Notice

Cabletron Systems reserves the right to make changes in specifications and other information contained in this document without prior notice. The reader should in all cases consult Cabletron Systems to determine whether any such changes have been made.

The hardware, firmware, or software described in this manual is subject to change without notice.

IN NO EVENT SHALL CABLETRON SYSTEMS BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS MANUAL OR THE INFORMATION CONTAINED IN IT, EVEN IF CABLETRON SYSTEMS HAS BEEN ADVISED OF, KNOWN, OR SHOULD HAVE KNOWN, THE POSSIBILITY OF SUCH DAMAGES.

Virus Disclaimer

Cabletron has tested its software with current virus checking technologies. However, because no anti-virus system is 100% reliable, we strongly caution you to write protect and then verify that the Licensed Software, prior to installing it, is virus-free with an anti-virus system in which you have confidence.

Cabletron Systems makes no representations or warranties to the effect that the Licensed Software is virus-free.

Copyright © December 1998, by Cabletron Systems, Inc. All rights reserved.

Printed in the United States of America.

Order Number: 9030539 E6

Cabletron Systems, Inc.
P.O. Box 5005
Rochester, NH 03866-5005

SPECTRUM, the SPECTRUM IMT/VNM logo, DCM, IMT, and VNM are registered trademarks, and SpectroGRAPH, SpectroSERVER, Inductive Modeling Technology, Device Communications Manager, and Virtual Network Machine are trademarks of Cabletron Systems, Inc.

C++ is a trademark of American Telephone and Telegraph, Inc.
UNIX is a trademark of UNIX System Laboratories, Inc.
OSF/Motif and Motif are trademarks of the Open Software Foundation, Inc.
X Window System is a trademark of X Consortium, Inc.
Ethernet is a trademark of Xerox Corporation.
IBM and NetView are registered trademarks of the International Business Machines Corporation.
CNI and Brixton are trademarks of the Computer Network Technology Corporation.
Restricted Rights Notice

(Applicable to licenses to the United States Government only.)

1. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.


2. (a) This computer software is submitted with restricted rights. It may not be used, reproduced, or disclosed by the Government except as provided in paragraph (b) of this Notice or as otherwise expressly stated in the contract.

(b) This computer software may be:

(1) Used or copied for use in or with the computer or computers for which it was acquired, including use at any Government installation to which such computer or computers may be transferred;

(2) Used or copied for use in a backup computer if any computer for which it was acquired is inoperative;

(3) Reproduced for safekeeping (archives) or backup purposes;

(4) Modified, adapted, or combined with other computer software, provided that the modified, combined, or adapted portions of the derivative software incorporating restricted computer software are made subject to the same restricted rights;

(5) Disclosed to and reproduced for use by support service contractors in accordance with subparagraphs (b) (1) through (4) of this clause, provided the Government makes such disclosure or reproduction subject to these restricted rights; and

(6) Used or copied for use in or transferred to a replacement computer.

(c) Notwithstanding the foregoing, if this computer software is published copyrighted computer software, it is licensed to the Government, without disclosure prohibitions, with the minimum rights set forth in paragraph (b) of this clause.

(d) Any other rights or limitations regarding the use, duplication, or disclosure of this computer software are to be expressly stated in, or incorporated in, the contract.

(e) This Notice shall be marked on any reproduction of this computer software, in whole or in part.
## Preface

User Prerequisites ........................................................................................................................................... xi
How This Manual Is Organized .................................................................................................................. xi
Conventions ...................................................................................................................................................... xiii
Related Documentation .................................................................................................................................. xiii
Questions about SPECTRUM Documentation .......................................................................................... xiv
How to Contact SPECTRUM Documentation ........................................................................................... xiv
Where to Find Related SPECTRUM Documentation ................................................................................ xiv

### Chapter 1 Introduction

Overview .......................................................................................................................................................... 1-1
SPECTRUM/NV-S Gateway Functions ........................................................................................................ 1-3
  Event Conversion ....................................................................................................................................... 1-3
  Configurable Mapping ............................................................................................................................. 1-4
  Default Mapping ..................................................................................................................................... 1-4
  Event Filtering ....................................................................................................................................... 1-5
  Variable Insertion .................................................................................................................................. 1-5
  RUNCMD Support ................................................................................................................................... 1-6
SPECTRUM/NV-S Gateway Components .................................................................................................... 1-6
  SPECTRUM/NV-S Gateway Process ........................................................................................................ 1-6
  SPECTRUM/NV-S Gateway Defaults ...................................................................................................... 1-7
  SPECTRUM/NV-S Gateway Configuration .............................................................................................. 1-7
  CNT/Brixton BrxPU2.1 SNA Server ........................................................................................................... 1-7

### Chapter 2 Starting the SPECTRUM/NV-S Gateway

Overview .......................................................................................................................................................... 2-1
Establishing SNA Connectivity .................................................................................................................... 2-2
Invoking SPECTRUM/NV-S Gateway Manually .......................................................................................... 2-3
Invoking SPECTRUM/NV-S Gateway Through SPECTRUM .................................................................... 2-5
Checking NetView Alert Display .................................................................................................................. 2-6
Executing NetView RUNCMD ...................................................................................................................... 2-6
RUNCMD Scripts ........................................................................................................................................ 2-7
  Sending A Message to Logged In Users ................................................................................................. 2-7
  Sending UNIX Mail ................................................................................................................................ 2-8
  SPECTRUM Command Line Interface (CLI) Support ......................................................................... 2-8
Chapter 3  The NVGW-Spectrum Process

Chapter 4  Event Mapping

Chapter 5  Vector Values
Chapter 6 Troubleshooting

Overview ................................................................. 6-1
On-Line Help .............................................................. 6-2
NetView NPDA Filters ................................................. 6-2
Operator Status Information ...................................... 6-3
Tracing ..................................................................... 6-3
Logical Data Scope .................................................... 6-3
Common Problems .................................................... 6-4
Expand Mappings ....................................................... 6-5
Run Command Messages ............................................ 6-5

Appendix A Sample SNA Configuration

Sample SNA Configuration .......................................... A-1

Appendix B brxspec.defaults File

Overview ...................................................................... B-1

Appendix C brxspec.config File

Overview ...................................................................... C-1
Appendix D  Messages

Overview ............................................................................................................................. D-1
CNT/Brixton Message Display Utility - bmsg .................................................................. D-2
CNT/Brixton Message Database (BMD) ........................................................................... D-2
Message Files ..................................................................................................................... D-3
Message Field ...................................................................................................................... D-4
Parameters Field ................................................................................................................ D-4
Options Field ....................................................................................................................... D-4
Cause Field ........................................................................................................................ D-5
Effect Field ......................................................................................................................... D-5
Action Field ......................................................................................................................... D-5

Appendix E  UNIX to NetView (UNVW) Messages

UNIX to NetView (UNVW) Messages ............................................................................... E-1

Index
| Figure 1-1. | SPECTRUM/NV-S Gateway Relationships | 1-2 |
| Figure 1-2. | Monitoring Functionality | 1-2 |
| Figure 1-3. | Control Functionality | 1-3 |
| Figure 4-1. | Sample alert_map record | 4-7 |
| Figure A-1. | Example SNA Configuration | A-2 |
Table 2-1. snmget Command Parameters ......................................................... 2-10
Table 2-2. snmpnext Command Parameters .................................................. 2-11
Table 2-3. snmpset Command Parameters .................................................... 2-12
Table 3-1. SNA Vector Values ........................................................................ 3-2
Table 5-1. General Alert Data Subvector ..................................................... 5-2
Table 5-2. Basic Alert Subvector .................................................................... 5-2
Table 5-3. Product Set ID Subvector .............................................................. 5-3
Table 5-4. Hardware Product Identifier Subvector ........................................ 5-4
Table 5-5. Hardware Product Identifier ......................................................... 5-4
Table 5-6. Emulated Product Identifier .......................................................... 5-5
Table 5-7. Hardware Product Common Name ................................................. 5-5
Table 5-8. Software Product Identifier Subvector ......................................... 5-5
Table 5-9. Software Product Program Number .............................................. 5-6
Table 5-10. Software Product Common Level .............................................. 5-6
Table 5-11. Software Product Common Name .............................................. 5-6
Table 5-12. Probable Causes Subvector ....................................................... 5-7
Table 5-13. User Causes Subvector ............................................................... 5-8
Table 5-14. User Causes Subfield ................................................................. 5-8
Table 5-15. Recommended Actions Subfield ................................................ 5-9
Table 5-16. Detailed Data Subfield ............................................................... 5-9
Table 5-17. Product Set ID Index Subfield .................................................... 5-9
Table 5-18. Install Causes Subvector ............................................................. 5-10
Table 5-19. Install Causes Subfield ............................................................... 5-10
Table 5-20. Recommended Actions Subfield ................................................ 5-11
Table 5-21. Detailed Data Subfield ............................................................... 5-11
Table 5-22. Product Set ID Index Subfield .................................................... 5-11
Table 5-23. Failure Causes Subvector .......................................................... 5-12
Table 5-24. Failure Causes Subfield ............................................................. 5-12
Table 5-25. Recommended Actions Subfield ................................................ 5-13
Table 5-26. Detailed Data Subfield ............................................................... 5-13
Table 5-27. Product Set ID Index Subfield .................................................... 5-13
Table 5-28. Cause Undetermined Subvector ................................................ 5-14
Table 5-29. Recommended Actions Subfield ................................................ 5-14
Table 5-30. Detailed Data Subfield ............................................................... 5-15
Table 5-31. Product Set ID Index Subfield .................................................... 5-15
Table 5-32. Hierarchy Resource List Subvector ............................................ 5-16
Table 5-33. Hierarchy Name List Subfield .................................................... 5-16
Table 5-34. Self-Defining Text Subvector ..................................................... 5-17
Table 5-35. Transparent Subvector ............................................................. 5-18
Table 6-1. Common Problems ................................................................. 6-4
Table A-1. NCP/VTAM Macros ................................................................. A-1
Preface

Describes user prerequisites, how this manual is organized, conventions used in this manual, and how to contact the SPECTRUM documentation department and access SPECTRUM documentation.

User Prerequisites

SPECTRUM/NV-S Gateway users should have a working knowledge of the capabilities and functions provided by the following products:

- SPECTRUM Enterprise Manager
- IBM NetView
- CNI/Brixton BrxPU2.1 SNA Server

See the SPECTRUM/NV-S Gateway Software Release Notice (SRN), which accompanies this manual, for information about product versions compatible with this release of the SPECTRUM/NV-S Gateway product.

How This Manual Is Organized

This manual is organized as follows:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 1</td>
<td></td>
</tr>
<tr>
<td>Introduction</td>
<td>Provides an overview of the SPECTRUM/NV-S Gateway product.</td>
</tr>
<tr>
<td>Chapter 2</td>
<td></td>
</tr>
<tr>
<td>Starting the SPECTRUM/NV-S Gateway</td>
<td>Describes how to establish SNA connectivity, invoke the SPECTRUM/NV-S Gateway, generate SPECTRUM events, and view SNA alerts at the NetView console.</td>
</tr>
</tbody>
</table>
How This Manual Is Organized

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 3</td>
<td>Describes the NVGW-Spectrum process, including the name, synopsis, description, files, and return value.</td>
</tr>
<tr>
<td>The NVGW-Spectrum Process</td>
<td></td>
</tr>
<tr>
<td>Chapter 4</td>
<td>Describes how the SPECTRUM/NV-S Gateway uses its configuration files to implement alarm filtering and event-to-alert conversion.</td>
</tr>
<tr>
<td>Event Mapping</td>
<td></td>
</tr>
<tr>
<td>Chapter 5</td>
<td>Lists all the values which you can specify for an SNA alert message using the SPECTRUM/NV-S Gateway configuration options.</td>
</tr>
<tr>
<td>Vector Values</td>
<td></td>
</tr>
<tr>
<td>Chapter 6</td>
<td>Describes several tools used to determine and resolve problems with the SPECTRUM/NV-S Gateway and the CNT/Brixton BrxPU2.1 SNA Server.</td>
</tr>
<tr>
<td>Troubleshooting</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appendix A</td>
<td>Presents a sample portion of an SNA host network configuration.</td>
</tr>
<tr>
<td>Sample SNA Configuration</td>
<td></td>
</tr>
<tr>
<td>Appendix B</td>
<td>Describes the brxspec.defaults file.</td>
</tr>
<tr>
<td>brxspec.defaults File</td>
<td></td>
</tr>
<tr>
<td>Appendix C</td>
<td>Describes the brxspec.config file.</td>
</tr>
<tr>
<td>brxspec.config File</td>
<td></td>
</tr>
<tr>
<td>Appendix D</td>
<td>Describes the CNT/Brixton Message Database, BMD, the general format of a message, and bmsg, the CNT/Brixton on-line help facility.</td>
</tr>
<tr>
<td>Messages</td>
<td></td>
</tr>
<tr>
<td>Appendix E</td>
<td>Lists the UNVW_msglib file, containing the SPECTRUM/NV-S Gateway-specific UNIX-to-NetView (UNVW) messages.</td>
</tr>
<tr>
<td>UNIX to NetView (UNVW) Messages</td>
<td></td>
</tr>
</tbody>
</table>
Conventions

This manual uses the following conventions:

• System text, including command strings you enter, scripts, code examples, and messages, appears in Courier font. For example:

   RUNCMD SP = SP_NAME, APPL =UNIX

• Directory names, filenames, and file paths appear in boldface. For example:

   usr/NVGW/.nvrc

• References to SPECTRUM publications and other publications appear in bold italics. For example:

   Brixton PU2.1 SNA Server - Configuration and Network Administration Guide

• Referenced chapter titles and section headings appear in italics (hypertext-blue italics for on-line readers).

• The acronym NVGW (for NetView Gateway) is used throughout this manual to refer to the SPECTRUM/NV-S Gateway product.

Related Documentation

Refer to the following documentation for more information on using the SPECTRUM/NV-S Gateway product.

SPECTRUM/NV-S Gateway Documentation:

• SPECTRUM/NV-S Gateway Software Release Notice (SRN)

   See Questions about SPECTRUM Documentation for more information.

