additional filtering based on the resource name or type must be done in SCANFLTR.

Alert Consolidation error descriptions

As preconfigured, SCAuto for NetView will open problem records for network resources with the following error descriptions when driven by VTAM messages:

Resource	**Error Description**
NCP	VTAM INOP RECEIVED FOR A COMC - NN LINE(S) DOWN
LINE	VTAM INOP RECEIVED FOR A LINE - NN CTRL(S) DOWN
REMOTE SNA CTRL	VTAM INOP RECEIVED FOR A CTRL - NN TERM(S) DOWN
LOCAL SNA CTRL	VTAM INOP RECEIVED FOR A LOCAL SNA CTRL
DIAL-UP CTRL	VTAM INOP RECEIVED FOR A SWITCHED CTRL

Network Alert Problem Update/Close

Users have the ability to automatically update and close problem records opened for Network Alerts, through a simple CLIST API. In this way, problem records opened for an INOPERATIVE resource based on VTAM, or other recovery messages, can be updated or closed. The update/close alert data will flow through the SCANFLTR CLIST, allowing filtering comparable to the original alert.

The API is invoked by calling the CLIST, **UPDALERT**, from any user-written CLIST. The user CLIST must first declare the following variable names used by the UPDALERT CLIST, as task global variables:

UALERTNAME Network resource name that encountered the error, 8 bytes.

UALERTTYPE (not used)

UALERTDATA Additional close data description to be added to the problem record, 48 bytes.

Example:

Assume that an INOPERATIVE network resource, in this case L3708C1, opens a problem record. When the resource is recovered, UPDALERT can be invoked to automatically close the problem.

```
/*REXX*/
UALERTTYPE=''

/* Name used to open the problem record         */
UALERTNAME='L3708C1'

/* Close the problem record                     */
UALERTSTAT='CLOSE'

/* Close data to be added to the problem record */
UALERTDATA='VTAM RECOVERY SUCCESSFUL

/* Store the task global variables              */
'GLOBALV PUTT UALERTSTAT UALERTTYPE UALERTNAME UALERTDATA'

/* Call the UPDALERT CLIST                      */
'UPDALERT'

EXIT
```

SCANFLTR CLIST

Alerts are among the data passed to the SCANFLTR CLIST. There, alerts can be further filtered, based on resource type, resource name, and/or resource description. For more information regarding this filter, see the section *Filter Processing*, in Chapter 3.

The default SCANFLTR distributed with SCAuto for NetView, has sample statements that can be used to filter out specific types of alert data, however all alerts call the SCANOPEN command and will open problem records in ServiceCenter.

Alert data is presented to SCANFLTR in the following format:

Date/Time Date and time, "mm/dd/yy hh:mm:ss", 17 bytes

Record Type: Literal value, 5 bytes

"ALERT", for problem open

"UPD-N", for update

"CLS-N", for close

Domain: Originating domain, 8 bytes

Name: Failing/recovered resource name, 8 bytes

Type: Failing/recovered resource type, 4 bytes

Error Description: The unique alert message text (for open), or description information (update or close), 48 bytes

Hierarchy: The complete resource hierarchy in the following format:

name1	8 bytes
type1	4 bytes
name2	8 bytes
type2	4 bytes
name3	8 bytes
type3	4 bytes
name4	8 bytes
type4	4 bytes
name5	8 bytes
type5	4 bytes

Alert Data and ServiceCenter

Problem data generated for network alerts is transferred by the Event Monitor to ServiceCenter's eventin queue, in Event Services, as standard pmo, pmu, and pmc events. These events are in turn processed by ServiceCenter Event Services mapping, and made available to the Problem Management Applications.

The standard event formats and mapping services are described in the ***ServiceCenter Event Services Utility*** manual.

The Alert data record fields are mapped to the eventin fields as follows:

eventin Field	**Alert record**
logical.name	Resource Name
network.name	Resource Name
cause.code	BLKID and Action Code
action	Description
action2	Hierarchy data fields, 1 through 5
action3	Hierarchy type fields, 1 through 5
type	"ALERT"
domain	Domain
model	Resource Type

Chapter 5 Abend Reporting

Overview

SCAuto for NetView provides automatic problem entry of abnormal job terminations, into ServiceCenter. All abends and non-zero completion codes are captured, formatted, and routed through SCAuto for further processing and filtering.

Abend Processing begins with the SMF installation exit routine, SCAACTRT, that implements the IEFACTRT exit. This exit receives control when a job or step terminates, whether normally or abnormally. SCAACTRT extracts detail data from the SMF type 30, sub-type 4 and 5 records, for all abend or non-zero completion codes. The information is then routed to an SCAuto task for processing, over the Program to Program Interface.

Abend Filtering

All abend or non-zero completion job and step completion data is passed through the SCAuto filter, SCANFLTR. As with other data types, the abend records may be accepted, and forwarded to ServiceCenter, or discarded. The sample filter distributed with SCAuto for NetView demonstrates record selection with IF-THEN processing. As distributed, only records with completion codes higher than CC=0004 are forwarded for opening problems.

Abend data is presented to SCANFLTR in the following format:

Date/Time Date and time, “mm/dd/yy hh:mm:ss”, 17 bytes

Record Type: “ABEND” literal, 5 bytes

Domain: Originating domain, 8 bytes

Name: Job name, 8 bytes

Type: “JOB ” literal, 4 bytes

Job description:

60 bytes, as follows:

jobname	8 bytes
“JOB”	4 bytes
stepname	8 bytes
“STEP”	4 bytes
program name	8 bytes
“PROG”	4 bytes
JES job number	8 bytes
“JNUM”	4 bytes
user Id	8 bytes
“UID”	4 bytes

Abend description:

68 bytes, as follows:

“JOBSTRT” literal, 8 bytes

Job start date and time, “mm/dd/yy hh:mm:ss”, 17 bytes

“,CC=” literal, 4 bytes

Completion code, formatted as Xnnnn, where X may be:

- S - System Abend Code
- U - User Abend Code
- C - Non-zero Completion code

and nnnn is the completion code, 5 bytes

“,SYSID=” literal, 7 bytes

System Id, 4 bytes

“,STEPS ” literal, 7 bytes

Number of steps in the job, 2 bytes

“,FAILED IN ” literal, 11 bytes

Failing step number, 2 bytes

blank filler, 1 byte

Problem Update/Close for Abends

SCAuto for NetView provides a simple interface to allow users to update, or close a problem that was opened for an abended job, when the job is rescheduled, or successfully re-run. The update/close abend data will flow through the SCANFLTR CLIST, allowing filtering comparable to the original abend.

Note: As presented to the filter, MSGVAR(1), containing the 5 character type indicator, will be set to “UPD-J”, or “CLS-J”, to update or close the abend record for the job, respectively.

The interface is invoked by calling the CLIST, **UPDABEND**, from any user-written CLIST. The user-written CLIST must first declare the following variable names used by the UPDABEND CLIST, as task global variables:

UABENDNAME Job name that the original problem was opened under, 8 bytes

UABENDTYPE (not used)

UABENDDATA Additional close data description to be added to the problem record, 48 bytes

UABENDSTAT ‘UPDATE’ or ‘CLOSE’

Example:

Assume that a production job, PRODJOB, has abended, and that a problem has been opened. When the job is rescheduled for execution, UPDABEND can be invoked to automatically close the problem.

```
/*REXX*/
UABENDTYPE=''

/* Name used to open the problem record         */
UABENDNAME='PRODJOB'

/* Close the problem record                     */
UABENDSTAT='CLOSE'

/* Close data to be added to the problem record */
UABENDDATA='JOB WAS RESCHEDULED'

/* Store the task global variables              */
'GLOBALV PUTT UABENDSTAT UABENDTYPE UABENDNAME UABENDDATA'

/* Call the UPDABEND CLIST                      */
'UPDABEND'

EXIT
```

Abend Data and ServiceCenter

Problem data generated for job abends is transferred by the Event Monitor to ServiceCenter's eventin queue, in Event Services, as standard pmo, pmu, and pmc events. These events are in turn processed by ServiceCenter Event Services mapping, and made available to the Problem Management Applications.

The standard event formats and mapping services are described in the ***ServiceCenter Event Services Utility*** manual.

The abend data record fields are mapped to the eventin fields as follows:

eventin Field	**Abend record**
logical.name	Job name
network.name	Job name
cause.code	Completion code and reason code
action	Description
network.address	User Id
type	"ABEND"
category	"abends"
domain	Domain
objid	Program name
version	Step name
model	"Job "
serial.no	Job number

Chapter 6 Generic Problem Services

Overview

SCAuto for NetView provides for simple, automatic problem entry of user generated problem records, into ServiceCenter, through the use of NetView or REXX CLISTs. CLISTs can be created for any event that requires tracking.