IBM Documentation:

• IBM Network Program Products Planning

• IBM VTAM Installation and Resource Definition
Questions about SPECTRUM Documentation

This section describes how to contact SPECTRUM documentation and access related SPECTRUM documentation.

How to Contact SPECTRUM Documentation

Send your questions, comments, or suggestions regarding SPECTRUM documentation to the Technical Communications Department directly via the following internet address:

spectrum-techdocs@ctron.com

Where to Find Related SPECTRUM Documentation

Use the following resources to find related SPECTRUM documentation referenced in this manual or information about any SPECTRUM topic referenced but not covered in detail this manual:

• SPECTRUM Enterprise Manager Documentation CD

• Cabletron’s Service and Support site at http://www.cabletron.com
Chapter 1

Introduction

Provides an overview of the SPECTRUM/NV-S Gateway product.

Overview

The SPECTRUM/NV-S Gateway (NVGW) provides an interface between SPECTRUM and IBM NetView to support the monitoring and control of SNMP networks.

Events generated by devices managed by SPECTRUM are logged and forwarded to NVGW where they are converted to an SNA alert format and transmitted to NetView. The NVGW can be extended to include specialized event-to-alert conversions to meet your particular network management requirements. See Chapter 4 for more information.

Using NetView's RUNCMD facility, a NetView operator can issue UNIX commands through NVGW. A special feature of this gateway is support for the following SNMP commands: snmpget, snmpnext, and snmpset, which for example could be used to obtain attribute information about a device managed through SPECTRUM.

NVGW uses the SNA functions provided by the CNT/Brixton BrxPU2.1 SNA Server, which manages the physical connection to the SNA network, processes the lower level SNA protocols, and provides external interfaces for client programs like NVGW. The functional relationships between NVGW, the Brixton server, SPECTRUM, UNIX, and NetView are depicted in Figure 1-1.
Figure 1-1. SPECTRUM/NV-S Gateway Relationships

NVGW provides both monitoring and control functionality. Monitoring functions include:

- Receiving events from SPECTRUM
- Converting SPECTRUM events to NMVT format
- Sending the events to NetView through the PU2.1 SSCP-PU session

Figure 1-2 depicts NVGW monitoring functions.

Figure 1-2. Monitoring Functionality
Control Functions include:

- Receiving RUNCMD commands from NetView
- Translating RUNCMD commands to UNIX or SNMP commands
- Executing the UNIX or SNMP commands
- Gathering UNIX and SNMP command output
- Returning RUNCMD replies to NetView

Figure 1-3 depicts control functions.

Figure 1-3. Control Functionality

SPECTRUM/NV-S Gateway Functions

NVGW provides several functions which support the conversion of SPECTRUM events into SNA alerts. It is designed to aid you in extending the base-level conversions and in processing requests from NetView operators.

Event Conversion

NVGW converts SPECTRUM events into SNA alerts. NVGW includes two configuration files that contain instructions for event-to-alert conversion. You can update these files to include new conversion instructions and shield you from the many complexities inherent in constructing SNA alerts.

Event conversion takes place when NVGW receives an event from SPECTRUM. During NVGW initialization, the event-to-alert conversion instructions are read from the configuration files into memory-resident tables. These tables are searched to find the specific instructions for converting the received SPECTRUM event into an SNA alert, which is then sent through the Brixton PU2.1 SSCP-PU session to the SNA host.
Configurable Mapping

The mapping of SPECTRUM events into SNA alerts is user-configurable. The `brxspec.config` file contains the alert mapping records for NVGW. Each alert mapping record identifies the type of event it maps and describes how these events should be converted into SNA alerts. To update the event-to-alert conversion, you either add new records to or modify records in the `brxspec.config` file. An example file is shown in Appendix C.

The conversion instructions, or alert definitions, are lists of SNA alert subvectors. Each entry defines an element of the SNA alert and assigns values to the element’s fields. The following subvectors are supported:

- Generic
- Basic
- Product Set ID
- Probable Causes
- User Causes
- Install Causes
- Failure Causes
- Cause Undetermined
- Resource Hierarchy List
- Self-Defining Text

In addition, transparent records are supported which enable you to construct other subvectors. Refer to IBM documentation for a more detailed description of each of these fields.

Default Mapping

The alert_map shown in Appendix C shows a very simple configuration. You may have wondered why the subvectors were so sparse — very few field values are defined for any subvector. This example actually describes a relatively complex SNA alert, but the complexity is hidden. When all of the mandatory fields for a subvector are not defined, NVGW places default values in the skipped fields.

The `brxspec.defaults` file contains a listing of all the subvector values which can be assigned (a list of the `brxspec.defaults` file is listed in Appendix B. Every subvector value is given a default value. Therefore, only the values you want to be different from the default values need to be defined.
Event Filtering

NVGW supports event filtering. This enables you to select which events to forward or not forward to NetView. Initial filtering is defined in the `.nvrc` resource file in the default NVGW directory, `/usr/NVGW`. All SPECTRUM events are ranked within SPECTRUM on a event-severity scale from 0-100. The `alarm_severity_threshold` parameter in the NVGW resource file, `.nvrc` is used to set a threshold which filters events that are greater than or equal to the threshold value.

A second level of filtering can be set in the alert maps. The filtering can be set using the following factors:

- **Event type** - Identifies numeric code of type of event.
- **Severity** - Identifies severity level of the event.
- **Model name** - Identifies name of the SPECTRUM model producing event.
- **Model type** - Identifies SPECTRUM model type of model producing event.

Both levels of filtering enable you to set very specific parameters to control the events that are passed from SPECTRUM to NVGW and on to the NetView host.

Variable Insertion

NVGW enables you to insert specific information from a SPECTRUM event into an SNA alert. Information from the SPECTRUM event is provided by special variables to NVGW, which replaces the special variables names with the corresponding value taken from the SPECTRUM event.

The supported variable names are:

- `%event_type`
- `%severity`
- `%time`
- `%model_name`
- `%model_type`
- `%user_name`
- `%message_text`

See Variable Assignments on Page 4-9 for a description of the variables and examples.
RUNCMD Support

NVGW enables SNA network operators to execute commands on remote UNIX systems by using NetView's RUNCMD facility.

From a NetView console, you use the RUNCMD to pass commands to a UNIX system running NVGW, which receives the request, translates the command from EBCDIC to ASCII, and invokes a local command processor (UNIX shell) to execute the command. The output generated by the local command process is converted back to EBCDIC, a SNA request message is generated by the NVGW, and the message is returned to the BrxPU2.1 SNA Server for transmission to NetView.

The commands and utilities supported by NVGW are defined in the `brxspec.config` file. The following are provided as defaults in the product:

- broadcast
- cat
- date
- ls
- mailmsg
- pwd
- snmpget
- snmpnext
- snmpset
- trmm

To add commands, they must be included in the `brxspec.config` file and placed in a directory that is accessible through the path variable.

SPECTRUM/NV-S Gateway Components

NVGW relies on several configuration files to convert SPECTRUM events into SNA alerts. It also requires SNA connections.

SPECTRUM/NV-S Gateway Process

The NVGW function is implemented by a process called NVGW-Spectrum. This process registers with SPECTRUM to receive events and connects to the BrxPU2.1 SNA Server to gain SNA access. This process reads the `brxspec.config` and `brxspec.defaults` files during initialization to determine how events should be converted into SNA alerts before being sent to NetView.
SPECTRUM/NV-S Gateway Defaults

NVGW refers to the **brxspec.defaults** file to establish values for fields which have not been configured in the **brxspec.config** file for a defined event.

SPECTRUM/NV-S Gateway Configuration

NVGW converts SPECTRUM events into SNA alerts according to the instructions listed in the **brxspec.config** file. To modify or extend the event-to-alert mappings, you must update the **brxspec.config** file.

CNT/Brixton BrxPU2.1 SNA Server

NVGW requires an SNA SSCP-PU control session to pass messages to NetView and to accept RUNCMD commands from the NetView host. The CNT/Brixton BrxPU2.1 SNA Server provides NVGW access to this kind of control session. It manages the physical connection to the SNA network, basic service management, and lower level SNA protocols. Refer to the **Brixton PU2.1 SNA Server - Sun Solaris Sparc Planning and Installation Manual** and the **Brixton PU2.1 SNA Server - Configuration and Network Administration Guide** for more information.
Chapter 2

Starting the SPECTRUM/NV-S Gateway

Describes how to establish SNA connectivity, invoke the SPECTRUM/NV-S Gateway, generate SPECTRUM events, and view SNA alerts at the NetView console.

Overview

Before you can start and implement NVGW, you must first install and configure as desired NVGW, SPECTRUM Enterprise Manager, and CNT/Brixton's BrxPU2.1 SNA Server software. To view NetView screens from the SpectroGRAPH station, you must also install the Brx3270 SNA terminal emulator on that station. See the SPECTRUM/NV-S Gateway Installation Guide for information on installing the NVGW product and the Brixton PU2.1 SNA Server - Sun Solaris Sparc Planning and Installation Manual for information on installing Brixton server software.

Because NVGW is pre-configured to immediately be able to pass SPECTRUM event data to NetView after it is installed, you do not have to configure those operational settings until you decide how you want to customize the product.

The SPECTRUM/NV-S Gateway can either be started manually or through a SPECTRUM icon. The manual procedure involves first establishing SNA connectivity, and then invoking the SPECTRUM/NV-S Gateway process. The following sections describe these procedures. The last sections discuss packaging the event data in an SNA alert, viewing the SNA alert at the NetView console, and the SPECTRUM/NV-S Gateway RUNCMD feature.

If you encounter problems with the NVGW after performing the procedures described in this chapter, see Chapter 6 for information on troubleshooting strategies.
Establishing SNA Connectivity

The SPECTRUM/NV-S Gateway utilizes the CNT/Brixton BrxPU2.1 SNA Server to transfer SNA alerts to the IBM mainframe. This server provides a connection to the SNA network and maintains control sessions with NetView. The SPECTRUM/NV-S Gateway uses the SSCP-PU control session to pass SNA alerts up to NetView. The SSCP-PU control session must be active before these messages can flow. Consequently, your first step is to establish SNA connectivity.

If you already employ other CNT/Brixton Systems SNA End Node products like the Brx3270 SNA terminal emulator, you may skip to the following section since your SNA connection is already set up. Otherwise, you need to configure a communications link with the SNA network.

The Brixton PU2.1 SNA Server - Sun Solaris Sparc Planning and Installation Manual describes how to install the BrxPU2.1 SNA Server. The Brixton PU2.1 SNA Server - Configuration and Network Administration Guide describes how the BrxPU2.1 SNA Server can be configured to establish SNA connectivity. The Brx3270 Open Client Configuration and User Manual details how the Brixton Brx3270 terminal emulator may be configured. The following description is intended as a “quick start.” If you have special considerations or encounter problems, see the referenced documentation.

The SPECTRUM/NV-S Gateway requires that at least one Physical Unit (PU) and its link to the SNA network be configured. You must update configurations on both the IBM mainframe and the BrxPU2.1 SNA Server. Follow these steps:

1. Configure the IBM mainframe.
   Example configuration macros for the IBM mainframe (NCP GEN) are listed in Appendix A. These examples assume a point-to-point, 9600 bps line.

2. Configure the BrxPU2.1 SNA Server.
   CNT/Brixton Systems provides a number of sample configuration files for its BrxPU2.1 SNA Server. These sample files may be found in the /Brixton_root/BrxPU21/config directory where Brixton defaults the Brixton_root directory name to /opt on Solaris systems. Consult with the system administrator who installed the BrxPU2.1 SNA Server for the exact Brixton_root directory pathname if the default pathname was not used. Unless a particular configuration file is explicitly specified on the command line when the BrxPU2.1 SNA Server is invoked, the default BrxPU2.1 configuration file used is the /Brixton_root/BrxPU21/brxpu2.config file. For more details on configuring the BrxPU2.1 SNA Server, refer to the Brixton PU2.1 SNA Server - Configuration and Network Administration Guide.
3. Ensure that the local BrxPU2.1 configuration corresponds with the SNA definitions on the IBM mainframe.

4. Start the BrxPU2.1 SNA Server to establish connectivity into the SNA network.

To invoke the BrxPU2.1 SNA server manually:

a. Log on as root user, navigate to the /Brixton_root/BrxPU21 directory, and then execute the ./brxsetup command.

b. Select option 4 - Start BrxGman from the setup menu to start all necessary Brixton components.

Refer to Appendix E for a list and a description of possible error messages and that may appear on your display when attempting to start the server. You can also use the Brixton bmsg utility to extrapolate more information from an error message.

5. Determine whether the SNA PU has been activated. You can either have the SNA network administrator check the status of the PU in NetView or you can use the Brixton Server Management interface to check the status.

To determine the status using the Brixton interface:

a. Execute the brxgmi command on the Brixton station.

b. Navigate from the System icon on the GUI that appears to the PU2.1 Servers icon.

c. Access the list of PU2.1 servers from the PU2.1 Servers icon. The server you set up should appear in the list with a status of “Active.” If it does not, highlight the server entry and select Activate from the Control option on the menu bar.

To terminate the BrxPU2.1 SNA Server:

Execute the ./brxsetup command, and then select option 6 - Stop BrxGman to terminate all Brixton components.

**Invoking SPECTRUM/NV-S Gateway Manually**

To pass SNA alerts to the host, NVGW requires an active SSCP-PU control session with the IBM mainframe. While the SSCP-PU session is inactive, SPECTRUM events are not forwarded to the host. NVGW indicates this with the following message:

PU not open on active reply
To start NVGW manually:

1. Navigate to the default NVGW directory
   
   ```
   cd .../Spectrum/NVGW/prod
   ```

2. Start NVGW.
   
   ```
   NVGW-Spectrum
   ```

   NVGW’s SpectroSERVER must be running before it can be started.

See Chapter 3, for NVGW start-up options.

The SPECTRUM/NV-S Gateway is now prepared to send converted SPECTRUM events to NetView. You can check the status of the PU (as described in Establishing SNA Connectivity on Page 2-2). An “Active” status for the PU indicates that NVGW has established contact with the PU.

To terminate the NVGW daemon process:

1. Use the `ps` command to determine NVGW’s process ID.

2. Execute the following command:
   
   ```
   kill -INT NVGW-Spectrum_pid
   ```

   It is important to use the `-INT` option in the `kill` command. NVGW keys off this signal to clean up and deregister with SPECTRUM’s event dispatcher.

After terminating NVGW you can introduce new event-to-alert mappings by updating the `brxspec.config` file and then restarting NVGW.
Invoking SPECTRUM/NV-S Gateway Through SPECTRUM

After the NVGW is installed, you must create an icon representing it in the SpectroGRAPH Topology View. The icon you create will enable you to start NVGW from the SpectroGRAPH.

To create the NVGW icon:

1. Select **Edit** from the **File** pull-down menu in the Topology View to change from view to edit mode.

2. Select **New Model...** from the **Edit** pull-down menu to invoke the Select Model Type dialog box.

3. Choose **NVGW**, the NetView Gateway model, from the Select Model Type dialog box.

   The Creating NVGW dialog box appears.

4. Click on the **OK** button on the Creating NVGW dialog box.

   The NVGW icon appears on the SpectroGRAPH.

5. Select **Close Edit** from the **File** menu to exit edit mode and return to view mode.

To start NVGW from the NVGW icon:

1. Invoke the **Icon Subviews** menu from the icon.

2. Select the **CsNVGWScript** script from the menu.

   The script performs the following functions:

   - Verifies that the CNT/Brixton BrxPU2.1 and Brx3270 install directories are present on the CNT/Brixton SNA End Node host.

   - Starts the BrxPU2.1 SNA Server if it is not already running on the CNT/Brixton SNA End Node host.

   - Starts the Brx3270 SNA terminal emulator and displays the 3270 display window on the local system.

   - Starts NVGW-Spectrum, the Gateway process, if the BrxPU2.1 SNA Server is available and the Gateway process was not already active.
Checking NetView Alert Display

SNA alerts are logged and displayed by NetView. To view the SNA alerts, move to the Network Problem Determination Aid (NPDA) screen and choose to view the Alert Dynamic (ALD) display. NetView continuously updates the Alert Dynamic display to show the most recent SNA alert conditions. Also check the NPDA filter setup to make sure that the event types sent by NVGW are not filtered out.

You can display more information about the SNA alert you generated from SPECTRUM. Press the Enter key and then select the number associated with your SNA alert. You will recognize the alert because the name of the model which generated the event report will be displayed in the left column. If no model name is associated with the event, the name will be CABLETRON.

At the top of the subsequent screen, a graphical image of the resource hierarchy indicates the relationship of the alert to network resources. In this case, the model name and model type are shown.

If you move to the Event Detail screen, you can view more detailed data about the alert.

Browse through some of the other screens to see more information about your SNA alert.

To collect events from an alternate SPECTRUM network management server, simply change the name of the SpectroSERVER host in the .nvrc file (configured during the install process), or pass the name of the alternate server host using the "-vnm" command line argument as follows:

```
NVGW-Spectrum -vnm <hostname>
```

where <hostname> is the name by which the server host is known to the local host and is currently running the SpectroSERVER with the gateway user in its users database. See Chapter 3 for more information.

Executing NetView RUNCMD

A network operator can issue commands to a service point for execution using NetView command, RUNCMD. This command is entered in NetView’s NCCF facility and has the following format:

```
RUNCMD SP = SP_NAME, APPL = APPLICATION_NAME,UNIX_COMMAND_STRING
```

where:

SP_NAME - Specifies the name of the service point which will execute the command.
APPLICATION_NAME – Defines the application name. You may enter any name (for example, UNIX).

UNIX_COMMAND_STRING – The UNIX command to be executed.

Example:
If you want to get the current date on the UNIX system where the service point is running, type the following command at the prompt in NetView’s NCCF facility:

```bash
run cmd sp=sp_name,appl=unix,date
```

where:

- **SP_NAME** – is the name of the service point.
- **date** – is the UNIX command that will be executed on the remote UNIX system.

NetView displays responses to the RUNCMD requests at the NetView console. The response to the above command would be:

```
Thurs Jul 04 12:34:34 EST 1992
```

Refer to Chapter 6 for an explanation of any error messages that may be returned in response to a RUNCMD request.

## RUNCMD Scripts

RUNCMD may be used to invoke either standard UNIX commands or user-defined scripts or programs. The next two sections demonstrate the use of this feature.

### Sending A Message to Logged In Users

The following script “broadcast” should be created and be accessible through your PATH variable:

```
#!/bin/sh
# example script to display a message to people logged into the service point workstation
# Usage: broadcast {message} {user}
# Where {message} is text delimited by quotes and {user}
# is the person to send the message to. If the {user} field is blank the message is sent to all users.
```

```bash
run cmd sp=sp_name,appl=unix,broadcast "{message}" {user}
```
RUNCMD Scripts

Sending UNIX Mail

```bash
if [ -z "$2" ]
then
  # {user} field is blank - display message to all users
  echo $1 | wall
else
  # {user} field has user name - send message to users
  # console
  echo $1 | write $2 console
fi
```

Issue the following RUNCMD command from NetView’s NCCF facility:

```
RUNCMD SP=SP_NAME,APPL=UNIX,broadcast "Hello Fred" fred
```

The message “Hello Fred” will be sent to all users logged on as fred on the system where the service point is running.

Sending UNIX Mail

The following script “mailmsg” should be created and be accessible through your PATH variable:

```
#!/bin/sh

echo $1 | mail $2
```

where:

$1 — Specifies the message to be mailed (enclosed in quotes).
$2 — Specifies to whom the message is to be mailed.

Issue the following command:

```
RUNCMD SP=SP_NAME,APPL=UNIX,mailmsg "hello world" fred
```

The message “hello world” will be mailed to fred.

SPECTRUM Command Line Interface (CLI) Support

The SPECTRUM Command Line Interface provides a non-graphical command-based interface to SpectroSERVER. RUNCMD can access the CLI to issue commands to the SpectroSERVER directly. Refer to the SPECTRUM Command Line Interface User’s Guide for details on CLI usage.

The following script “dispmodel” is provided as an example of CLI usage from RUNCMD. Its purpose is to display all models in the SpectroSERVER database which match the given filter. This script should be created and be accessible through your PATH variable:
#!/bin/sh
# script to display all models that match the given
# $1 is the name of the vnm host.
# $2 is the filter.
# connect to the specified SpectroSERVER
connect $1
# display the specified models
show models | grep $2
# disconnect from the SpectroSERVER
disconnect

where:

$1 — Specifies the name of the SpectroSERVER host to which you wish to
      make the request.

$2 — Specifies the model name to search for. If a partial name is specified, the
      script will display all model names beginning with that string.

Issue the following command:

RUNCMD SP=SP_NAME,APPL=UNIX,dispmodel
   SpectroSERVER_hostname model_name

NVGW-Spectrum SNMP Support

The NVGW-Spectrum product provides support for the SNMP commands
snmpget, snmpset, and snmpnext. These commands are installed in the
Gateway directory and they are used within the TCP/IP environment to either
set or extract variables in an SNMP agent.

The snmpget Command

The snmpget command, issued via the RUNCMD facility from the IBM host,
initiates a network management query to a remote management agent. The
response from the management agent is sent back to the IBM host.

The syntax of the snmpget command is:

snmpget -h remotehost [-c communityname] [-t timeout]
   [-i requestid] [-p remoteport] oidname...

The snmpget command parameters are listed and described in Table 2-1.
NVGW-Spectrum SNMP Support

The snmpget Command

For example, the following command has the effect of displaying the device description, "sysDescr," and the ID of the agent software "sysObjectIID" for the host named anyhost:

```
RUNCMD SP=SP_NAME,APPL=UNIX,
    snmpget -h anyhost 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2.0
```


The snmpnext Command

The snmpnext command, issued via the RUNCMD facility from the IBM host, attempts to retrieve and display specified subtrees of the MIB.

The syntax of the command is:

```
snmpnext  -h remotehost  [-c communityname] [-t timeout] 
[-i requestid] [-p remoteport] oidname...
```

The snmpnext command parameters are shown in Table 2-2.

### Table 2-2. snmpnext Command Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h remotehost</td>
<td>Required</td>
<td>Management request is sent to the IP address that is specified as remotehost</td>
</tr>
<tr>
<td>-c communityname</td>
<td>Optional</td>
<td>Management requests are generated and accepted using the community name specified as communityname. Note that the communityname is case-insensitive. This field defaults to 'public' and should only be used for test purposes.</td>
</tr>
<tr>
<td>-t timeout</td>
<td>Optional (seconds)</td>
<td>If a response to the request is not received within the period specified by timeout, the request is terminated and an error message is sent to the host. The default value for this field is 30 seconds. A timeout value of 0 seconds causes the request to wait indefinitely for a response.</td>
</tr>
<tr>
<td>-i requestid</td>
<td>Optional</td>
<td>The management request is identified by the number specified as requestid, rather than using the zero default value.</td>
</tr>
<tr>
<td>-p remoteport</td>
<td>Optional</td>
<td>The management request is sent to the UDP port that is specified as remoteport, rather than that assigned in the /etc/services database.</td>
</tr>
<tr>
<td>oid_name</td>
<td>Required</td>
<td>The object of interest is explicitly identified as a sequence of non-negative integer values separated by a dot. Note that more than one oid_name may be specified as part of the parameter list.</td>
</tr>
</tbody>
</table>

For example, the following command has the effect of retrieving the IP routing table for the host named anyhost:

```
RUNCMD sp=SP_NAME,APPL=UNIX, 
   snmpnext  -h anyhost 1.3.6.1.2.1.4.21.1 
```
The following command has the effect of retrieving the Internet Control Message Protocol (ICMP) statistics for the host named anyhost:

```
RUNCMD sp=SP_NAME,APPL=UNIX,snmpnext -h anyhost
1.3.6.1.2.1.5
```

### The snmpset Command

The `snmpset` command, issued via the RUNCMD facility from the IBM host, initiates a network management request to a remote management agent to alter the items of management information with values specified by the user.

It is invoked with the syntax:

```
snmpset -h remotehost [-c communityname] [-t timeout] [-i requestid] [-p remoteport] { oid_name oid_type oid_value }...
```

The `snmpset` command parameters are shown in **Table 2-3**

### Table 2-3. `snmpset` Command Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h remotehost</td>
<td>Required</td>
<td>Management request is sent to the IP address that is specified as remotehost.</td>
</tr>
<tr>
<td>-c communityname</td>
<td>Optional</td>
<td>Management requests are generated and accepted using the community name specified as <code>communityname</code>. Note that the <code>communityname</code> is case insensitive. This field defaults to ‘public’ and should only be used for test purposes.</td>
</tr>
<tr>
<td>-t timeout</td>
<td>Optional</td>
<td><em>(seconds)</em> If a response to the request is not received within the period specified by <code>timeout</code>, the request is terminated and an error message is sent to the host. The default value for this field is 30 seconds. A <code>timeout</code> value of 0 seconds causes the request to wait indefinitely for a response.</td>
</tr>
<tr>
<td>-i requestid</td>
<td>Optional</td>
<td>The management request is identified by the number specified as <code>requestid</code>, instead of using the zero default value.</td>
</tr>
<tr>
<td>-p remoteport</td>
<td>Optional</td>
<td>The management request is sent to the UDP port specified as remoteport, rather than that assigned in the <code>/etc/services</code> database.</td>
</tr>
<tr>
<td>oid_name</td>
<td>Required</td>
<td>Each item of management information to be altered is specified by a set of three parameters. Note that more than one management information set may be specified.</td>
</tr>
<tr>
<td>oid_type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>oid_value</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The oid_name is the explicit object of interest id and it comprises a sequence of dot-separated non-negative integers.

The following oid_type types are supported:

- Integer
- Counter
- Gauge
- OctetString
- ObjectId
- IPAddr
- TimeTicks

The oid_value is the value assigned to the specified oid_type. Specify a value to match the type field. For example, the following command has the (hypothetical) effect of altering the “sysDescr” and “sysObjectId” values for the host named anyhost. The former is altered to the “OctetString” value “hello,” whereas the latter is altered to the “ObjectId” value 1.2.3.4.5.6.7.

RUNCMD=SP_NAME,APPL=UNIX,snmpset -h anyhost
1.3.6.1.2.1.1.1.0 OctetString "hello" 1.3.6.1.2.1.1.2.0
ObjectId 1.2.3.4.5.6.7

Token Ring Management Module Support

The Gateway is released with a script, trmm, which provides support for Cabletron’s Token Ring Management Module. It is located in the same user-defined directory that the Gateway is installed in. trmm utilizes the snmpset, snmpget, and snmpnext programs which are also released with the Gateway.

This script provides the capability to enable and disable boards and ports, and display the station table and aliases for the ports on the desired TRMM hub.

Note that the trmm script accesses the file /etc/ethers in order to support the ‘alias’ functionality. Therefore, ensure that there is an entry in /etc/ethers for every host that the trmm script may access. The syntax of the trmm command is as follows:

trmm {trmm hub} {action} {board number or “alias”} {port number}

The parameters are defined as follows:

{trmm hub} - Specifies the name of the TRMM hub in the /etc/hosts file.
{action} - Is either enable, disable, stations, reset, or restart.
{board number or “alias”} - Specifies the board to enable or disable or the word “alias.”
{port number} - Specifies the port to enable or disable.
The following examples demonstrate how NetView's RUNCMD may be used with the \texttt{trmm} script:

1. Issue RUNCMD to enable port number 2 of board number 3 on the TRMM named 'trhub1' using the service point named 'cab1p'.

`RUNCMD SP=CAB1P,APPL=UNIX,trmm trhub1 enable 3 2`

2. Issue RUNCMD to display the station table of the TRMM hub named 'trmm1' showing all stations with no aliases using the service point named 'cab1p'.

`RUNCMD SP=CAB1P,APPL=UNIX,trmm trmm1 stations`

3. Issue RUNCMD to display the station on a TRMM hub named 'trmm1' and the associated aliases using the service point named 'cab1p'.