SCAuto is distributed with a sample collection of CLISTs, for opening problems for a variety of events. Statements to invoke the CLISTs through NetView Message Automation facilities are also provided, for tailoring the automation table.

Processing within the CLIST should include:

- Collecting pertinent event data
- Defining task global variables
- Assigning the data to the task variables
- Invoking the GENPROB CLIST

CLIST Processing

The user-written CLIST is responsible for declaring and populating task global variables, that will be used by the GENPROB CLIST for submitting the data to the SCAuto for NetView system. The task global variable names expected by the GENPROB CLIST are:

GPMH1	First level component name, 8 bytes (required)
GPMT1	First level component type, 4 bytes (required)
GPMH2	Second level component name, 8 bytes
GPMT2	Second level component type, 4 bytes
GPMH3	Third level component name, 8 bytes
GPMT3	Third level component type, 4 bytes
GPMH4	Fourth level component name, 8 bytes
GPMT4	Fourth level component type, 4 bytes
GPMH5	Fifth level component name, 8 bytes
GPMT5	Fifth level component type, 4 byes
GPMTYPE	Constant Record type="GENER", 5 bytes
GPMD	User defined problem description, 48 bytes (required)

In order to successfully open a problem record, the CLIST must minimally provide:

- The resource name of the problem component (GPMH1)
- The resource type of the component (GPMT1)
- A brief description of the problem (GPMD)

Failure to supply a value for any one of these three fields will result in an error message recorded in the NetView log. Additionally, there may be no gaps in the specification of hierarchy values or incomplete specification of a hierarchy pair.

For example, if the CLIST sets values for the task global variables, GPMH1, GPMT1, GPMH3, and GPMT3, skipping GPMH2 and GPMT2, then an error message would be recorded indicating that an invalid order of hierarchy values was specified. Likewise, if values are set for GPMH1, GPMT1, and GPMT2, omitting GPMH2, a message would be logged indicating a hierarchy pair had an incomplete specification.

The component fields (GPHM1 - GPMT5) designed to accommodate the full network hierarchy of a failing component. A generic problem generated for a given resource may not necessarily have five levels of hierarchy to be specified. The component entries should be populated from the first to the fifth level (from minor to major severity) since filtering will be performed for the last component entry filled.

Once the task global variables have been assigned as appropriate, the CLIST, GENPROB, should be invoked. GENPROB reformats the data, to include the originating domain, and a date and time stamp for the problem. The record is then routed into SCAuto for NetView for processing and filtering.

Note: For the CLISTs to be available to NetView, they must reside in the one of the partitioned datasets identified by the DSICLD DD statement in the NetView start-up procedure.

To activate the CLIST via NetView's Message Automation, a statement must be added to the Message Automation Table, the DSITBLxx member of the DSIPARM dataset.

Example:

Assume that a CLIST, named @IEC150I, has been written to open a problem for message IEC150I. The following statement should be added to the Message Automation Table:

```
IF MSGID='IEC150I' & TEXT=TEMPTEXT
  THEN EXEC(CMD('IED150I' TEMPTXT) ROUTE(ONE SCANAUTO))
```

Before activating the modified message automation table, issue the following NetView command to verify that the table does not contain syntax errors:

```
AUTOTBL MEMBER=DSITBLxx,TEST
```

where DSITBLxx is the member name of the Message Automation Table that was modified above, or that contains a %include statement for it.

To activate the modified table, enter:

```
AUTOTBL MEMBER=DSITBLxx
```

Filtering Generic Problems

As with other problem or alert records, the generic problem records may be discarded or kept based on user defined criteria such as thresholds, or frequency of occurrence. Refer to the section *Filter Processing* in Chapter 3 for general filtering information.

Generic problem data is presented to the filter, SCANFLTR, in the following format:

Date/Time Date and time, *mm/dd/yy hh:mm:ss*, 17 bytes

Record Type "GENER" literal, 5 bytes

Domain Domain, 8 bytes

Name Resource name, 8 bytes

Type Resource type, 4 bytes

Hierarchy: The complete resource hierarchy in the following format:

data1	8 bytes
type1	4 bytes
data2	8 bytes
type2	4 bytes
data3	8 bytes
type3	4 bytes
data4	8 bytes
type4	4 bytes
data5	8 bytes
type5	4 bytes

Update/Close for Generic Problems

Generic problems that have been opened through CLISTs, via NetView Message Automation, may be updated or closed in the same manner. The update/close data will also flow through the SCANFLTR CLIST, allowing filtering comparable to the original problem.

Note: As presented to the filter, MSGVAR(1), containing the 5 character type indicator, will be set to UPD-G, or CLS-G, to update or close the problem record, respectively.

The interface is invoked by calling the CLIST, **UPDGENER**, from any user-written CLIST. The user-written CLIST must first declare the following variable names used by the UPDGENER CLIST, as task global variables:

UGPMNAME Resource name for which the original problem was opened, 8 bytes

UGPMTYPE (not used)

UGPMDATA Additional close data description to be added to the problem record, 48 bytes

UGPMSTAT UPDATE or CLOSE, 5 bytes

Example:

Assume that a problem was opened for message IEA404A, indicating a WTO buffer shortage. When the shortage is relieved, and message IEA406I is displayed, UPDGENER can be invoked to automatically close the problem. See members @IEA404Aand @IEA406I in the distribution CLIST library for implementation details.

Generic Problem Data and ServiceCenter

Generic problem data, that passes filtering, is transferred by the Event Monitor to ServiceCenter's eventin queue, in Event Services, as standard pmo, pmu, and pmc events. These events are in turn processed by ServiceCenter Event Services mapping, and made available to the Problem Management Applications.

The standard event formats and mapping services are described in the ***ServiceCenter Event Services Utility*** manual.

The generic problem data record fields are mapped to the eventin fields as follows:

eventin Field	**Generic problem record**
logical.name	Resource name
network.name	Resource name
cause.code	"SCANV390 GENERIC PROBLEM"
action	Description
action2	data1-5
action3	type1-5
type	"GENER"
domain	Domain
model	Resource type

Chapter 7 SCAuto API

Overview

SCAuto for NetView introduces an alternative interface for opening, updating, or closing problems in ServiceCenter. For the first time, NetView users can take advantage of the Event Record structure supported by other SCAuto products. The Event Record structure supports the ServiceCenter maximum of 32K bytes for events. The records can be produced by any language for which a NetView services interface is provided, including Assembler, PL/I, C, and REXX. SCAuto provides a command, SCANSEND, for routing the formatted event directly to the logging task.

Event Record Structure

The SCAuto Event records are comprised of comma delimited header data, followed by the event fields.

evtype Event type, maximum 17 bytes. May be one of the standard ServiceCenter event types, for example *pmo* (problem open), *email* (electronic mail), etc., or a user-defined custom event.

evtime Date and time the event occurred, maximum 20 bytes. The format of this field should be as expected by the event scheduler. This can vary depending on the event scheduler's operator record, and ServiceCenter defaults. By default, the event scheduler uses Universal Time, and the format: mm/dd/yy hh:mm:ss.

evsysseq A reserved sequence number for the event, maximum 33 bytes. This is a keyed field that must be unique, so any values placed in this field are ignored when creating events. However, a comma is required as a place holder for the field.

evusrseq User specified sequence number, maximum 33 bytes. May be used for any purpose the application requires.

evsysopt	Reserved, maximum 25 bytes. A comma is required as a place holder for this field.
evuser	User or application for which this event is scheduled, maximum 25 bytes.
evpswd	Application password, if required, maximum 25 bytes.
evsepchar	Application separator character, 1 byte. This is the character used to delineate the event fields in evfields. Most commonly, this character is set to the character **^**.
header end	A single character to indicate the end of the header data, and the beginning of the event fields. Must be either a colon (most usual), or a comma.
evfields	Event fields. Data specific for the application. For the standard event types, all sub-fields within evfields are separated by the character specified in evsepchar. Note that there is likely to be a practical limit of approximately 30K for the evfields length. ServiceCenter has a limit of 32K for an event record, and space must be reserved for additional fields and application messages. The server will quietly truncate evfields if it is too long.

The header data and the event fields may be separated by either a comma or a colon, as follows:

evtype,evtime,evsysseq,evuserseq,evsysopt,evuser,evpswd,evsepchar: evfields

-or-

evtype,evtime,evsysseq,evuserseq,evsysopt,evuser,evpswd,evsepchar, evfields

Example of an event record:

pmo,01/15/2000 22:15:48,,866065020,,BobHelpdesk,,^:logname^

netname^866065020^^Testing Problem Open^^^^^^^^^^^^^^^^^

Note that it is not necessary to supply a value for all fields, but if a field is omitted, a comma must be used as place holder in the header data, or the designated separater character in the evfields data.

Sample REXX Program

Perhaps the easiest way to implement an application for generating custom events, is by writing a REXX program to produce the event structure. A sample REXX program is provided in the CLIST library distributed with SCAuto for NetView, and is reprinted here for your convenience.

```
  /* CLIST TO TEST SCAUTO API, VIA SCANSEND */

EVTYPE   = 'pmo,'
EVTIME   = '07/28/1999 13:10:00,'
EVSYSSEQ = ','                    /* NOT USED FOR OPEN */
EVUSRSEQ = '8877665544,'
EVSYSOPT = ','                    /* RESERVED */
EVUSER   = 'NetOp1,'
EVPSWD   = ','
EVSEPCHAR = '^'
ENDHEADER = ':'                   /* LITERAL */
EVFIELDS  = 'Logname^Netname^123456789^^Testing problem open^'
EVFIELDS  = EVFIELDS || '^^^^^^^^^^^^^^^^^^'

EVENT = EVTYPE || EVTIME || EVSYSSEQ || EVUSRSEQ  || EVSYSOPT
EVENT = EVENT  || EVUSER || EVPSWD   || EVSEPCHAR || ENDHEADER
EVENT = EVENT  || EVFIELDS

address netvasis SCANSEND EVENT

EXIT
```

The event record is built by first defining the header fields. Note that a comma is used as a place holder for the fields for which no data is specified, EVSYSSEQ, EVSYSOPT, and EVPSWD. Next, the event fields, evfields, are built by assigning, and appending data fields, delimited by the character specified in EVSEPCHAR. Finally, all of the fields are concatenated as a single record, stored in one REXX variable.

The record is submitted to SCAuto by issuing the SCASEND command and supplying the record variable. Note particularly the command line:

```
address netvasis SCANSEND EVENT
```

The NetView/REXX command, *address netvasis*, retains the mixed case specified in the construction of the record. The default action for REXX, is to fold all text to uppercase. Refer to the NetView and REXX documentation for more information about writing NetView programs in REXX or other high level languages (HLLs).

Chapter 8 Inventory Automation

Overview

SCAuto for NetView provides support for Inventory Automation, to ensure that the ServiceCenter inventory database stays synchronized with the SNA network configuration, and locally attached system resources.

The Inventory Automation facilities within SCAuto, can be used to build the inventory database, with device names, parent devices, device types, serial numbers, model numbers and other information. Not only is accurate inventory available, but the current configuration and parent/child relationships of your devices is also maintained. Any changes to your configuration are reflected automatically.

Key personnel may be automatically notified of changes, eliminating guesswork or confusion over the system or network configuration. Additionally, ServiceCenter reports can be generated that indicate which devices have been added or removed, so an audit trail of configuration changes can be maintained.

Getting Started

Seeding the ServiceCenter inventory database is a one-time-only process, in which every resource in the SNA network, and every locally attached device is discovered. This potentially large set of data is transmitted to ServiceCenter, where the available information for each resource is added to the ServiceCenter database. SCAuto for NetView provides filtering of these data, through the SCANFLTR, to provide the ability to reject data that are not necessary for your configuration management requirements. The default filter distributed with SCAuto discards all terminal (TERM) and application (APPL) resources defined to VTAM. Refer the section *Filtering Inventory Data* on page 8-4 for more information.

The Sample option

During the initial installation and customizing of SCAuto for NetView, it was recommended that the *SAMPLE* option be used. This option limits discovery of the system and network resources. It will populate the ServiceCenter database with up to 20 records for each execution of the RFRESH command. (RFRESH is the command that is used to request discovery of resources, and is described fully in the following section.) Before seeding the ServiceCenter database with the entire system and network configuration, it is recommended that you test with the *SAMPLE* option to ensure that the inventory records are being populated in a manner consistent with your requirements.

To change the value of the SAMPLE option configuration parameter, edit the INVCNFG CLIST as follows:

To enable the *SAMPLE* option specify:

```
INV_SAMPLE='YES'
```

To disable the *SAMPLE* option specify:

```
INV_SAMPLE='NO'
```

Seeding the inventory database

To seed the inventory database:

1. Approximate the total number of SNA resources defined to VTAM. In multi-domain environments, use the largest number of resources defined to any single instance of VTAM.
2. Calculate the approximate number of resources that will be maintained in the ServiceCenter database by taking the total number of resources defined to VTAM and subtracting the number of resources that will be filtered in the SCANFLTR CLIST by resource type. For example, if you filter all TERMinals and all APPLications defined to VTAM, then the total number of resources maintained in ServiceCenter will be substantially smaller then the total resources defined to VTAM.
    a. If you have fewer than 5,000 resources that will be maintained by ServiceCenter, taking into account the resource types to be filtered out, then a RFRESH ALL ALL would gather all data in one domain and send it into ServiceCenter. Discovery of 5,000 resources takes approximately one hour to add to the ServiceCenter database, depending on machine size, CPU load, and priority of ServiceCenter.

b. If you have more than 5,000 resources, or you would rather build the ServiceCenter configuration database in increments, then use the RFRESH command to build the database by resource type. This allows you to discover resources for a given type, thereby reducing the number of records added to ServiceCenter at one time. This incremental building of the ServiceCenter database is done by executing the RFRESH command in NetView for various types of resources.

Sample Method:

The following is a method of incremental database building using resource types:

RFRESH NEW TERM
Refreshes the ServiceCenter database with all of the NEW terminals discovered in the network.

RFRESH NEW APPL
Refreshes the ServiceCenter database with all of the NEW applications discovered in the network.

RFRESH NEW DASD
Refreshes the ServiceCenter database with all of the NEW DASD units discovered on the system.

RFRESH NEW ALL
Refreshes the ServiceCenter database with ALL of the resources not previously discovered.

Important: Wait for the message: SC*A9326 SCANINVT*: *REFRESH COMPLETE, TASK IS TERMINATED*, before executing each new RFRESH command. This indicates that all the data has been sent to the logging task. The Event Monitor, SCEVMON, reads the inventory records from the *log* file, and forwards them to the Event Services input queue, in ServiceCenter.

Filtering Inventory Data

SCAuto for NetView provides the means to filter inventory data, through the standard filter CLIST, SCANFLTR. In this way, data collected for the system and network resources may be kept or discarded. The SCANFLTR CLIST should be customized to meet your specific configuration and inventory management needs. If you require access to all resources, the filter can be modified to allow all data to pass. More commonly, however, the data for certain resources is not critical to the inventory management process.

SCANFLTR can be used to filter data based on resource type, name, domain and/or description. The version of SCANFLTR distributed with SCAuto for NetView filters out, or discards, all resources that are terminals or VTAM application definitions, by default. These items typically account for the largest part of the resource data discovered through Inventory Automation, but are usually the least significant. Refer to the section *Filter Processing* in Chapter 3 for more information.

Inventory data is presented to the filter, SCANFLTR, in the following format:

Date/Time Date and time, *mm/dd/yy hh:mm:ss*, 17 bytes

Record Type *INVEN* literal, 5 bytes

Domain Domain, 8 bytes

Name Resource name, 8 bytes

Type Resource type, 4 bytes

Hierarchy: The complete resource hierarchy* in the following format:

data1 8 bytes

type1 4 bytes

data2 8 bytes

type2 4 bytes

data3 8 bytes

type3 4 bytes

data4 8 bytes

type4 4 bytes

data5 8 bytes

type5 4 bytes

*Contents of these fields vary by resource type

Refreshing the Inventory Database

The ServiceCenter inventory database is *refreshed* with new inventory data automatically or by operator command. The automatic refresh is done at user-defined intervals. This interval defines how often Inventory Automation will check for changes in the network and system configurations. An operator command, **rfresh**, is provided for manually invoking a variety of inventory refresh actions. SCAuto for NetView maintains the latest configuration information sent to ServiceCenter in *cache* so that redundant or duplicate information is not sent.

Automatic refresh interval

The ServiceCenter inventory database can be *refreshed* with new inventory data at any time. The automatic refresh interval is set in the SCANREFR CLIST member distributed with SCAuto for NetView. The default refresh time interval is every 12 hours. This interval defines how often Inventory Automation checks for changes in the network and system configurations. Once the inventory database has been initially *seeded*, only the configurations changes are sent to ServiceCenter—not the complete configuration.

To modify this default interval, uncomment (remove the * at the beginning of the line) and update the EVERY command in the SCANREFR CLIST member to the desired refresh interval by changing the default interval of 12 hours (0 12:00:00):

```
EVERY 0 12:00:00, ID=SCANREFR, RFRESH NEW ALL
```

where 0 12:00:00 indicates 0 days and 12 hours. This will refresh any new configuration changes every 12 hours. This refresh interval can be changed to accommodate your needs. If you need more immediate notification of changes, set a smaller interval. If CPU resource utilization is critical, you can increase the interval, for example, to every three days.