`RUNCMD SP=CAB1P,APPL=UNIX,trmm trmm1 stations alias`

4. Issue RUNCMD to reset the counters on the TRMM hub named 'hub1' using the service point named 'cab1p'.

`RUNCMD SP=CAB1P,APPL=UNIX,trmm hub1 reset`

5. Issue RUNCMD to restart the TRMM hub named 'trhub3' using the service point named 'cab1p'.

`RUNCMD SP=CAB1P,APPL=UNIX,trmm trhub3 restart`
Chapter 3

The NVGW-Spectrum Process

Describes the NVGW-Spectrum process, including the name, synopsis, description, files, and return value.

NVGW-Spectrum Process

Name:

NVGW-Spectrum

Converts Cabletron's SPECTRUM events into SNA alerts, which can be monitored by IBM's NetView.

Synopsis:

NVGW-Spectrum [ -f config_file ] [ -b brxpu2.1_system ]
[ -vnm spectrum_system ] [ -d defaults_file ]
[ -p brxpu2.1_puname ] [ -t ] [ -x ]

Description:

NVGW-Spectrum is typically run as a background process (daemon), capturing events generated by SPECTRUM, converting these events into SNA alerts, and sending these alerts to the CNT/Brixton BrxPU2.1 SNA Server to be forwarded to an IBM mainframe where they can be monitored with NetView. NVGW-Spectrum registers with SPECTRUM to receive messages from SPECTRUM's event dispatcher function, and also attaches to the CNT/Brixton BrxPU2.1 SNA Server to gain connectivity to the SNA network.

The capture and conversion of SPECTRUM events are controlled by the NVGW-Spectrum defaults and configuration files. These files contain records which list SPECTRUM event conversions. These files contain information on how the event should be translated into a NetView alert.
The defaults file lists the default SNA vector values (see Table 3-1) provided by NVGW-Spectrum when necessary values have not been included in the configuration file. The configuration file specifies how SPECTRUM events are converted into SNA alerts.

### Table 3-1. SNA Vector Values

<table>
<thead>
<tr>
<th><strong>SNA Vector Values</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>-f config_file</td>
<td>File pathname of file containing user configured alarm to alert conversions. Default: brxspec.config.</td>
</tr>
<tr>
<td>-b brxpu2.1_system</td>
<td>Name of host system where the CNT/Brixton BrxPU2.1 SNA Server is running. Default: local system name.</td>
</tr>
<tr>
<td>-vnm spectrum_system</td>
<td>Name of host system where SpectroSERVER is running. Default: name specified in .nvrc resource file vnm_name parameter.</td>
</tr>
<tr>
<td>-d defaults_file</td>
<td>Pathname of file containing default values for SNA alert components. Default: brxspec.defaults.</td>
</tr>
<tr>
<td>-p brxpu2.1_puname</td>
<td>Name of PU to attach to for SSCP-PU control session access. Default: any available PU.</td>
</tr>
<tr>
<td>-t</td>
<td>Trace the events, states, and actions taken by NVGW-Spectrum. Trace points are logged to the brxnvgw_trace file in the process working directory. When this trace file is full, it is renamed brxnvgw_trace.1 and a new trace file is created. Default: no tracing.</td>
</tr>
<tr>
<td>-x</td>
<td>This option is used for test purposes only and provides a hexadecimal dump of SNA alert record formed from user input. Takes user input for model name, model type, event type, and severity. Finds matching mapping record and writes the resulting SNA alert to stdout. Note that when using this option, no data is passed to the CNT/Brixton BrxPU2.1 SNA Server to be processed and sent to the host.</td>
</tr>
</tbody>
</table>

After the alarm has been converted, NVGW-Spectrum sends the resulting SNA alert to the CNT/Brixton BrxPU2.1 SNA Server for transmission to the IBM mainframe. At the IBM mainframe, these alerts can be monitored using NetView’s Network Problem Determination Aid (NPDA).

To stop NVGW-Spectrum, issue the `kill -INT NVGW-Spectrum_<pid>` command. This allows NVGW-Spectrum to unregister from SPECTRUM.

**Return Value:**

Exits abnormally if an error is detected in the configuration file, if the CNT/Brixton BrxPU2.1 SNA Server cannot be attached, or if an error is detected registering with SPECTRUM.
Event Mapping

Describes how the SPECTRUM/NV-S Gateway uses its configuration files to implement alarm filtering and event-to-alert conversion.

Overview

NVGW comes pre-configured with default mappings for converting SPECTRUM events into SNA alerts. Together, the `brxspec.config` and `brxspec.defaults` files define how to convert each event received by NVGW into an SNA alert. The `brxspec.config` file describes how to convert specific SPECTRUM events into SNA alerts, and the `brxspec.defaults` file simplifies your task by providing default vector and subvector values.

By updating these files, you choose which SPECTRUM events will be converted into SNA alerts and transmitted to the IBM mainframe. Choosing which events to convert is called “filtering.” NVGW enables you to match the values in received events against values maintained in an “events to convert” list.

After accepting an event to convert, NVGW uses entries from the `brxspec.config` file to create an SNA alert. These alert definitions outline skeletons of SNA alert messages. Since you configure the alert definitions, you select the components and their values to be included in each SNA alert.

The following sections describe how you can extend NVGW to incorporate your own event-to-alert conversions.
Default Configuration

The `brxspec.defaults` file lists all the SNA Alert Subvectors and fields which you can specify in your own configuration file. For each field a default value is given. This file is referenced when a record in the `brxspec.config` file is missing mandatory values. These values are filled from the corresponding default entries in the defaults file.

The `brxspec.defaults` file contains descriptions for 10 SNA Alert Subvectors and one transparent NVGW record which allows you to build your own subvectors. These records comprise one “alert_def” record. An example defaults file is listed in Appendix B.

You can change `brxspec.defaults` file values, but do not delete records from the file.

User Configuration

Configuration file `brxspec.config` controls the filtering and event conversion provided by NVGW. It maps all the currently supported SPECTRUM events to the appropriate IBM alerts. You can change any of the mappings to meet individual system requirements.

Record Types

The `brxspec.config` record has five major record types:

- `commands` - There is one commands record in the file. It lists all the commands that the NetView operator can invoke through the NVGW. This is a mandatory record, but it can be empty.

- `queue_size` - There is one queue_size record in the file. It specifies the total number of RUNCMDs which can be outstanding at one time for this system.

- `alert_map` - There can be multiple alert_map records in the file. Each record represents one event-to-alert conversion for events matching alert_map filter criteria. The alert_map comprises one filter record and one alert_def record.

- `filter` - There is one filter record per alert_map record. It identifies SPECTRUM event instances by specifying expected values for the alarm’s fields.
alert_def - There is one alert_def record per alert_map record. It encompasses a series of records which define how to construct an SNA alert for an event which matches the filter record’s criteria. Each subordinate record represents an SNA Alert subvector. Values from the event can be inserted into the SNA alert definition; additional data can also be included.

Commands

The commands record lists the commands that are accepted by NVGW from the NetView operator:

```
commands (command1 command2....commandn)
```

where commandx is the first word in the command string. The last element of the NetView RUNCMD is the command string:

```
RUNCMD SP=SP_NAME,APPL=UNIX,UNIX_command_string
```

NVGW translates the command string from EBCDIC to ASCII, accepts the request if the command is allowed, and invokes the UNIX operating system to execute the command string. For a command to be accepted, the first word of the UNIX_command_string must match a command in the commands record.

The NVGW handles RUNCMDs through the following series of steps:

1. Translates UNIX_command_string into lower case ASCII characters.
2. Interprets the first word of UNIX_command_string as the command (white-space delineated).
3. Executes the command as a NVGW child process, enabling the command to execute with the same privileges as the gateway.
4. Invokes UNIX_command_string in “sh,” the Bourne shell.
5. Redirects output generated by the command to a temporary working file.

The command is executed as follows:

```
sh -c "(UNIX_command_string) > temp_file 2>&1"
```

Commands can be identified as either absolute or relative pathnames. Commands can be any program or shell program. For example:

- UNIX commands
- UNIX applications
- User shells
All output from the commands should be ASCII text. The following is an example of the commands entry in the `brxspec.config` file:

```plaintext
commands ( 
broadcast
cat
date
ls
mailmsg
pwd
snmpget
snmpnext
snmpset
trmm
)
```

**Queue Size**

The `queue_size` record is used to set the number of NetView operator requests that can be outstanding at one time. The maximum value of `n` is 20.

```plaintext
queue_size (n)
```

If the maximum number of simultaneous SNA network operator requests has been reached, NVGW requests subsequent requests until the queued requests are completed.

**Filtering**

With filtering, you can limit the kind and number of SNA alerts that flow to the IBM mainframe. Because NVGW must determine how to convert an event into an SNA alert, filtering is an inherent operation of NVGW. This section describes how to implement NVGW filtering.

The IBM mainframe can also filter SNA alerts, displaying only specified types of SNA alerts. NetView uses the Alert ID field in the Generic Subvector to implement filtering. This field is carried in most SNA alerts. You can configure Alert ID values to facilitate NetView's filtering. Filtering at the IBM mainframe is complicated. It is recommended that you send only important event information to NetView. Consequently, you should initially employ the NVGW filters to transmit specific event conditions.
Event Filtering

When NVGW receives an event, it initially filters by the severity defined in the NVGW resource file (.nvrc). It sequentially searches the brxscknfig file to find the appropriate alert mapping record. The “filter” record determines which event matches which alert definition. SPECTRUM events contain several fields of information about the event report. Several of these fields are used by NVGW to identify each alert:

- event type
- severity
- model name
- model type

By specifying values for these identifiers in a “filter” record, you match events to alert mapping records. The '*' character allows for wild-carding. In the following example, the record identifies an event which originates from a system named “fred,” involves the communications subsystem, and has a severity of 5:

```
filter (
    model_name = "fred"
    model_type = "coms"
    event_type = 12
    severity = 5
)
```

If a received event does not match these criteria, then NVGW continues searching for another filter which matches. If a received event does match, then NVGW uses the associated “alert_def” record to construct an SNA alert. The “alert_map” record groups the “filter” and “alert_def” records.

```
alert_map ( # alert_map: 1 filter and 1 alert_def
    filter () # filter identifies event
    alert_def() # alert_def defines SNA alert format
        # for this event matching filter
)
```

To group types of events together, use the wild-card option. For example,

```
filter (  
    model_name = "fr*d"
    model_type = "*"
)
```

identifies every event received from a system whose name begins with “fr” and ends with “d” (fred and freckled satisfy system name value).

After matching an event with an alert definition record, NVGW maps the SPECTRUM event into an SNA alert.
Alert Mapping

NVGW enables you to configure how SPECTRUM events are converted into SNA alerts. An alert_def record defines the mapping requirements for events which satisfy the associated filter criteria. You define the event-to-alert conversion in the brxspec.config file. The alert_def record removes many of the complexities inherent in formatting SNA alerts. Chapter 6 lists testing aids which aid in debugging alert mappings.

An SNA alert is transferred to the IBM mainframe via a Network Management Vector Transport (NMVT) message. As the name implies, an NMVT contains a series of vectors. The major vector is the Alert vector. The Alert vector comprises many different subvectors. Some subvectors are composed of other subvectors and subfields. Many of the subvectors employ “code points” to transfer particular information to NetView. Code points are numbers which index NetView text messages. By choosing code points, you tailor the information NetView displays to the network operator for your alert condition. The Alert message and its code points are described in detail in the Systems Network Architecture (SNA) Formats Manual.

The alert_def allows you to simply select values for the most useful subvectors. NVGW handles the formatting. Choose which subvectors you need, and fill in the appropriate values. Values can be either constants you define or variables present in the SPECTRUM event (value assignment is discussed in the next section).

Chapter 5 lists all the SNA subvectors you can include in your SNA alert. Refer to the Systems Network Architecture (SNA) Formats Manual when selecting values for these subvectors. NVGW does not validate your values, except for checking the general data-type.

All alert_map records have the structure reflected in Figure 4-1.
You can define values for the subvector fields. Alternatively, if a value is not defined for a mandatory field, NVGW will supply the default value from the `brxspec.defaults` file.

Chapter 5 lists all the SNA subvectors you can include in your alert mapping. The settable values for each subvector are also described.

**NOTE**

NVGW automatically supplies the Date/Time Subvector and the Product Set ID Subvector representing the SNA product in every SNA alert generated.

### Record Formats

This section details the assignment of values to subvector fields. Depending on the subvector field, the data-type of the defined value varies. Each subvector field, however, is coded as a single data-type. There are four data-types: numeric, string, hexstring, and record.

Vector values can also be assigned variables present in the SPECTRUM event. The data-type of the variable must match the data-type of the vector value.

### Numeric Assignments

NVGW recognizes positive decimal integers and hexadecimal numbers. A vector value with a numeric data-type uses `=` for assignment. Hexadecimal numbers are prefixed with an ‘0x.’ A numeric data-type cannot exceed 32 bits in binary form. NVGW fills vector values of the numeric data-type with the least-significant bits.
Record Formats

String Assignments

**Example:**

```
value = 2
value = 0x0002
```

**String Assignments**

Strings are offset with ‘”’ marks. NVGW converts all strings from ASCII to EBCDIC before placing the string in the SNA alert. NVGW fills vector values of the string data-type beginning with the leftmost character. The ‘\’ character is an escape character which causes NVGW to skip the next character during the string expansion. You can also include special variables in your text strings.

**Example:**

```
value = “cabletron”
value = “error condition:\nwhat happened?”  # reads: “error condition:
                         #  what happened?”
```

Hexstring Assignments

Hexstrings are offset with ‘”’ marks. NVGW moves all hexstrings into the SNA alert unmolested. White space is ignored. NVGW fills vector values of the hexstring data-type beginning with the leftmost character.

**Example:**

```
value = “0493fe00”  # 0493fe00
value = “0493 fe00 0493 fe00”
```

Records

Records are lists of zero or more assignments. A record serves to group a series of assignments. In the alert_def record, records correspond directly to subvectors and subfields. Records may be nested in records. A record's start is denoted by ‘(‘ and its end by ‘)’.