Operator RFRESH command

The ServiceCenter inventory database can be *refreshed* with new inventory data at any time. This gives the user control of when a refresh is done and how often. Inventory refresh is activated by entering the RFRESH command from within the NetView environment. The syntax of the RFRESH command is as follows:

RFRESH request type resource type [delete hours]

where:

request type	ALL	Refresh resources ignoring entries in the cache file
	NEW	Refresh only resources not already in cache
	DEL	Delete resources
resource type -	ALL	Refresh all resource types
	type	Refresh only by specific, predefined resource type
delete hours	number of hours of a resource must have been inactive before deleting it (optional)	

Examples of using the RFRESH command:

RFRESH NEW COMC

This command refreshes the ServiceCenter inventory database for all resources that are Communications Controllers (COMC (NCP)) not currently in the ServiceCenter database.

RFRESH ALL LINE

This command refreshes the ServiceCenter inventory database for all resources that are SNA lines (LINE) regardless of whether or not they were previously in the ServiceCenter database.

RFRESH NEW ALL

This command refreshes the ServiceCenter inventory database for all new resources that were not previously in the ServiceCenter database.

RFRESH DEL ALL 168

This command sends an inventory record deleted message to the ServiceCenter inventory database for all resources that have not been active within the last week (168 hours).

Resource types

The following is a list of the valid resource types that are discovered and maintained by the Inventory Automation feature of SCAuto for NetView:

COMC	Communication Controller/NCP
LINE	SNA line
CTRL	Cluster Controller
TERM	Terminal
LDEV	Local Attached Terminal
LMAJ	Local Attached Major Node Definition
LCTL	Local Attached Cluster Controller
LOCL	All Local Attached Devices
APPL	Application
AMAJ	Application Major Node
CDRM	Cross Domain Resource Manager
CDRS	Cross Domain Resource
SMAJ	Switched Major Node definition
LGRP	Line Group Definition
CLST	Cluster Controller
LKST	Link Station
CHAN	S/390 Channel
CPU	Host CPU
DASD	Local attached DASD
TAPE	Local attached tape drive
UREC	Local attached unit record equipment
NETW	VTAM network name
VTAM	All VTAM network devices

Inventory Data and ServiceCenter

Inventory data, that passes filtering, is transferred by the Event Monitor to ServiceCenter's *eventin* queue, in Event Services, as standard icma, icmu, and icmd events for updating the ServiceCenter Inventory Database.

The standard event formats and mapping services are described in the ***ServiceCenter Event Services Utility*** manual.

The inventory data record fields may be populated are listed below. Actual contents and source fields vary by resource type.

logical.name	(resource name)
vendor	
parent	
model	
network.name	(resource name)
serial.no.	(serial number)
type	(resource type)
last.update	(date/time)
description	
network.address	
domain	
protocol	
protocol.addr	
operating.system	

Chapter 9 Problem Determination

Diagnostic Measures

If an event processed by SCAuto for NetView, fails to reach the event-in queue in ServiceCenter, there are a number of checks that can be performed to isolate the problem:

- Check NetView status
- Ensure that all SCAuto for NetView tasks are active
- Check for SCAuto error messages
- Verify that the event was created
- Check the network connection to the ServiceCenter system
- Check the ServiceCenter system
- Call Peregrine Customer Support to help resolve your problem, if needed

Check NetView status

Check the system console log, and the active NetView log for any error messages or indication of abnormal conditions. Verify that the Sub-System Interface (SSI) task is active.

Ensure that all SCAuto for NetView tasks are active

The SCAuto tasks that run under NetView control can be displayed with the SCAuto supplied command, SCANVDIS. Alternatively, NetView commands may be used. For example, to display all NetView tasks, enter:

```
LIST STATUS=TASKS
```

The SCAuto for NetView tasks are:

SCANTASK	Main SCAuto task
SCARTTASK	Message routing
SCANSMFT	SMF information processing
SCANVELT	Event logging
SCANINVT	Inventory (optional)
SCANAUTO	Autotask

SCAuto tasks may be stopped and started collectively, with the SCANSTOP and SCANSTRT commands respectively (refer to Chapter 3). To restart an individual task, use the NetView command, START. Use the AUTOTASK command for restarting the automated operator, SCANAUTO.

The event monitor, SCEVMON, runs as an MVS started task, and may be restarted with the MVS start command, for example:

```
S SCEVMON
```

Check for SCAuto error messages

Both the event logging task, and the event monitor use the SCAuto log for recording messages. The *log* file is located in the hierarchical file system, and has the following name and location, by default:

```
/etc/scauto/scanv390/v1.0/run/scanv390.log
```

Verify that the event was created

Events to be forwarded to ServiceCenter are first recorded in the events-to log. This file is also located on the hierarchical file system, in the same directory as the message log. The file name is:

```
scevents.to.hostid.service
```

Where *hostid* is the IP address or host name (if using DNS) of the service center system, and *service* is the port address of the server, for example: *scserver.12690*.

The file may be viewed in a text editor, or displayed in a terminal shell session, with the UNIX *cat* command. If the expected event is not found in the file, it is possible that the data are being discarded based on filtering criteria. Review the SCAuto filters in NetView, SCANFLTR and the alert filter, FILTER.

Check the network connection to the ServiceCenter system

Verify that there is connectivity between the ServiceCenter system, and the system on which SCAuto for NetView is running, by issuing the *oping*, from the shell. This TCPIP command sends an echo request to the remote host. If this command fails, contact the network administrator for assistance.

If command succeeds, verify that the IP address or host name (if using DNS), and the port address are correctly specified in the initialization file. The default location and file name of this file is:

```
/etc/scauto/scanv390/v1.0/run/scanv390.ini
```

Check the ServiceCenter system

Ensure that ServiceCenter is running on the remote host, and that the Event Scheduler has been started. Refer to ServiceCenter documentation appropriate for the server platform.

Contact Peregrine Customer Support

If you need assistance solving your problem, you can contact Peregrine Customer Support. Please have all relevant files, logs, and initialization parameter values at hand before calling.

Error Messages

SCA0001: MQS TO SCANAUTO FAILED

Explanation: An attempt was made to send a message to the autotask, SCANAUTO, via the NetView DSIMQS macro, and the request failed.

Probable cause: The designated receiving task has stopped.

Result: SCAuto for NetView initialization failure.

SCA0002: UNABLE TO GET STORAGE - DSIGET

Explanation: An attempt was made to obtain storage via the NetView DSIGET macro failed.

Probable cause: Internal error in module, or global storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA0003: INITIALIZATION OF SCANV390 COMPLETE

Explanation: Initialization of SCAuto for Netview has completed successfully.

Result: Processing continues.

SCA0005: UNABLE TO LOCATE SCANTASK TASK. INIT FAILURE

Explanation: Initialization of SCAuto for Netview has failed.

Probable cause: Internal error in module.

Result: SCANAUTO autotask will be deactivated.

SCA0101 UNABLE TO GET STORAGE - DSIGET

Explanation: An attempt was made to obtain storage via the NetView DSIGET macro failed.

Probable cause: Internal error in SCANVXLP/T, or global storage problem.

Result: DSIELTSK will be deactivated.

SCA0102 UNABLE TO GET WORK BLOCK - DSILCS

Explanation: The module requested an SWB control block via the NetView macro DSILCS, and the request failed.

Probable cause: NetView internal error or global storage problem.

Result: DSIELTSK will be deactivated.

SCA0103 UNABLE TO FREE WORK BLOCK - DSILCS

Explanation: Module attempted to free an SWB control block via the NetView macro DSILCS and the request failed.

Probable Cause: Internal error in SCANVXLP/T or global storage problem.

Result: DSIELTSK will be de-activated.

SCA0104 UNABLE TO SEND MESSAGE TO SCANTASK - DSIMQS

Explanation: The module attempted to send a message to SCANTASK via the NetView macro DSIMQS, and the request failed.

Probable cause: The task, SCANTASK, has not been started, or has been stopped. Restart SCANTASK if it has failed to initialize and then restart DSIELTSK.

Result: DSIELTSK will be de-activated.

SCA0201: NO DSRB PROVIDED

Explanation: Module was invoked by NetView without a pointer to the DSRB control block.

Probable cause: Logic error in NetView.

Result: SCANTASK will not be started for message or command processing.

SCA0202: NO USER MESSAGE PROVIDED WITH DSRB

Explanation: Module was invoked by NetView with a DSRB that did not point to a valid message to be processed.

Probable cause: Logic error in NetView.

Result: SCANTASK will be de-activated.

SCA0203: INITIALIZATION OF DST FAILED

Explanation: Task initialization for SCANTASK has not been executed or has run unsuccessfully.

Probable cause: Task initialization parameters may be in error. Check the NetView log for additional messages.

Result: SCANTASK will not be started for message or command processing. Restart SCANTASK.

SCA0204: INVALID DSRB CODE

Explanation: SCANTASK was driven with a function code other than *new message*.

Probable cause: Logic error in NetView.

Result: SCANTASK will be de-activated.

SCA0205: UNABLE TO GET STORAGE - DSIGET

Explanation: An attempt to obtain storage via the NetView DSIGET macro failed.

Probable cause: Global NetView storage problem.

Result: SCANTASK will be deactivated.

SCAN0206: MQS TO SCANAUTO FAILED

Explanation: An attempt was made to send a message to the autotask, SCANAUTO, via the NetView DSIMQS macro, and the request failed.

Probable cause: The receiving task has not been started or has been stopped.

Result: SCANTASK will be de-activated.

SCA0301 UNABLE TO GET STORAGE - DSIGET

Explanation: An attempt to obtain storage via the NetView DSIGET macro failed.

Probable cause: Global NetView storage problem.

Result: SCANTASK will be deactivated.

SCA0304 INVALID DATA - INVALID OPTION

Explanation: An invalid option was specified in the DSIPARM initialization parameter member, SCANINIT.

Probable cause: Invalid or missing parameter.

Result: Initialization process for SCAuto for Netview is terminated.

SCA0401: MQS FAILED

Explanation: An attempt was made to obtain storage via the NetView DSIGET macro failed.

Probable cause: Global NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA0402: DSIGET FAILED

Explanation: An attempt to obtain storage via the NetView DSIGET macro failed.

Probable cause: Global NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA0403: SCANV390 FILTER WAS NOT PREVIOUSLY LOADED

Explanation: No Alert filter has been activated for SCAuto for NetView.

Probable cause: Possible initialization problems, including missing or invalid filter.

Result: Processing continues. The filter may be loaded with INITFLTR.

SCA0404: BAD ADDRESS FOR THRESHOLD DISPLAY

Explanation: An invalid vector address was passed in TVBUFLD.

Probable cause: Internal processing error.

Result: SCANAUTO autotask is terminated.

SCA0405: NO ACCUMULATED THRESHOLD DATA

Explanation: All threshold counts are currently zero.

Result: Processing continues

SCA0406: BAD ADDRESS FOR DSIFRE

Explanation: An invalid vector address was detected in the TVBUFLD address vector.

Probable cause: Internal processing error.

Result: SCANAUTO autotask is terminated.

SCA0501: UNABLE TO GET STORAGE - DSIGET

Explanation: An attempt to obtain storage via the NetView DSIGET macro failed.

Probable cause: Global NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA0502: UNABLE TO FREE STORAGE - DSIFREE

Explanation: An attempt to release storage via the NetView DSIFREE macro failed.

Probable cause: Internal processing error, or other NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA0601: DSILCS FOR SWB FAILED.

Explanation: An attempt to obtain an SWB control block via the NetView DSILCS macro failed.

Probable cause: Internal error in SCAN0506 or global NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA0602: DSILCS FOR CWB FAILED

Explanation: An attempt to obtain an CWB control block via the NetView DSILCS macro failed.

Probable cause: Internal error in SCAN0506 or global NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA0603: DSICES FAILED

Explanation: An attempt to obtain the address of the SCANFLTR CLIST, via the NetView DSICES macro failed.

Probable cause: Internal error in SCAN0506 or CLIST SCANFLTR was not found.

Result: SCANAUTO autotask will be deactivated.

SCA0604: DSIFRE FOR PDB FAILED

Explanation: An attempt to release a PDB control block via the NetView DSIFRE macro failed.

Probable cause: Internal error in SCAN0506 or global NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA0605: DSIFRE FOR SWB FAILED

Explanation: An attempt to release an SWB control block via the NetView DSILCS macro failed.

Probable cause: Internal error in SCAN0506 or global NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA0606: DSIPRS PDBSIZE FAILED

Explanation: An attempt to calculate the size of a PDB control block via the NetView DSILPRS macro failed.

Probable cause: Logic error in NetView.

Result: SCANAUTO autotask will be deactivated.

SCA0607: DSILCS FREE FOR CWB FAILED

Explanation: An attempt to release a CWB control block via the NetView DSILCS macro failed.

Probable cause: Internal error in SCAN0506 or global NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA0608: DSIGET FOR PDB FAILED

Explanation: An attempt to obtain a PDB control block via the NetView DSIGET macro failed.

Probable cause: Internal error in SCAN0506 or global NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA0609: DSIPRS FAILED

Explanation: An attempt to parse PDB control block data via the NetView DSIGET macro failed.

Probable cause: Internal error in SCAN0506 or logic error in NetView.

Result: SCANAUTO autotask will be deactivated.

SCA0610: DSIFRE FOR WORKAREA FAILED

Explanation: An attempt to release a workarea, via the NetView macro, DSIFRE, failed.

Probable cause: Internal error in SCAN0506 or global NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA0611: DSIGET FOR MESSAGE FAILED

Explanation: An attempt to obtain a message buffer via the via the NetView macro, DSIGET, failed.

Probable cause: Logic error in NetView.

Result: SCANAUTO autotask will be deactivated.

SCA0612: DSIGET FOR WORKAREA FAILED

Explanation: An attempt to obtain a workarea, via the NetView macro, DSIGET, failed.

Probable cause: Internal error in SCAN0506 or global NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA0613: CLIST INVOCATION FAILED

Explanation: Module attempted to call CLIST SCANFLTR, and the attempt failed.

Probable cause: Internal error in SCAN0506 or internal NetView error.

Result: SCANAUTO autotask will be deactivated.

SCA0614: DSIFRE FOR INTERTASK STORAGE FAILED

Explanation: An attempt to release a message buffer for the last alert received, via the via the NetView macro, DSIFRE, failed.

Probable cause: Internal error in SCAN0506 or global NetView storage problem..

Result: SCANAUTO autotask will be deactivated.

SCA0615: TVBUFLD NOT INITIALIZED

Explanation: The TVBUFLD contained an invalid vector address.

Probable cause: Internal error control error.

Result: SCANAUTO autotask will be deactivated.

SCA0801: UNABLE TO GET STORAGE - DSIGET

Explanation: An attempt to obtain storage via the NetView DSIGET macro failed.

Probable cause: Global NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA0802: MQS TO DST FAILED

Explanation: An attempt to send a message to another task via the NetView macro DSIMQS, failed.

Probable cause: The SCANTASK task has not been started or has stopped. Restart SCANTASK if it has failed to initialize.

Result: SCANAUTO autotask will be deactivated.

SCA0803: MODULE INVOKED WITHOUT VALID TVBUFLD VECTOR

Explanation: Module was invoked by the SCANFLTR CLIST, and the TVBUFLD field contained an invalid vector address for the Alert buffer.

Probable cause: SCANFLTR CLIST invoked SCAN0507 twice while processing only one alert.

Result: Warning message only. Processing continues.

SCA0804: UNABLE TO FREE STORAGE - DSIFRE

Explanation: An attempt to RELEASE storage via the NetView DSIFRE macro failed.

Probable cause: Internal processing error or global NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA0805: MODULE NOT RUN UNDER AUTOTASK SCANAUTO

Explanation: Module was invoked in an environment other than under SCANAUTO autotask of SCAuto for NetView.

Probable cause: User invocation of this module from another environment.

Result: No processing takes place.

SCA0806: TVBUFLD NOT INITIALIZED

Explanation: The TVBUFLD field contained an invalid vector address.

Probable cause: Internal control error.

Result: SCANAUTO autotask will be deactivated.

SCA0901: NO DSRB PROVIDED

Explanation: Module was invoked by NetView without a pointer to the DSRB control block.

Probable cause: Logic error in NetView.

Result: SCANTASK will not be started for message or command processing.

SCA0902: NO USER MESSAGE PROVIDED WITH DSRB

Explanation: Module was invoked by NetView with a DSRB that did not point to a valid message to be processed.

Probable cause: Logic error in NetView.

Result: SCANTASK will be de-activated.

SCA0903: INITIALIZATION OF DST FAILED

Explanation: Task initialization for SCANTASK has not been executed or has run unsuccessfully.

Probable cause: Task initialization parameters may be in error. Check the NetView log for additional messages.

Result: SCANTASK will not be started for message or command processing. Restart SCANTASK.

SCA0904: INVALID DSRB CODE

Explanation: SCANTASK was driven with a function code other than *new message*.

Probable cause: Logic error in NetView.

Result: SCANTASK will be de-activated.

SCA0905: UNABLE TO GET STORAGE - DSIGET

Explanation: An attempt to obtain storage via the NetView DSIGET macro failed.