**Example:**

```
value ( )                          # no values specified
value (                          # nested record
  value_1 = 0x0493
  value_2 = “stringtime”
  value_3 (                     # nested record
    value_4 = 4
  )
)
```
Variable Assignments

NVGW makes fields in a SPECTRUM event available to you through special variables:

- \%model_name - System name of the event originator.
- \%model_type - Type of system generating the event condition.
- \%event_type - Code identifying the type of event being reported.
- \%severity - Code indicating the severity level associated with this event condition.
- \%user_name - Name of associated user.
- \%message_text - Text carrying information about the event condition.
- \%time - Time of the event condition.

All these special variables are expanded into strings and satisfy the string data-type criteria. In addition, when these variables are found within subvector text strings, they are expanded.

Example:

```plaintext
value = \%model_name                     # assign value event system name
value = "\%model_type :\%severity"      # eg. comms : 5
```

There is a limit on the amount of text that may be sent to NetView. By using these variables, you may be restricting the amount of text that will appear from the SPECTRUM event received.

You may need to modify the variable line_length (default=250) to accommodate the variable insertion.

The gateway is designed to send text to NetView in the form:

```plaintext
<xxx> - text
```

where:

- \<xxx\> - Indicates the severity of the event.
- \text - Specifies the expansion of the SPECTRUM event message. Note that this field may be truncated to build the NMVT.
Record Formats

Variable Assignments
Vector Values

Lists all the values which you can specify for an SNA alert message using the SPECTRUM/ NV-S Gateway configuration options.

Overview

As described in previous chapters, an SNA alert can encompass many different subvectors, where each subvector describes information regarding the SNA alert. NVGW enables you to easily build complex SNA alert messages. Many of the details and formatting problems inherent in building SNA alerts have been eliminated. Instead of constructing a literal SNA alert, you can choose the important values and options, and NVGW fills in the rest of the SNA alert.

You can insert 10 different types of subvectors into an SNA alert message. In addition, the transparent subvector enables you to introduce other subvectors. You define new subvectors with the transparent record, and NVGW includes it in the SNA alert without checking validity.

Refer to IBM’s System Network Architecture (SNA) Formats document for a full description of each subvector and its allowed values.
Generic

The Generic Alert Data Subvector is a mandatory subvector. This subvector includes several code points which are translated into text messages by NetView. Refer to Table 5-1 and the example below.

Example:

```plaintext
cvec.generic(
    type = 0x03          # performance
    description = 0xb000 # undetermined
    id = 0               # not specified
)
```

Basic

The Basic Alert Subvector is an optional subvector that has been replaced by the Generic Alert Data Subvector, but remains defined for older implementations of SNA. This subvector includes general alert information. Refer to Table 5-2 and the example below.

Table 5-1. General Alert Data Subvector

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>number</td>
<td>Alert Type: code point indicates severity.</td>
</tr>
<tr>
<td>description</td>
<td>number</td>
<td>Alert description code: code point describes alert condition.</td>
</tr>
<tr>
<td>id</td>
<td>number</td>
<td>Alert Id: unique identifier based on 32-bit CRC algorithm.</td>
</tr>
</tbody>
</table>

NOTE

NVGW does not calculate the alert id; however, you may specify one.

Example:

```plaintext
vec.generic(
    type = 0x03          # performance
    description = 0xb000 # undetermined
    id = 0               # not specified
)
```

Table 5-2. Basic Alert Subvector

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>number</td>
<td>Alert Type: code point indicates severity.</td>
</tr>
<tr>
<td>cause</td>
<td>number</td>
<td>General cause code: indicates generic cause of exception.</td>
</tr>
<tr>
<td>component</td>
<td>number</td>
<td>Specific Component Code: indicates generic type of component.</td>
</tr>
</tbody>
</table>
Table 5-2.  Basic Alert Subvector (Continued)

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>number</td>
<td>Alert description code: code point describes alert condition.</td>
</tr>
<tr>
<td>action</td>
<td>number</td>
<td>User Action Code: index to text screens (product dependent).</td>
</tr>
<tr>
<td>text</td>
<td>number</td>
<td>Detail text reference code: index to text screens (product dependent).</td>
</tr>
</tbody>
</table>

Example:

```ini
vec.basic (  
    type = 0x03    # performance  
    cause = 0x07   # general cause  
    component = 0x0e # component remote product  
    description = 0 # alert description  
    action = 0     # user action (prod dependent)  
    text = 0       # detail text (prod dependent)  
)
```

Product Set ID

The Product Set ID Subvector is an optional subvector which you insert into the SNA alert to identify a network component. There can be more than one Product Set ID Subvector in an SNA alert. NVGW prepends its own Product Set ID to any SNA alert transmitted to the host. Your Product Set ID Subvector follows immediately after this NVGW subvector. Your Product Set ID Subvector identifies one or more products in the network which are associated with the SNA alert. Refer to Table 5-3.

Table 5-3.  Product Set ID Subvector

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hw_product</td>
<td>record</td>
<td>Hardware Product Identifier Subvector: identifies hardware product by listing model and name.</td>
</tr>
<tr>
<td>sw_product</td>
<td>record</td>
<td>Software Product Identifier Subvector: identifies software product by listing version level and name.</td>
</tr>
</tbody>
</table>
Hardware Product Identifier Subvector

The Hardware Product Identifier Subvector uniquely describes one product. You can include several of these subvectors to identify different products. Refer to Table 5-4.

Table 5-4. Hardware Product Identifier Subvector

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>number</td>
<td>Product classification: indicates IBM or non-IBM hardware.</td>
</tr>
<tr>
<td>id</td>
<td>record</td>
<td>Hardware Product Identifier: describes machine and model type. A Product Identifier Subfield.</td>
</tr>
<tr>
<td>name</td>
<td>record</td>
<td>Hardware Product Common Name: common name. A Product Identifier Subfield.</td>
</tr>
</tbody>
</table>

Hardware Product Identifier

This subfield identifies a product uniquely by serial number (plant and sequence). This is a mandatory subfield. Refer to Table 5-5.

Table 5-5. Hardware Product Identifier

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>machine</td>
<td>string</td>
<td>Machine type: four numeric characters.</td>
</tr>
<tr>
<td>model</td>
<td>string</td>
<td>Machine model number: three upper-case alphanumeric characters.</td>
</tr>
<tr>
<td>plant</td>
<td>string</td>
<td>Plant of manufacture: two numeric characters.</td>
</tr>
<tr>
<td>sequence</td>
<td>string</td>
<td>Sequence number: seven upper-case alphanumeric characters, right-justified, with 0's fill on left.</td>
</tr>
</tbody>
</table>

SPECTRUM/NV-S Gateway provides a format type equal to 0x12.
**Emulated Product Identifier**

This subfield identifies the type of product this product emulates. This subfield is optional. Refer to Table 5-6.

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>machine</td>
<td>string</td>
<td>Machine type of product being emulated: four numeric characters.</td>
</tr>
<tr>
<td>model</td>
<td>string</td>
<td>Model number of product being emulated: three upper-case alphanumeric characters.</td>
</tr>
</tbody>
</table>

**Hardware Product Common Name**

This subfield identifies the product by its common name. This subfield is optional. Refer to Table 5-7.

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Name: up to fifteen upper-case alphanumeric characters.</td>
</tr>
</tbody>
</table>

**Software Product Identifier Subvector**

The Software Product Identifier Subvector uniquely describes one product. You can include several of these subvectors to identify different products. Refer to Table 5-8.

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>number</td>
<td>Product classification: indicates IBM or non-IBM software.</td>
</tr>
<tr>
<td>program_num</td>
<td>record</td>
<td>Software Product Program Number: serial-type information. A Product Identifier Subfield.</td>
</tr>
<tr>
<td>level</td>
<td>record</td>
<td>Software Product Common Level: version level of software. A Product Identifier Subfield.</td>
</tr>
<tr>
<td>name</td>
<td>record</td>
<td>Software Product Common Name: common name for product. A Product Identifier Subfield.</td>
</tr>
</tbody>
</table>
Software Product Program Number

This optional subfield contains a program product number used for distribution. Refer to Table 5-9.

Table 5-9. Software Product Program Number

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>program_num</td>
<td>string</td>
<td>Program product number: seven upper-case alphanumeric characters.</td>
</tr>
</tbody>
</table>

Software Product Common Level

This optional subfield describes the software version of the product. Refer to Table 5-10.

Table 5-10. Software Product Common Level

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>string</td>
<td>Common version identifier: two numeric characters, right-justified with zero fill on left.</td>
</tr>
<tr>
<td>release</td>
<td>string</td>
<td>Common release identifier: two numeric characters, right-justified with zero fill on left.</td>
</tr>
<tr>
<td>modification</td>
<td>string</td>
<td>Common modification identifier: two numeric characters, right-justified with zero fill on left.</td>
</tr>
</tbody>
</table>

Software Product Common Name

This mandatory subfield contains the software common name. Refer to Table 5-11 and the example below.

Table 5-11. Software Product Common Name

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Name: up to thirty characters.</td>
</tr>
</tbody>
</table>

Example:

```plaintext
vec.product_set {
    hw_product {
        class = 0x9  # non-IBM hardware
        id {
```
Probable Causes

The Probable Causes Subvector is a mandatory subvector. This subvector contains code points indicating the probable cause of the SNA alert. Several reason values (code points) can be listed, appearing in order of decreasing probability. Refer to Table 5-12 and the example below.

Table 5-12. Probable Causes Subvector

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reason</td>
<td>number</td>
<td>Probable cause code point: index to text information.</td>
</tr>
</tbody>
</table>

Example:

```c
vec.probable_causes (
    reason = 0x0003  # processor switch
    reason = 0x2304  # incorrect number called
)
```
User Causes

The User Causes Subvector is an optional subvector. This subvector contains code points for text information regarding probable user causes of alert conditions. Recommended actions to rectify the alert condition are also present. Additionally, you may insert additional detailed data. Refer to Table 5-13.

Table 5-13. User Causes Subvector

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user-causes</td>
<td>record</td>
<td>User Causes: probable user causes. A User Causes Subvector Subfield.</td>
</tr>
<tr>
<td>recommended-actions</td>
<td>record</td>
<td>Recommend Actions: actions to fix alert condition. A User Causes Subvector Subfield.</td>
</tr>
<tr>
<td>detailed-data</td>
<td>record</td>
<td>Detailed Data: product specific detailed data to be displayed. A User Causes Subfield.</td>
</tr>
</tbody>
</table>

User Causes Subfield

The User Causes Subfield is a mandatory subfield. It contains a list of code points representing text information describing probable causes of the alert condition. The code points are arranged in decreasing order of probability. Refer to Table 5-14.

Table 5-14. User Causes Subfield

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>number</td>
<td>User cause code point: index to text information.</td>
</tr>
</tbody>
</table>
Recommended Actions Subfield

The Recommended Actions Subfield is mandatory. It contains code points representing text information which lists recommended actions to fix the alert condition. You can list several code points (actions). Refer to Table 5-15.

Table 5-15. Recommended Actions Subfield

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>number</td>
<td>Recommended action code point: index to text information.</td>
</tr>
</tbody>
</table>

Detailed Data

The Detailed Data Subfield is optional. It contains product specific information to display. Refer to Table 5-16.

Table 5-16. Detailed Data Subfield

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>number</td>
<td>Product ID subvector code: specifies the type of Product ID subvector being indexed and the data to be extracted.</td>
</tr>
<tr>
<td>count</td>
<td>number</td>
<td>Count: index to Product ID subvector being accessed.</td>
</tr>
<tr>
<td>data_id</td>
<td>number</td>
<td>Data ID: type of data in this subfield.</td>
</tr>
<tr>
<td>encoding</td>
<td>number</td>
<td>Data encoding: how the data in this subfield is encoded.</td>
</tr>
<tr>
<td>data</td>
<td>hexstring</td>
<td>Detailed data: hexstring (maximum length is 44 bytes).</td>
</tr>
</tbody>
</table>

Product Set ID Index Subfield

The Product Set ID Index Subfield is optional. This subfield specifies a particular Product ID Subvector in this alert. Refer to Table 5-17 and the example below.

Table 5-17. Product Set ID Index Subfield

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>number</td>
<td>Product ID subvector code: specifies the type of Product ID subvector being indexed and the data to be extracted.</td>
</tr>
<tr>
<td>count</td>
<td>number</td>
<td>Count: index to Product ID subvector being accessed.</td>
</tr>
</tbody>
</table>
Example:

```plaintext
vec.user_causes (
    user_causes (code = 0x7000)       # operator intervention required
    recommended_actions (action = 0x0000) # perform problem determination
)
```

**Install Causes**

The Install Causes Subvector is an optional subvector that contains code points for text information regarding probable install causes of alert conditions. Recommended actions to rectify the alert condition are also present. Additionally, you may insert additional detailed data. Refer to Table 5-18.

**Table 5-18. Install Causes Subvector**

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>install_causes</td>
<td>record</td>
<td>Install Causes: probable install causes. An Install Causes Subvector Subfield.</td>
</tr>
<tr>
<td>recommended_actions</td>
<td>record</td>
<td>Recommend Actions: actions to fix alert condition. An Install Causes Subvector Subfield</td>
</tr>
<tr>
<td>detailed_data</td>
<td>record</td>
<td>Detailed Data: product specific detailed to data to be displayed. An Install Causes Subvector Subfield</td>
</tr>
<tr>
<td>product_set_id_index</td>
<td>record</td>
<td>Product Set ID Index: specifies subvector in Product Set ID Subvector. An Install Causes Subfield.</td>
</tr>
</tbody>
</table>

**Install Causes Subfield**

The Install Causes Subfield is a mandatory subfield. It contains a list of code points representing text information describing probable causes of the alert condition. The code points are arranged in decreasing order of probability. Refer to Table 5-19.

**Table 5-19. Install Causes Subfield**

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>number</td>
<td>Install cause code point: index to text information.</td>
</tr>
</tbody>
</table>
Recommended Actions Subfield

The Recommended Actions Subfield is mandatory. It contains code points representing text information which lists recommended actions to fix the alert condition. You can list several code points (actions). Refer to Table 5-20.