Probable cause: Global NetView storage problem.

Result: SCANTASK will be deactivated.

SCAN0207: MQS TO SCANAUTO FAILED

Explanation: An attempt was made to send a message to the autotask, SCANAUTO, via the NetView DSIMQS macro, and the request failed.

Probable cause: The receiving task has not been started or has been stopped.

Result: SCANTASK will be de-activated.

SCAN15021: ALERT FILTER CAN ONLY BE RUN IN SCANAUTO TASK

Explanation: An attempt was made to execute filter processing modules in a task other than SCANAUTO autotask of NetView.

Probable cause: Possible user error.

Result: The filter processing request is rejected. Processing continues.

SCA1503: UNABLE TO GET STORAGE - DSIGET

Explanation: An attempt to obtain storage for filter processing, via the NetView DSIGET macro failed.

Probable cause: Global NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA1504: UNABLE TO ACCESS DSIPARM

Explanation: Unable to access the DSIPARM dataset.

Probable cause: Incorrect JCL specification in NetView start-up procedure.

Result: SCANAUTO autotask will be deactivated.

SCA1505: FILTER MEMBER DOES NOT EXIST OR IS EMPTY

Explanation: The FILTER member of the DSIPARM dataset, which normally contains the Alert filter, is empty or does not exist.

Probable cause: Member name is misspelled, member does not exist, or does not contain minimally a single record entry.

Result: SCANAUTO autotask will be deactivated.

SCA1506: PERMINENT I/O ERROR READING FILTER

Explanation: An I/O error was encountered while attempting to read the Alert Filter from the DSIPARM dataset.

Probable cause: Physical DASD device error.

Result: SCANAUTO autotask will be deactivated.

SCA1507: DSIPARM IS NOT OPEN

Explanation: The DSIPARM dataset is not open, or does not exist.

Probable cause: Incorrect JCL specification in NetView start-up procedure.

Result: SCANAUTO autotask will be deactivated.

SCA1508: INTERNAL LOGIC ERROR

Explanation: Invalid NetView control block encountered while accessing DSIPARM dataset.

Probable cause: Internal error in SCAuto or NetView.

Result: SCANAUTO autotask will be deactivated.

SCA1509: UNABLE TO DISCONNECT FROM DSIPARM

Explanation: An error occurred in the NetView macro processing while attempting to disconnect from the Alert filter in the DSIPARM dataset.

Probable cause: Physical DASD device problem.

Result: SCANAUTO autotask will be deactivated.

SCA1601: INVALID MAIN VECTOR ADDRESS

Explanation: The TVBUFLD contained an invalid vector address.

Probable cause: Internal control error.

Result: SCANAUTO autotask will be deactivated.

SCA1602: INVALID FILTER MAP HEADER

Explanation: The internal Alert filter map header contained invalid data.

Probable cause: Internal control error.

Result: SCANAUTO autotask will be deactivated.

SCA1701: NO SPACE AVAILABLE FOR NEW DOMAIN

Explanation: The number of different domains using threshold processing exceeded 100.

Probable cause: More than 100 domains in the network.

Result: The alert being processed will be unconditionally passed through the filter.

SCA1702: ERROR IN SCAN0506

Explanation: An error was detected in module SCAN0506.

Probable cause: Internal control error, or other error indicated in network logs.

Result: SCANAUTO autotask will be deactivated.

SCA1703: DSIGET FOR THRESHOLD BUFFER FAILED

Explanation: An attempt to obtain storage for the threshold buffer, via DSIGET, failed.

Probable cause: Global Netview storage problem.

Result: SCANAUTO autotask will be deactivated.

SCAN1801: *EDIT* xxxxxxxx LINE NO nnnn

Explanation: An error was detected in the Alert filter, where xxxxxxxx is the error description, and nnnnis the line number in which the error was found.

Probable cause: Invalid record entered in DSIPARM member, FILTER.

Result: The invalid record will be rejected and bypassed.

SCA1802 UNABLE TO GET STORAGE - DSIGET

Explanation: An attempt to obtain storage for filter processing, via the NetView DSIGET macro, failed.

Probable cause: Global NetView storage problem.

Result: SCANTASK will be deactivated.

SCA1901: MQS FAILED

Explanation: An attempt to obtain storage via the NetView DSIGET macro, for displaying the Alert filter, failed.

Probable cause: Global NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA1902: INVALID FILTER MAP ADDRESS

Explanation: The TVBUFLD passed to this module does not contain a valid address.

Probable cause: NetView logic error.

Result: SCANAUTO autotask will be deactivated.

SCA1903: INVALID FILTER MAP HEADER

Explanation: The TVBUFLD passed to this module does point to a valid filter header.

Probable cause: Internal processing error.

Result: SCANAUTO autotask will be deactivated.

SCA1904: DSIGET FAILED

Explanation: An attempt to obtain storage via the NetView DSIGET macro, for displaying the Alert filter, failed.

Probable cause: Global NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA1905: SCANV390 FILTER WAS NOT PREVIOUSLY LOADED

Explanation: No Alert filter has been activated for SCAuto for NetView.

Probable cause: Possible initialization problems, including missing or invalid filter.

Result: Processing continues. The filter may be loaded with INITFLTR.

SCA2501: NO HIERARCHY VALUES WERE SPECIFIED

Explanation: None of the possible five hierarchy name/type pairs was assigned a value.

Probable cause: Error in user defined CLIST that invokes GENPROB.

Result: The input data is discarded. GENPROB terminates with completion code 4.

SCA2502: INVALID COMBINATION OF HIERARCHY VALUES SPECIFIED

Explanation: An error was detected in the specification of a hierarchy name/type pair. Either a name was specified without a type, or a type was specified without a name.

Probable cause: Error in user defined CLIST that invokes GENPROB.

Result: The input data is discarded. GENPROB terminates with completion code 8.

SCA2503: NO PROBLEM DESCRIPTION WAS SPECIFIED

Explanation: The problem description field was not assigned a value, but is a required field.

Probable cause: Error in user defined CLIST that invokes GENPROB.

Result: The input data is discarded. GENPROB terminates with completion code 12.

SCA2504: INVALID COMBINATION OF HIERARCHY VALUES SPECIFIED

Explanation: A gap was found in the specification of the possible five hierarchy name/type pairs. All hierarchy pairs must be assigned contiguously from pair 1 upwards.

Probable cause: Error in user defined CLIST that invokes GENPROB.

Result: The input data is discarded. GENPROB terminates with completion code 16.

SCA2505: A SETUP RECORD TYPE OF xxxxx CANNOT BE USED

Explanation: The problem type field contained an invalid type, of either ALERT, ABEND, or INVEN, which are processed by other modules.

Probable cause: Error in user defined CLIST that invokes GENPROB.

Result: The input data is discarded. GENPROB terminates with completion code 20.

SCA2506: INVALID RECORD SETUP TYPE USED: xxxxxxxx

Explanation: The problem type field contained an unknown, invalid type.

Probable cause: Error in user defined CLIST that invokes GENPROB.

Result: The input data is discarded. GENPROB terminates with completion code 20.

SCA2601: UNABLE TO GET STORAGE - DSIGET

Explanation: An attempt to obtain storage via the NetView DSIGET macro failed.

Probable cause: Global NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA2602: MQS FAILED

Explanation: An attempt to send a message to another task via the NetView macro DSIMQS, failed.

Probable cause: The SCANTASK task has not been started or has stopped. Restart SCANTASK if it has failed to initialize.

Result: SCANAUTO autotask will be deactivated.

SCA2603: HIERARCHY PAIR SPECIFIED NOT COMPLETE

Explanation: One of five possible hierarchy pairs specified a name without a type, or a type without a name.

Probable cause: Probable error in user CLIST.

Result: Message will be discarded. No problem will be opened.

SCA2604: MODULE NOT INVOKED BY GENPROB CLIST

Explanation: Module was invoked in a context other than by the GENPROB CLIST.

Probable cause: User invocation of this module from another environment.

Result: No processing takes place.

SCA2701: INVALID NAME GIVEN TO UPDGENER: xxxxxxxx

Explanation: The abend name assigned to UGPMNAME global variable is invalid, or the variable has no value.

Probable cause: User error in calling CLIST.

Result: No abend update event is created.

SCA2702: INVALID TYPE GIVEN TO UPDGENER: xxxxxxxx

Explanation: The abend type assigned to UGPMTYPE global variable is invalid, or the variable has no value.

Probable cause: User error in calling CLIST.

Result: No abend update event is created.

SCA2703: INVALID CLOSE DATA GIVEN TO UPDGENER: xxxxxxxx

Explanation: The UGPMDATA global variable has not been assigned a valid value.

Probable cause: User error in calling CLIST.

Result: No abend update event is created.

SCA2704: INVALID SETUP RECORD TYPE GIVEN TO UPDGENER: xxxxxxxx

Explanation: The UGPMRECT global variable has been assigned an invalid value.

Probable cause: User error in calling CLIST.

Result: No abend update event is created.

SCA3601: UNABLE TO GET STORAGE - DSIGET

Explanation: An attempt to obtain storage via the NetView DSIGET macro failed.

Probable cause: Global NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA3602: MQS FAILED

Explanation: An attempt to send a message to another task via the NetView macro DSIMQS, failed.

Probable cause: The SCANTASK task has not been started or has stopped. Restart SCANTASK if it has failed to initialize.

Result: SCANAUTO autotask will be deactivated.

SCA3603: INVALID MESSAGE TYPE RECEIVED

Explanation: An unexpected or unidentified message has been received from another task via the NetView macro DSIMQS.

Probable cause: Internal processing error, or invocation of SCAuto modules by another task.

Result: Message will be discarded. Processing continues.

SCA3604: MODULE NOT RUN UNDER AUTOTASK - SCANAUTO

Explanation: Module was invoked in an environment other than under SCANAUTO autotask of SCAuto for NetView.

Probable cause: User invocation of this module from another environment.

Result: No processing takes place.

SCA3701: NO DSRB PROVIDED

Explanation: Module was invoked by NetView without a pointer to the DSRB control block.

Probable cause: Logic error in NetView.

Result: SCANTASK will not be started for message or command processing.

SCA3702: NO USER MESSAGE PROVIDED WITH DSRB

Explanation: Module was invoked by NetView with a DSRB that did not point to a valid message to be processed.

Probable cause: Logic error in NetView.

Result: SCANTASK will be de-activated.

SCA3703: TVBUFLD INVALID

Explanation: The TVBUFLD contained an invalid vector address.

Probable cause: Internal control error.

Result: SCANTASK will not be started for message or command processing.

SCA3704: INIT FAILED, NO BUFFER ALLOCATED

Explanation: Task initialization for SCANTASK has not been executed or has run unsuccessfully.

Probable cause: Task initialization parameters may be in error. Check the NetView log for additional messages.

Result: SCANTASK will not be started for message or command processing. Restart SCANTASK.

SCA3705: INVALID DSRB CODE

Explanation: Module was invoked by NetView without a pointer to a valid DSRB control block.

Probable cause: Logic error in NetView.

Result: SCANTASK will not be started for message or command processing.

SCA3706: UNABLE TO GET STORAGE - DSIGET

Explanation: An attempt to obtain storage via the NetView DSIGET macro failed.

Probable cause: Global NetView storage problem.

Result: SCANTASK will be deactivated.

SCA3707: MQS FAILED

Explanation: An attempt was made to send a message via the NetView DSIMQS macro, and the request failed.

Probable cause: The receiving task has not been started or has been stopped.

Result: SCANTASK will be de-activated.

SCA3708: MAX STEP ABEND MESSAGES ON QUEUE EXCEEDED

Explanation: An attempt add a step abend record to the queue has failed, because the maximum queue length has been reach.

Result: No new step abends will be collected until the number of jobstep-end records on the queue is reduced at job termination.

SCA3801: INVALID NAME GIVEN TO UPDABEND: xxxxxxxx

Explanation: The abend name assigned to UABENDNAME global variable is invalid, or the variable has no value.

Probable cause: User error in calling CLIST.

Result: No abend update event is created.

SCA3802: INVALID TYPE GIVEN TO UPDABEND: xxxxxxxx

Explanation: The abend type assigned to UABENDTYPE global variable is invalid, or the variable has no value.

Probable cause: User error in calling CLIST.

Result: No abend update event is created.

SCA3803: INVALID CLOSE DATA GIVEN TO UPDABEND: xxxxxxxx

Explanation: The UABENDDATA global variable has not been assigned a value.

Probable cause: User error in calling CLIST.

Result: No abend update event is created.

SCA4601: UNABLE TO GET STORAGE - DSIGET

Explanation: An attempt to obtain storage via the NetView DSIGET macro failed.

Probable cause: Global NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA4602: MQS FAILED

Explanation: An attempt to send a message to another task via the NetView macro DSIMQS, failed.

Probable cause: The SCANTASK task has not been started or has stopped. Restart SCANTASK if it has failed to initialize.

Result: SCANAUTO autotask will be deactivated.

SCA4603: MODULE NOT INVOKED BY INVVPD CLIST

Explanation: Module was invoked in a context other than by the INVVPD CLIST.

Probable cause: User invocation of this module from another environment.

Result: No processing takes place.

SCA5001: ERROR ENCOUNTERED OR TIMEOUT IN WAIT

Explanation: A timeout or other error occurred while waiting for a VTAM message.

Probable cause: High system activity, or timeout value was set too low.

Result: No data is collected.

SCA5002: ERROR ENCOUNTERED IN PROCESSING MESSAGES

Explanation: A non-zero return code was encountered while retrieving a VTAM message.

Probable cause: Internal processing error.

Result: Processing is discontinued.

SCA5003: RCVD AN INVALID VTAM LINE OR PU NAME

Explanation: An unrecognized or invalid value was encountered while processing a VTAM message.

Probable cause: Internal processing error.

Result: Processing is discontinued.

SCA5101: ERROR ENCOUNTERED OR TIMEOUT IN WAIT

Explanation: A timeout or other error occurred while waiting for a VTAM message.

Probable cause: High system activity, or timeout value was set too low.

Result: No data is collected.

SCA5102: ERROR ENCOUNTERED IN PROCESSING MESSAGES

Explanation: A non-zero return code was encountered while retrieving a VTAM message.

Probable cause: Internal processing error.

Result: Processing is discontinued. Return code = 99.

SCA5201: ERROR ENCOUNTERED IN TIMEOUT OR WAIT

Explanation: A timeout or other error occurred while waiting for a VTAM message.

Probable cause: High system activity, or timeout value was set too low.

Result: Processing is discontinued.

SCA5202: ERROR ENCOUNTERED IN PROCESSING MESSAGES

Explanation: A non-zero return code was encountered while retrieving a VTAM message.

Probable cause: NetView internal processing error.

Result: Processing is discontinued.

SCA5203: VALID VTAM MAJNODE NAME NOT FOUND

Explanation: The major node name was invalid or not found in the message.

Probable cause: Internal processing error.

Result: Processing is discontinued.

SCA5301: UNABLE TO GET STORAGE - DSIGET

Explanation: An attempt to obtain storage via the NetView DSIGET macro failed.

Probable cause: Global NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA5302: MQS TO DST FAILED

Explanation: An attempt to send a message to another task via the NetView macro DSIMQS, failed.

Probable cause: The SCANTASK task has not been started or has stopped. Restart SCANTASK if it has failed to initialize.

Result: SCANAUTO autotask will be deactivated.

SCA5303: MODULE NOT INVOKED BY SCANHIER CLIST

Explanation: Module was invoked in a context other than by the SCANHIER CLIST.

Probable cause: User invocation of this module from another environment.

Result: No processing takes place.

SCA5401: ERROR ENCOUNTERED IN TIMEOUT OR WAIT

Explanation: A timeout or other error occurred while waiting for a message.

Probable cause: High system activity, or timeout value was set too low.

Result: Processing is discontinued.

SCA5402: ERROR ENCOUNTERED IN PROCESSING MESSAGES

Explanation: A non-zero return code was encountered while retrieving a message.

Probable cause: NetView internal processing error.

Result: Processing is discontinued.

SCA5501: UNABLE TO GET STORAGE - DSIGET

Explanation: An attempt to obtain storage via the NetView DSIGET macro failed.

Probable cause: Global NetView storage problem.

Result: SCANAUTO autotask will be deactivated.

SCA5502: MQS TO DST FAILED

Explanation: An attempt to send a message to another task via the NetView macro DSIMQS, failed.

Probable cause: The SCANTASK task has not been started or has stopped. Restart SCANTASK if it has failed to initialize.

Result: SCANAUTO autotask will be deactivated.

SCA5503: HIERARCHY PAIR SPECIFIED NOT COMPLETE

Explanation: One of five possible hierarchy pairs specified a name without a type, or a type without a name.

Probable cause: Probable error in user CLIST.

Result: Message will be discarded. No problem will be opened.

SCA5504: MODULE NOT INVOKED BY UPDATE CLIST

Explanation: Module was invoked in a context other than by one of the update CLISTs, UPDALERT, UPDABEND, or UPDGENER.

Probable cause: User invocation of this module from another context.

Result: No processing takes place.

SCA5601: INVALID NAME GIVEN TO UPDALERT: xxxxxxxx

Explanation: The abend name assigned to UALERTNAME global variable is invalid, or the variable has no value.

Probable cause: User error in calling CLIST.