Table 5-20. Recommended Actions Subfield

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>number</td>
<td>Recommended action code point: index to text information.</td>
</tr>
</tbody>
</table>

Detailed Data

The Detailed Data Subfield is optional. It contains product specific information to display. Refer to Table 5-21.

Table 5-21. Detailed Data Subfield

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>number</td>
<td>Product ID subvector code: specifies the type of Product ID subvector being indexed and the data to be extracted.</td>
</tr>
<tr>
<td>count</td>
<td>number</td>
<td>Count: index to Product ID subvector being accessed.</td>
</tr>
<tr>
<td>data_id</td>
<td>number</td>
<td>Data ID: type of data in this subfield.</td>
</tr>
<tr>
<td>encoding</td>
<td>number</td>
<td>Data encoding: how the data in this subfield is encoded.</td>
</tr>
<tr>
<td>data</td>
<td>hexstring</td>
<td>Detailed data: hexstring (maximum length is 44 bytes).</td>
</tr>
</tbody>
</table>

Product Set ID Index Subfield

The Product Set ID Index Subfield is optional. This subfield specifies a particular Product ID Subvector in this alert. Refer to Table 5-22 and the example below.

Table 5-22. Product Set ID Index Subfield

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>number</td>
<td>Product ID subvector code: specifies the type of Product ID subvector being indexed and the data to be extracted.</td>
</tr>
<tr>
<td>count</td>
<td>number</td>
<td>Count: index to Product ID subvector being accessed.</td>
</tr>
</tbody>
</table>
Example:

```c
vec.install_causes {
    install_causes (code = 0x1300) #incorrect software generation
    recommended_actions (action = 0x1500) # correct installation problem
}
```

# Failure Causes

The Failure Causes Subvector is an optional subvector. This subvector contains code points for text information regarding probable failure causes of alert conditions. Recommended actions to rectify the alert condition are also present. Additionally, you may insert additional detailed data. Refer to Table 5-23.

## Table 5-23. Failure Causes Subvector

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>failure_causes</td>
<td>record</td>
<td>Failure Causes: probable install causes. A Failure Causes Subvector Subfield.</td>
</tr>
<tr>
<td>recommended_actions</td>
<td>record</td>
<td>Recommend Actions: actions to fix alert condition. A Failure Causes Subvector Subfield.</td>
</tr>
<tr>
<td>detailed_data</td>
<td>record</td>
<td>Detailed Data: product specific detailed to data to be displayed. A Failure Causes Subfield.</td>
</tr>
<tr>
<td>product_set_id_index</td>
<td>record</td>
<td>Product Set ID Index: specifies subvector in Product Set ID Subvector. A Failure Causes Subfield.</td>
</tr>
</tbody>
</table>

# Failure Causes Subfield

The Failure Causes Subfield is a mandatory subfield. It contains a list of code points representing text information describing probable causes of the alert condition. The code points are arranged in decreasing order of probability. Refer to Table 5-24.

## Table 5-24. Failure Causes Subfield

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>number</td>
<td>User cause code point: index to text information.</td>
</tr>
</tbody>
</table>


### Recommended Actions Subfield

The Recommended Actions Subfield is mandatory. It contains code points representing text information which lists recommended actions to fix the alert condition. You can list several code points (actions). Refer to Table 5-25.

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>number</td>
<td>Recommended action code point: index to text information.</td>
</tr>
</tbody>
</table>

### Detailed Data

The Detailed Data Subfield is optional. It contains product specific information to display. Refer to Table 5-26.

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>number</td>
<td>Product ID subvector code: specifies the type of Product ID subvector being indexed and the data to be extracted.</td>
</tr>
<tr>
<td>count</td>
<td>number</td>
<td>Count: index to Product ID subvector being accessed.</td>
</tr>
<tr>
<td>data_id</td>
<td>number</td>
<td>Data ID: type of data in this subfield.</td>
</tr>
<tr>
<td>encoding</td>
<td>number</td>
<td>Data encoding: how the data in this subfield is encoded.</td>
</tr>
<tr>
<td>data</td>
<td>hexstring</td>
<td>Detailed data: hexstring (maximum length is 44 bytes).</td>
</tr>
</tbody>
</table>

### Product Set ID Index Subfield

The Product Set ID Index Subfield is optional. This subfield specifies a particular Product ID Subvector in this alert. Refer to Table 5-27 and the example below.

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>number</td>
<td>Product ID subvector code: specifies the type of Product ID subvector being indexed and the data to be extracted.</td>
</tr>
<tr>
<td>count</td>
<td>number</td>
<td>Count: index to Product ID subvector being accessed.</td>
</tr>
</tbody>
</table>
Example:

```c
vec.failure_causes {
    failure.causes ( code = 0x6000 )        # device failure
    recommended.actions (action = 0x1400)   # restart
}
```

## Cause Undetermined

The Cause Undetermined Subvector is an optional subvector. This subvector contains recommended actions to rectify the alert condition. Additionally, you may insert additional detailed data. Refer to Table 5-28.

### Table 5-28. Cause Undetermined Subvector

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>recommended_actions</td>
<td>record</td>
<td>Recommended Actions: actions to fix alert condition. A Cause Undetermined Subvector Subfield.</td>
</tr>
<tr>
<td>detailed_data</td>
<td>record</td>
<td>Detailed Data: product specific detailed data to be displayed. A Cause Undetermined Subfield.</td>
</tr>
</tbody>
</table>

### Recommended Actions Subfield

The Recommended Actions Subfield is mandatory. It contains code points representing text information which lists recommended actions to fix the alert condition. You can list several code points (actions). Refer to Table 5-29.

### Table 5-29. Recommended Actions Subfield

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>number</td>
<td>Recommended action code point: index to text information.</td>
</tr>
</tbody>
</table>
Detailed Data

The Detailed Data Subfield is optional. It contains product specific information to display. Refer to Table 5-30.

Table 5-30. Detailed Data Subfield

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>number</td>
<td>Product ID subvector code: specifies the type of Product ID subvector being indexed and the data to be extracted.</td>
</tr>
<tr>
<td>count</td>
<td>number</td>
<td>Count: index to Product ID subvector being accessed.</td>
</tr>
<tr>
<td>data_id</td>
<td>number</td>
<td>Data ID: type of data in this subfield.</td>
</tr>
<tr>
<td>encoding</td>
<td>number</td>
<td>Data encoding: how the data in this subfield is encoded.</td>
</tr>
<tr>
<td>data</td>
<td>hexstring</td>
<td>Detailed data: hexstring (maximum length is 44 bytes).</td>
</tr>
</tbody>
</table>

Product Set ID Index Subfield

The Product Set ID Index Subfield is optional. This subfield specifies a particular Product ID Subvector in this alert. Refer to Table 5-31 and the example below.

Table 5-31. Product Set ID Index Subfield

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>number</td>
<td>Product ID subvector code: specifies the type of Product ID subvector being indexed and the data to be extracted.</td>
</tr>
<tr>
<td>count</td>
<td>number</td>
<td>Count: index to Product ID subvector being accessed.</td>
</tr>
</tbody>
</table>

Example:

```python
vec.cause_undetermined (
    recommended_actions (
        action = 0x0000 # perform problem determination
        action = 0x1702 # add link
    )
)
```
Hierarchy Resource List

The Hierarchy Resource List Subvector is optional. This subvector identifies resources which cannot be represented by SNA addresses. Several elements may be listed in hierarchical order. Refer to Table 5-32.

Table 5-32. Hierarchy Resource List Subvector

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element</td>
<td>record</td>
<td>Hierarchy Name List Subfield: one or more names.</td>
</tr>
</tbody>
</table>

Hierarchy Name List Subfield

The Hierarchy Name List Subfield is mandatory. It contains a list of names of resources in a hierarchy. The last name identifies the resource to which the alert pertains. Refer to Table 5-33 and the example below.

Table 5-33. Hierarchy Name List Subfield

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Resource name (maximum length is 8 characters).</td>
</tr>
<tr>
<td>flags</td>
<td>number</td>
<td>Flags: value of 0 means display, 0x40 means do not display.</td>
</tr>
<tr>
<td>type</td>
<td>number</td>
<td>Resource Type Identifier: category of resource.</td>
</tr>
</tbody>
</table>

Example:

```
vec.hierarchy_resource_list (  
  element (  
    name = %model_name  
    flag = 0            # display  
    type = 0            #source type unspecified
  )  
  element (  
    name = %model_type  
    flags = 0x40        # do not display  
    type = 0            # resource type unspecified
  )  
)
```

This example utilizes the variable replacement function of SPECTRUM/NV-S Gateway. The "%model_name," and "%model_type" variables will be replaced with the corresponding values in the current SPECTRUM event. If the value is not present, then no element record is produced for that value.
Self-Defining Text

The Self-Defining Text Subvector is an optional subvector. This subvector transfers text messages to NetView, where they will be displayed for the network operator. This subvector is especially useful when the code points available in other subvectors do not fully specify the alert condition. Variable insertion enables you to construct very specific text messages. Refer to Table 5-34 and the example below.

Table 5-34. Self-Defining Text Subvector

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>language</td>
<td>number</td>
<td>Generic Language Code: language identifier.</td>
</tr>
<tr>
<td>text</td>
<td>string</td>
<td>Text message: a text message.</td>
</tr>
<tr>
<td>char_set</td>
<td>number</td>
<td>Character set for text: part of Coded character set ID.</td>
</tr>
<tr>
<td>code_page</td>
<td>number</td>
<td>Code page for text: part of Coded character set ID.</td>
</tr>
<tr>
<td>country</td>
<td>number</td>
<td>Country Code: country component of the national language ID.</td>
</tr>
<tr>
<td>sender</td>
<td>number</td>
<td>Sender ID code: characterizes sender of text.</td>
</tr>
</tbody>
</table>

Example:

```plaintext
vec.self_defining_text (
    language = 0x001400   # English
    char_set = 640
    code_page = 500
    country = 0           # not specified
    sender = 0x12          # Control Program
    text =                 # Text string
        "<%severity>%message_text"
)
```

This example uses the variable replacement capabilities of SPECTRUM/NV-S Gateway. The variables (indicated by '%') will be replaced by the corresponding values in the current SPECTRUM Alarm.

Transparent

The Transparent Subvector is an optional subvector; it is not defined by SNA. The transparent subvector enables you to build your own subvectors. Specify a string of hexadecimal digits. SPECTRUM/NV-S Gateway will insert this hexstring (in binary) directly into the SNA alert. Be sure to format your subvector correctly, especially the length field. Refer to Table 5-35 and the example below.
Table 5-35.  Transparent Subvector

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hexstring</td>
<td>hexstring</td>
<td>String of hexadecimal digits.</td>
</tr>
</tbody>
</table>

**Example:**

```python
c vec.transparent (
    hexstring = "0493fe00"  # probable causes vector: undetermined
)
```

The vector breaks out as: length = 4, key =0x93, and the reason code point = 0xfe00.

**NOTE**

The vector breaks out as: length = 4, key =0x93, and the reason code point = 0xfe00.
Troubleshooting

Describes several tools used to determine and resolve problems with the SPECTRUM/ NV-S Gateway and the CNT/Brixton BrxPU2.1 SNA Server.

Overview

Several tools are provided with NVGW and with the CNT/Brixton BrxPU2.1 SNA Server that you can use to determine and resolve operational problems.

For more troubleshooting information for the CNT/Brixton BrxPU2.1 SNA Server and its SNA connections, refer to the Brixton PU2.1 SNA Server - Configuration and Network Administration Guide.

The following tools may help in problem determination:

The CNT/Brixton Message Database (BMD) - This CNT/Brixton-supplied message database documents all informational and error messages outputted by CNT/Brixton System software. A set of NVGW-specific messages, UNVW_msglib, is installed into the BMD during the Gateway installation. The CNT/Brixton on-line database utility, bmsg, is available to browse the BMD (see Appendix D). For more information about the BMD and the bmsg utility, refer to the Brixton PU2.1 SNA Server - Configuration and Network Administration Guide.

The CNT/Brixton Data Link Monitor (brxscope) - This CNT/Brixton-supplied utility can be used to monitor SNA traffic on SDLC, IBM Token Ring, and Ethernet connections. For more information on brxscope, refer to the Brixton PU2.1 SNA Server - Configuration and Network Administration Guide.
Tracing - NVGW offers an internal tracing capability as does the CNT/Brixton BrxPU2.1 SNA Server. The internal Gateway trace is discussed later in this chapter, and tracing the BrxPU2.1 SNA Server is documented in the Brixton PU2.1 SNA Server - Configuration and Network Administration Guide.

Expanded Mappings - NVGW allows you to dump the formatted SNA alert records that are mapped to the captured SPECTRUM events.

On-Line Help

The most frequently used tool is bmsg, the CNT/Brixton on-line help facility. All informational and error messages are stored in the CNT/Brixton Messages Database (BMD). You can view the message files directly or use bmsg to display information for specific messages by entering:

```
bmsg COMPxxxx
```

where COMP represents the first four characters of the message group identifier, and xxxx is the message number. For example:

```
bmsg UNVW0002
```

The information displayed about the message describes the meaning of the message and indicates the action you should take. For more information about bmsg, the on-line help facility, refer to Appendix D and the Brixton PU2.1 SNA Server - Configuration and Network Administration Guide. NVGW-specific messages in UNVW_msglib are listed in Appendix E.

NetView NPDA Filters

Most of the default alert mappings in brxspec.config create a “temp” event type. By default, NetView NPDA filters them out. To verify that the filters are working, check the status of NetView NPDA filters using the following command:

```
NPDA DFILTER AREC
```

If the “TEMP” event type is in “block” status, issue the following commands:

```
SRFILTER AREC PASS E TEMP
SRFILTER AREC PASS DEFAULT
```

This will remove the block for “TEMP” event types and also make the default action (used when event type is unknown or not in the list) pass the filters. Once your initial testing is complete, you can re-map the events in brxspec.config to another type and turn the NPDA temp filter back on.
Operator Status Information

To determine the status of the CNT/Brixton BrxPU2.1 SNA Server and its resources, including NVGW, the Brixton GUI Management interface can be used. See Establishing SNA Connectivity on Page 2-2 for information on working with the interface.