Result: No abend update event is created.

SCA5602: INVALID TYPE GIVEN TO UPDALERT: xxxxxxxx

Explanation: The abend type assigned to UALERTTYPE global variable is invalid, or the variable has no value.

Probable cause: User error in calling CLIST.

Result: No abend update event is created.

SCA5603: INVALID CLOSE DATA GIVEN TO UPDALERT: xxxxxxxx

Explanation: The UALERTDATA global variable has not been assigned a valid value.

Probable cause: User error in calling CLIST.

Result: No abend update event is created.

SCA5701: ERROR ENCOUNTERED IN TIMEOUT OR WAIT

Explanation: A timeout or other error occurred while waiting for a VTAM message.

Probable cause: High system activity, or timeout value was set too low.

Result: No data is collected for the resource.

SCA5702: ERROR ENCOUNTERED IN PROCESSING MESSAGES

Explanation: A non-zero return code was encountered while retrieving a VTAM message.

Probable cause: Internal processing error.

Result: Data collection for the resource is discontinued.

SCA8901: MQS TO TASK FAILED (SCANTASK/SCAN0100)

Explanation: An attempt was made to send a message to SCANTASK, via the NetView DSIMQS macro, and the request failed.

Probable cause: The receiving task has not been started or has been stopped.

Result: SCANINVT will be de-activated.

SCA8902: UNABLE TO GET STORAGE - DSIGET

Explanation: An attempt to obtain storage via the NetView DSIGET macro failed.

Probable cause: Global NetView storage problem.

Result: SCANTASK will be deactivated.

SCA9201: MQS TO TASK FAILED (SCANTASK/SCAN1002)

Explanation: An attempt was made to send a message to SCANTASK, via the NetView DSIMQS macro, and the request failed.

Probable cause: The receiving task has not been started or has been stopped.

Result: SCANINVT will be de-activated.

SCA9202 UNABLE TO GET STORAGE - DSIGET

Explanation: An attempt to obtain storage via the NetView DSIGET macro failed.

Probable cause: Global NetView storage problem.

Result: SCANTASK will be deactivated.

SCA9301 INITIALIZATION FAILED (INITAIM)

Explanation: A non-zero return code was detected during Inventory initialization.

Probable cause: Internal control error.

Result: Inventory processing is suspended.

SCA9302 SHUTDOWN FAILED (FREEAIM)

Explanation: A non-zero return code was detected in post-processing, after completion of SNA inventory.

Probable cause: Internal control error.

Result: The inventory process terminates normally.

SCA9303 MAX INVENTORY RECORDS - NO MORE CAN BE ADDED

Explanation: The maximum number of inventory records stored in cache has been reached.

Probable cause: More than 100,00 resources discovered.

Result: Any resources not already in cache will be detected as new resources in subsequent executions of RFRESH.

SCA9304 AIMTASK SHUTDOWN DURING REFRESH

Explanation: A command to shutdown SCAuto for NetView was received during inventory processing.

Probable cause: User command.

Result: Inventory processing is halted, and shutdown proceeds.

SCA9305 NUMBER OF HOURS IS INVALID - DEFAULT 2 WEEKS

Explanation: The number of hours specified for deletion of old resources is invalid.

Probable cause: Parameter specification error in the SCANREFR CLIST.

Result: Processing continues with the default value of 2 weeks.

SCA9309 INVENTORY SAMPLE OPTION IN AFFECT

Explanation: The sample option, which limits discovery of resources, is enable in the INVCNFG CLIST.

SCA9310 INVENTORY REFRESH STARTING

Explanation: Initialization of inventory task is complete, and the refresh discovery process is started.

SCA9311 INVENTORY REFRESH DELETE COMPLETE

Explanation: Deletion of old resources has completed processing.

SCA9312 INVENTORY REFRESH NEW COMPLETE

Explanation: Inventory discovery of “New” resources has completed.

SCA9313 INVENTORY REFRESH ALL COMPLET

Explanation: The Inventory task has completed refresh discovery processing for all resource types.

SCA9314 CACHE GETMAIN FAILED, NEW OPTION FAILED

Explanation: The requested storage for new discovery processing was denied.

Probable cause: Global storage problem.

Result: No discovery processing is performed.

SCA9315 AIMREAD: 100,000 RECORDS IN AIMDATA FILE, NO NEW INVENTORY RECORDS CAN BE ADDED

Explanation: The Inventory in-memory cache holds a maximum of 100,000 records. This informational message is displayed when the maximum is reached. Any resources not in cache will be discovered in any subsequent RFRESH NEW processing.

SCA9320 taskname: TASK READY TO PROCESS REFRESH CMD

Explanation: Initialization of the Inventory Task is complete.

Result: processing continues.

SCA9321 taskname: DSIFRE FAILED FOR USER STORAGE

Explanation: An attempt to release working storage, via the NetView DSIFREE macro, failed.

Probable cause: Internal processing error, or other NetView storage problem.

Result: Termination processing continues.

SCA9322 taskname: DSIFRE FAILED FOR QUEUED STORAGE

Explanation: An attempt to release storage queued storage, via the NetView DSIFREE macro, failed.

Probable cause: Internal processing error, or other NetView storage problem.

Result: Termination processing continues.

SCA9323 INVALREQ: xxxx....

Explanation: An invalid or unexpected message was received via NetView Message Queueing Services.

Probable cause: Internal processing error.

Result: The message is ignored. Processing continues.

SCA9324 taskname: DSIFRE FAILED FOR MQS BUFFER

Explanation: An attempt to release storage MQS storage buffer, via the NetView DSIFREE macro, failed.

Probable cause: Internal processing error, or other NetView storage problem.

Result: Termination processing continues.

SCA9325 taskname: DSIGET FAILED FOR USER STORAGE

Explanation: An attempt to obtain working storage via the NetView DSIGET macro failed.

Probable cause: Global NetView storage problem.

Result: Inventory processing terminates.

SCA9326 taskname: REFRESH COMPLETE, TASK TERMINATED

Explanation: Inventory processing has completed.

Result: The Inventory Task will terminate.

SCA9327 taskname: ENQ ERROR

Explanation: An attempt to ENQ the TVB chain has failed.

Probable cause: Internal processing error.

Result: Inventory Task will terminate.

SCA9328 taskname: DEQ ERROR

Explanation: An attempt to DEQ the TVB chain has failed.

Probable cause: Internal processing error.

Result: Inventory Task will terminate.

SCA9329 taskname: TASK ALREADY EXISTS

Explanation: Duplicate task, with the same task name, has been found on the TVB chain.

Probable cause: Internal processing error.

Result: Inventory Task will terminate.

SCA9410: TIMOUT TOO SHORT IN VPD REQUEST

Explanation: A timeout occurred while waiting for VPD message.

Probable cause: High system activity, or timeout value was set too low.

Result: No data is collected for the resource.

SCA9411: ERROR ENCOUNTERED WHILE PROCESSING VPD DATA FOR xxxx

Explanation: A non-zero return code was encountered while retrieving a VPD message for the specified resource.

Probable cause: Internal processing error.

Result: Data collection for the resource is discontinued.

SCAN15021: SCANVIC NOT INVOKED UNDER SCANAUTO

Explanation: An attempt was made to invoke the SCAuto initialization CLIST, in a task other than SCANAUTO autotask of NetView.

Probable cause: Possible user error.

Result: The initialization processing request is rejected.

SCA9930: SCANINVT NOT ACTIVATED CORRECTLY, RESTART SCANV390

Explanation: The inventory task, SCANINVT failed to initialize properly.

Probable cause: Internal control error.

Result: RFRESH command fails.

SCA9932: UNABLE TO START INVENTORY TASK, SCANINVT

Explanation: A timeout occurred while initializing the inventory facilities.

Probable cause: High system activity, or internal NetView problem.

Result: RFRESH command fails.

SCA9933: UNABLE TO START INVENTORY TASK, SCANINVT

Explanation: The initialization or start-up of the inventory facilities of SCAuto for NetView failed.

Probable cause: Internal NetView problem, or possible setup problem.

Result: RFRESH command fails.

SCA9934: REFRESH ALREADY IN PROGRESS, TRY AGAIN LATER

Explanation: A previous invocation of RFRESH has not yet completed processing.

Probable cause: User error.

Result: The second RFRESH command fails. The original continues processing.

SCA9936: INVENTORY NOT CURRENTLY ACTIVATED, SEE INVCNFG

Explanation: The inventory facilities of SCAuto for NetView are not enabled..

Probable cause: The SCANVINV parameter in the INVCNFG CLIST is set to “NO”, or to an invalid value.

Result: RFRESH command fails.

Index

M

N

O

P

R

S

T

U

V

.


Peregrine
SYSTEMS®
The Infrastructure Management Company™