Tracing

To trace NVGW processing, choose the -t tracing option on the NVGW-Spectrum command line. As NVGW executes, trace information is recorded in a trace file named brxnvgw_trace. This trace file is limited in size. When it is full, this file is renamed brxnvgw_trace.1 and a new trace file is created. These files are created in NVGW-Spectrum’s current working directory.

The brxspectrum_trace file records a variety of different “trace points,” indicating events such as:

- Initialize NVGW
- Open the CNT/Brixton BrxPU2.1 SNA Server connection
- Detect the CNT/Brixton BrxPU2.1 SNA Server disconnect
- Register with SPECTRUM
- Unregister with SPECTRUM
- Receive SPECTRUM events
- Display event identification information
- Generate alert from event
- Dump generated alert
- Send messages to the CNT/Brixton BrxPU2.1 SNA Server
- Receive messages from the CNT/Brixton BrxPU2.1 SNA Server
- Receive unexpected messages
- Wait for SNA responses
- Wait for the CNT/Brixton BrxPU2.1 SNA Server open replies
- Wait for the CNT/Brixton BrxPU2.1 SNA Server close replies
- Detect SSCP-PU control session inactive

To turn off tracing, you must restart NVGW-Spectrum without the -t tracing option.

Logical Data Scope

The CNT/Brixton brxscope, or line trace, utility is available to monitor SNA traffic on the DLC interfaces supported by the CNT/Brixton BrxPU2.1 SNA Server. Details on using brxscope are documented in the Brixton PU2.1 SNA Server - Configuration and Network Administration Guide.
Common Problems

Table 6-1 summarizes the types of problems you may encounter while running NVGW. The table lists problem symptoms, probable causes, and corrective actions.

Table 6-1. Common Problems

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Probable Cause</th>
<th>Corrective Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>• NVGW-Spectrum unable to connect to the CNT/Brixton BrxPU2.1 SNA Server</td>
<td>The CNT/Brixton BrxPU2.1 SNA Server is not running.</td>
<td>Start the CNT/Brixton BrxPU2.1 SNA Server.</td>
</tr>
<tr>
<td>• NVGW-Spectrum unable to connect to the CNT/Brixton BrxPU2.1 SNA Server</td>
<td>Network is not up.</td>
<td>Establish network connectivity.</td>
</tr>
<tr>
<td>• NVGW-Spectrum is running on a different system than the CNT/Brixton BrxPU2.1 SNA Server</td>
<td>Incorrect hostname specified.</td>
<td>Invoke NVGW-Spectrum with the correct hostname of the remote CNT/Brixton BrxPU2.1 SNA Server. Use <code>-b </code>hostname.</td>
</tr>
<tr>
<td>• NVGW-Spectrum unable to connect to the CNT/Brixton BrxPU2.1 SNA Server</td>
<td>Services for brxadmin_pu2 and brxpu2_espd do not match on both systems.</td>
<td>Ensure <code>/etc/services</code> files match on both systems. Ensure NIS has updated services.</td>
</tr>
<tr>
<td>• NVGW-Spectrum reports PU not active.</td>
<td>SSCP-PU session not active.</td>
<td>Ask SNA systems programmer to activate the PU. Refer to the Brixton PU2.1 SNA Server - Configuration and Network Administration Guide for troubleshooting the SNA connection.</td>
</tr>
<tr>
<td>• NVGW-Spectrum reports unavailable PU</td>
<td>Incorrect pu_name specified.</td>
<td>Specify a known PU name.</td>
</tr>
<tr>
<td>• Brixton PU status is Active</td>
<td>Another application is attached to the SSCP-PU session.</td>
<td>Kill the other application or attempt to attach to another PU.</td>
</tr>
</tbody>
</table>
Expand Mappings

This facility is for testing NVGW’s alarm filtering, default value insertion, and special variable expansion. To use this facility, choose the -$x$ option on the NVGW-Spectrum command line.

Run Command Messages

The following are the Run command messages.

Message:
CNM01  DSI358I RUNCMD FAILED. ID ‘BRXPU2’ IS INVALID, SENSE CODES = X’08060000’

Cause:
Invalid SP value specified.

Action:
Specify a valid SP value.

Message:
CNM01  DSI264I RUNCMD FAILED FOR BRXPU2 - RTNCD = X’04’, FDBK2 = X’04’, SYSTEM SENSE = X’080A’, USER SENSE = X’0000’

Cause:
Invalid command specified.

Action:
Specify a valid command.

Message:
CNM01  DSI651I KEYWORD MISSING - ONE OF FOLLOWING REQUIRED: APPL

Cause:
Omitted APPL=UNIX.

Action:
Respecify, including APPL=UNIX.
Cause:
Attempted to perform multiple commands.

Action:
Perform commands one at a time.

Message:
CNM01  DSI651I KEYWORD MISSING - ONE OF FOLLOWING REQUIRED:
SP

Cause:
Omitted SP directive.

Action:
Respecify command with SP directive.

Message:
CNM01  DSI566I RUNCMD FAILED. NO SERVICE POINT COMMAND GIVEN ON THE COMMAND

Cause:
Omitted service point command.

Action:
Respecify with service point command.

Message:
No data returned.

Cause:
Generated from a redirect.

Action:
No action necessary. The command executed correctly. This message is for information only.
Sample SNA Configuration

Presents a sample portion of an SNA host network configuration.

Sample SNA Configuration

It is recommended that you give the SNA host system programmer a copy of this appendix to help coordinate the updates to the local configuration and SNA host network configuration.

The NCP/VTAM GEN lists all the SNA resources connected to an SNA Communications Controller. Four macros (see Table A-1) define the resources associated with a PU2 device emulated by the CNT/Brixton BrxPU2.1 SNA Server.

Table A-1. NCP/VTAM Macros

<table>
<thead>
<tr>
<th>MACRO</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUP</td>
<td>Specifies certain common characteristics and functions for a group of links and devices.</td>
</tr>
<tr>
<td>LINE</td>
<td>Represents the physical line connecting to the PU2 device.</td>
</tr>
<tr>
<td>PU</td>
<td>Represents the physical unit, the PU2 device.</td>
</tr>
<tr>
<td>LU</td>
<td>Represents a logical unit attached to the PU2 device.</td>
</tr>
</tbody>
</table>

The sample configuration connects to the SNA network with one Point-to-Point link and one Physical Unit Type 2 (PU2). Refer to Figure A-1.
The Brixton BrxPU2.1 SNA Server emulates a 3274 Model C Series Cluster Controller (BPU011). The UNIX workstation attaches to the SNA network via a Point-to-Point SDLC line (BLN01). Part of VTAM/NCP Generation for this example is included below.

```
GROUP MACRO

GRPBRX0 GROUP DIAL=NO,
LNCTL=SDLC,
TYPE=NCP,
ISTATUS=ACTIVE,

* LINE OPERANDS MOVED UP TO GROUP MACRO
CLOCKING=EXT,
DISCNT=NO,
SERVLIM=5,
TRANSFER=9,
SPDSEL=NO,

* PU OPERANDS MOVED UP TO GROUP MACRO
IRETRY=YES,
MAXDATA=265,
MAXOUT=7,
PASSLIN=11,

* LU OPERANDS MOVED UP TO GROUP MACRO
MODETAB=ISTINCLM,
SSCPFム=USSSCS,
USSTAB=HIS3270,
```
PACING=1,
VPACING=1

******************************************************************
*                     LINE MACRO                           *
*                                                            *
******************************************************************
BLN01    LINE     ADDRESS=(01,FULL),
          SPEED=9600,
          NRZI=NO,
          DUPLEX=FULL

******************************************************************
*                     PU MACRO                             *
*                                                            *
******************************************************************
BPU011   PU       ADDR=C1,
          PUTYPE=2
The `brxspec.defaults` file lists all the subvectors and subvector values which can be set by the user. Each of the subvector values is given a default value. NVGW references these values to flesh out an SNA alert which the user has not fully configured.

The `brxspec.defaults` file is listed as follows:

```plaintext
# This file is integral to the operation of the SPECTRUM/NV-S Gateway. No # entries should ever be deleted. You may, however, change the default # values if you wish.
# alert_def (   vec.generic (       type = 0x02       # performance       description = 0xfe00       # undetermined       id = 0x43000000       # not specified   )
```
vec.basic (
    type = 0x02 # temporary loss
    cause = 0x07 # general cause
    component = 0x0e # component remote product
    description = 0 # alert description (prod dependent)
    action = 0 # user action (prod dependent)
    text = 0 # detail text (prod dependent)
)
vec.probable_causes (reason= 0xfe00) # undetermined
vec.user_causes (
    user_causes ( code = 0x7000 ) # operator intervention required
    recommended_actions (action=0x0000) # perform problem determination
    detailed_data (
        code = 0x02 # hardware product common name
        count = 1 # first Product ID subvector
        data_id = 0x07 # error code
        encoding = 0x11 # coded graphic char set 00640-00500 +
        data = "40404040404040" # a hexstring (EBCDIC spaces)
    )
    product_set_id_index (
        code = 0x02 # hardware product common name
        count = 1 # first Product ID subvector
    )
)
vec.install_causes (
    install_causes ( code = 0x1300 ) # incorrect software generation
    recommended_actions (action=0x0000) # perform problem determination
    detailed_data (
        code = 0x02 # hardware product common name
        count = 1 # first Product ID subvector
        data_id = 0x07 # error code
        encoding = 0x11 # coded graphic char set 00640-00500 +
        data = "40404040404040" # a hexstring (EBCDIC spaces)
    )
    product_set_id_index (
        code = 0x02 # hardware product common name
        count = 1 # first Product ID subvector
    )
)
vec.failure_causes (
    failure_causes ( code = 0x6000 ) # device failure
    recommended_actions (action=0x0000) # perform problem determination
    detailed_data (
        code = 0x02 # hardware product common name
        count = 1 # first Product ID subvector
        data_id = 0x07 # error code
        encoding = 0x11 # coded graphic char set 00640-00500 +
        data = "40404040404040" # a hexstring (EBCDIC spaces)
    )
    product_set_id_index (
        code = 0x02 # hardware product common name
        count = 1 # first Product ID subvector
    )
)
vec.cause_undetermined (
    recommended_actions (action=0x0000) # perform problem determination
    detailed_data (
        code = 0x02 # hardware product common name
        count = 1 # first Product ID subvector
        data_id = 0x07 # error code
    )
Encoding = 0x11 # coded graphic char set 00640-00500 +
data = “40404040404040” # a hexstring (EBCDIC spaces)

Product_set_id_index {
  code = 0x02 # hardware product common name
  count = 1 # first Product ID subvector
}

# variable names for %model_name, %event_type, %severity_and %message_text
# may be used in the text string. Note, however, that the max_msg_text size
# in the file .nvrc must be reduced to accommodate these variables; e.g., text =
# “%model_name: event = %event_type severity = %severity_msg = %message_text”

Self_defining_text {
  language = 0x001400 # English
  char_set = 640
  code_page = 500
  country = 0 # not specified
  sender = 0x12 # Control Program
  text = “<%severity>-%message_text”
}

Hierarchical_resource_list {
  complete=0x80
  element {
    name = %model_name
    flags = 0 # display
    type =0 # resource type unspecified
  }
}

Product_set {
  hw_product {
    class = 0x9 # non-IBM hardware
    id {
      machine = “0000”
      model = “000”
      plant = “00”
      sequence = “0000000”
    }
    name {
      name = “Sun Server”
    }
    emulated_product_id {
      machine = “0000”
      model = “000”
    }
  }
  sw_product {
    class = 0xc # non-IBM software
    name {
      name = “spectrum”
    }
    progran_num {
      progran_num = “0000000”
    }
    level {
      version = “00”
      release = “00”
      modification = “02”
    }
  }
}
Appendix C

brxspec.config File

Describes the brxspec.config file

Overview

The brxspec.config file describes the filtering and mapping of SPECTRUM events into SNA alerts. An SNA alert is formatted as a special Network Management Vector Transport (NMVT) message. The alert_map record describes the appropriate event-to-alert conversion for events which meet its filter criteria.

There can be many alert_map records. NVGW will check each SPECTRUM event it receives with the filter record data of each alert_map. After finding a match, NVGW uses the corresponding alert_def record to formulate an SNA alert.

The filter record compares any listed model name, model type, event type, or severity against the values received in a SPECTRUM event.

A user can specify all the particulars of an SNA alert construction. Alternatively, a user can utilize some or all of the default values specified in the brxspec.defaults file.

An example brxspec.config file is listed as follows:

```plaintext
# brxspec.config
#
# pre-packaged SPECTRUM/NV-S Gateway configuration file.
# You should add alert_map records as you begin to build different SNA alert messages.
#
# commands
#
# commands the NetView operator can issue using the RUNCMD.
#
```
commands (  
  broadcast  
cat  
date  
ls  
mailmsg  
pwd  
snmpget  
snmpnext  
snmpset  
timm  
)
#
# queue_size
#
# Maximum number of outstanding RUNCMD requests.
# queue_size = (15)
#
# alert_map
#
# Here is a very simple alert_map record. As you build
# different SNA alert messages, these records can become
# quite complex.
#
# What does this alert_map record accomplish? Well, the filter
# record indicates that this alert_map is for any SPECTRUM alarm
# received -- '*' is a wildcard.
#
# The alert_def record is pretty sparse: it specifies that the
# generic, product set id, probable causes, cause undetermined,
# self-defining text, and hierarchy resource list subvectors
# are to be included. All the values for these vectors, however, are
# defaulted. The brxspec.defaults file specifies these defaulted
# values.
#
# The resulting SNA alert carries a lot of significant information
# to the IBM mainframe and NetView: the time, the system resource
# having a problem, and a string specifically describing the
# problem.
#
alert_map (  
  filter (  
    model_name = "*"  
  )  
)  

alert_def (  
  vec.generic ( )  
  vec.product_set ( hw_product() sw_product() )  
  vec.probable_causes ( )  
  vec.cause_undetermined ( )  
  vec.self_defining_text ( )  
  vec.hierarchy_resource_list ( )  
)  
)
Appendix D

Messages

Describes the CNT/ Brixton Message Database, BMD, the general format of a message, and bmsg, the CNT/ Brixton on-line help facility.

Overview

The CNT/Brixton Message Database (BMD) is installed with the CNT/Brixton BrxPU2.1 SNA Server. NVGW installs a set of Gateway-related messages, UNVW_msglib, into the BMD when it is installed. All informational and error messages generated by the CNT/Brixton BrxPU2.1 SNA Server and some messages generated by NVGW are derived from the BMD. Product components running in "user space" access the BMD during run-time. Product components running in "kernel space" access message tables that are constructed at compilation time.

The message handling scheme provides the following advantages:

• All messages generated by the CNT/Brixton BrxPU2.1 SNA Server and 3270 are documented. When a message is generated, users have a reference that describes the reason the message was generated, its impact, and how to respond.

• On-line message documentation. BMD messages are standard UNIX text files and may be browsed using normal system editors and utilities, such as grep. A CNT/Brixton application called bmsg is available and, when given a message number, prints the full content of the message.

• Configurable message formats. Users are free to modify the message content as they wish.

• Consistent message handling for all CNT/Brixton Systems SNA End Node products.

BMD messages specific to NVGW are listed in Appendix E.
CNT/Brixton Message Display Utility - bmsg

bmsg is invoked as follows:

bmsg [message_number [, message_number...]] [-v]

where:

message_number – Specifies the message to display.

-v – Requests bmsg to print the current version number, and exit. -v overrides all other options.

If no message_number is given, bmsg prompts the user and displays the requested message. This continues until the user types “Q” to quit. An example of bmsg operation follows:

% bmsg
Please enter message identifier (Q to quit): ABCD0001

MESSAGE: ABCD0001
%s This is an example message string which starts with a string parameter.

PARAMETERS:
1. A string (%s)

CAUSE:
This field normally explains what condition(s) causes this message to be displayed.

EFFECT:
This field normally displays what effect the condition(s) may have had.

ACTION:
This field normally indicates what actions should be taken when this message is displayed.
Please enter message identifier (Q to quit):Q

CNT/Brixton Message Database (BMD)

The CNT/Brixton BMD is a UNIX directory. The CNT/Brixton BMD directory contains several message files, each representing one product component. Each product component owns a range of message numbers and stores message information in its own message file. The file is uniquely named xxxx.msglib, where xxxx is the product component identifier.
The CNT/Brixton BMD directory may be located anywhere on the user’s search path, as specified by the environment variable $PATH. If you wish to modify a message file, copy it out of the BMD directory into the current working directory. If $PATH includes the current directory “.”, the message file in the current working directory will be accessed, rather than the version in the BMD directory.

**Message Files**

A message library is a text file containing multiple message entries. Each message has up to seven defining fields:

- Message number
- Message string
- Message string parameters
- Message display options
- Cause
- Effect
- Action

Message fields are demarcated by blank lines. Messages are separated by a full line of hyphens (“-”). An example message entry is listed as follows:

MESSAGE:
ABCD0002Line %s has reached the retry limit (%d)

PARAMETERS:
1. LINE name (%s)
2. Retry limit (%d)

CAUSE:
The specified LINE has attempted to retransmit a message retry limit times.

EFFECT:
LINE moves into disconnected state.

ACTION:
Check the configured MAXDATA size for the line. If you are transmitting oversize frames, they will be rejected.

Use the Operator to examine the line statistics. If the line condition is poor, it may be necessary to increase the retry limit or increase the retry timeout value.
Message Field

The MESSAGE field contains the message number and its associated string. The message number, for example, ABCD0001, is fixed. The message string is similar to a “printf” formatted string. The order of the parameters in the string is fixed. The following parameter insertion values are defined:

%<i>s</i> - Inserts a string <tt>(printf(“%s”))</tt>.
%<i>d</i> - Inserts a decimal integer <tt>(printf(“%d”))</tt>.
%<i>x</i> - Inserts a hexadecimal integer <tt>(printf("0x%08x"))</tt>.
%% - Escapes the '%' so this character can be included in string <tt>(printf("%\%"))</tt>.

The message string can be up to 1016 characters long and cannot contain any blank lines.

Parameters Field

The parameters in the message string are described in the PARAMETERS field. Each parameter is defined on one line. This field can be up to 1024 characters and cannot contain any blank lines.

Options Field

The OPTIONS field describes how the messages should be displayed. Options may include:

**NO_PRINT** - Do not display this message.
**NO_MESSAGE_NUMBER** - Do not prefix the message with its number.
**PROGRAM_NAME** - Prefix the message by the program name.
**PROCESS_ID** - Prefix the message by the user process id.
**TIME** - Prefix the message by the date and time.
Cause Field

The CAUSE field describes why the message is displayed. This field can be up to 1024 characters and cannot contain any blank lines.

Effect Field

The EFFECT field describes the effect on the product component due to the condition described by the associated messages. This field can be up to 1024 characters and cannot contain any blank lines.

Action Field

The ACTION field describes the actions the user or administrator should take when the associated message is displayed. This field can be up to 1024 characters and cannot contain any blank lines.
UNIX to NetView (UNVW) Messages

Lists the UNVW_msglib file, containing the SPECTRUM/NV-S Gateway-specific UNIX-to-NetView (UNVW) messages.

Message:
UNVW0000 @(#)UNVW_msglib 3.1.0 5/12/95

Cause:
This message is never displayed. It reflects the current version of the UNVW_msglib under the Cabletron Development Environment.

Message:
UNVW0001 unable to open defaults file

Cause:
Event mapper could not open your defaults file. Either it could not find the file or did not have read permissions.

Effect:
Event mapper exits.

Action:
1. Specify pathname of the defaults file using the command line option “-d”.

2. Move the defaults file to the current working directory and name brxspec.defaults -- the default defaults file name.
3. Check the permissions on the defaults file. Event mapper must have read permissions.

Message:
UNVW0002 expected default alert_def record

Cause:
Event mapper did not find the default alert_def in your defaults file.

Effect:
Event mapper exits.

Action:
Put an alert_def record in your defaults file. The defaults file must contain all entries which can be selected for the SNA alert values. Therefore, you should make only minimal, cosmetic changes to the defaults file.

Message:
UNVW0003 no default mappings

Cause:
Event mapper did not find the default alert_def in your defaults file.

Effect:
Event mapper exits.

Action:
Put an alert_def record in your defaults file. The defaults file must contain all entries which can be selected for the SNA alert values. Therefore, you should make only minimal, cosmetic changes to the defaults file.

Message:
UNVW0004 unable to open configuration file

Cause:
Event mapper could not open your configuration file. Either it could not find the file or did not have read permissions.

Effect:
Event mapper exits.
Action:
1. Specify pathname of the configuration file using the command line option “-f”.
2. Move the configuration file to the current working directory and name brxspec.config -- the default configuration file name.
3. Check the permissions on the configuration file. Event mapper must have read permissions.

Message:
UNVW0005 expected commands record

Cause:
Event mapper did not find the commands record in your configuration file.

Effect:
Event mapper exits.

Action:
The commands record is mandatory. Place the commands record immediately after the agents record in the configuration file. If no agents record is specified, place the commands record first.

Message:
UNVW0006 expected queue_size record

Cause:
Event mapper did not find the queue_size record in your configuration file.

Effect:
Event mapper exits.

Action:
The queue_size record is mandatory. Place the queue_size record immediately after the commands record.

Message:
UNVW0007 expected alert_map record

Cause:
Event mapper did not find an expected alert_map record record in your configuration file.
Effect:
Event mapper exits.

Action:
Check the parentheses on all alert_map records. After each alert_map record, another alert_map record must start (check spelling) or the configuration file must end.

---

Message:
UNVW0007 expected alert_map record

Cause:
Event mapper did not find an expected alert_map record record in your configuration file.

Effect:
Event mapper exits.

Action:
Check the parentheses on all alert_map records. After each alert_map record, another alert_map record must start (check spelling) or the configuration file must end.

---

Message:
UNVW0008 max alert_map records exceeded (%d)

Parameters:
1. maximum allowed alert_maps (string).

Cause:
Event mapper detected more than the allowed maximum for alert_maps in your configuration.

Effect:
Event mapper only uses the alert_map records up to the maximum allowed one configured.

Action:
Contact Cabletron Support if you require a larger number of alert_map records.

---

Message:
UNVW0009 default definition of %s missing

Parameters:
1. vector name (string).
Cause:
Event mapper could not find its own internal representation of a vector.

Effect:
Event mapper exits.

Action:
Internal error. Contact Cabletron Support.

Message:
UNVW000A symbol type unknown setting defaults

Cause:
Event mapper does not understand internally defined types.

Effect:
Event mapper does not process unknown type.

Action:
Internal error. Contact Cabletron Support.

Message:
UNVW000B assignment expected

Cause:
Event mapper expected an assignment statement, either an "=" sign or a subordinate record.

Effect:
Event mapper exits.

Action:
Correct the file in error.

Message:
UNVW000C symbol expected

Cause:
Event mapper expected symbolic name.

Effect:
Event mapper exits.

Action:
Correct the file in error.
Message: UNVW000D unknown attribute: %s

Parameters:  
1. attribute name (string).

Cause:
Event mapper encountered an unknown attribute name. Event mapper has no listing of the specified attribute for the vector being processed.

Effect:
Event mapper exits.

Action:
Correct the file in error.
1. Check spelling.
2. Check for attribute for same vector in defaults file or refer to Event mapper documentation.

Message: UNVW000E incompatible type for attribute %s

Parameters:  
1. attribute name (string).

Cause:
Event mapper detected a type conflict while assigning the specified attribute a value.

Effect:
Event mapper exits.

Action:
Correct the file in error.

Message: UNVW000F symbol type unknown setting assignment

Cause:
Event mapper does not understand internally defined types.

Effect:
Event mapper does not process unknown type.

Action:
Internal error. Contact Cabletron Support.
Message:
UNVW0010 expected agent list

Cause:
Event mapper expected a list of agents enclosed by "(" and ")".

Effect:
Event mapper exits.

Action:
Correct the file in error.

Message:
UNVW0011 error parsing agent %s

Parameters:
1. agent name (string).

Cause:
Event mapper detected error while parsing specified agent’s values.

Effect:
Event mapper exits.

Action:
Correct the file in error.

Message:
UNVW0012 expected agent value

Cause:
Event mapper expected agent names in the agent list.

Effect:
Event mapper exits.

Action:
Correct the file in error.

Message:
UNVW0013 expected commands list

Cause:
Event mapper expected a list of commands enclosed by "(" and ")".
Effect:
Event mapper exits.

Action:
Correct the file in error.

---------------------------------------------

Message:
UNVW0014 expected command value

Cause:
Event mapper expected command names in the command list.

Effect:
Event mapper exits.

Action:
Correct the file in error.

---------------------------------------------

Message:
UNVW0015 expected queue_size list

Cause:
Event mapper expected a list of queue_sizes enclosed by "(" and ")".

Effect:
Event mapper exits.

Action:
Correct the file in error.

---------------------------------------------

Message:
UNVW0016 expected queue_size value

Cause:
Event mapper expected queue_size numbers in the queue_size list.

Effect:
Event mapper exits.

Action:
Correct the file in error.

---------------------------------------------

Message:
UNVW0017 specified queue size not allowed
Cause:
Event mapper detected an illegal queue size value.

Effect:
Event mapper exits.

Action:
Correct the file in error.

Message:
UNVW0018 expected alert mapping components

Cause:
Event mapper expected a list of alert_map components enclosed by "(" and ")".

Effect:
Event mapper exits.

Action:
Correct the file in error.

Message:
UNVW0019 expected filter record

Cause:
Event mapper did not find an expected filter record record in your alert_map record.

Effect:
Event mapper exits.

Action:
Add filter record to your alert_map record.

Message:
UNVW001A expected alert_def record

Cause:
Event mapper did not find an expected alert_def record record in your alert_map record.

Effect:
Event mapper exits.

Action:
Add alert_def record to your alert_map record.
Message:
UNVW001B alert_map record unterminated

Cause:
Event mapper did not find an expected ")" to terminate the alert_map record.

Effect:
Event mapper exits.

Action:
Correct the file in error.

Message:
UNVW001C expected filter components

Cause:
Event mapper expected a list of filter components enclosed by "(" and ")".

Effect:
Event mapper exits.

Action:
Correct the file in error.

Message:
UNVW001D unknown filter attribute: %s

Parameters:
1. attribute name (string).

Cause:
Event mapper encountered an unknown attribute name. Event mapper has no listing of the specified attribute for filter records.

Effect:
Event mapper exits.

Action:
Correct the file in error.

1. Check spelling.
2. Check for allowed filter attribute in Event mapper documentation.
Message:
UNVW001E incompatible type for filter attribute %s

Parameters:
1. attribute name (string).

Cause:
Event mapper detected a type conflict while assigning the specified filter attribute a value.

Effect:
Event mapper exits.

Action:
Correct the file in error.

Message:
UNVW001F expected filter attribute

Cause:
Event mapper expected a filter attribute in the record being processed.

Effect:
Event mapper exits.

Action:
Correct the file in error.

Message:
UNVW0020 expected alert_def components

Cause:
Event mapper expected a list of alert_def components enclosed by "(" and ")".

Effect:
Event mapper exits.

Action:
Correct the file in error.

Message:
UNVW0021 expected vector name

Cause:
Event mapper expects vector records in the alert_def record.
Effect:
Event mapper exits.

Action:
Correct the file in error.

________________________________________________

Message:
UNVW0022 unknown vector (%s) defined in alert_def record

Parameters:
1. vector name (string).

Cause:
Event mapper encountered an unknown vector name. Event mapper has no listing of the specified vector.

Effect:
Event mapper exits.

Action:
Correct the file in error.

1. Check spelling.
2. Check for allowed vectors in Event mapper documentation.

________________________________________________

Message:
UNVW0023 error parsing the %s vector

Parameters:
1. vector name (string).

Cause:
Event mapper detected an error while parsing the specified vector.

Effect:
Event mapper exits.

Action:
Correct the file in error.

________________________________________________

Message:
UNVW0024 default definition of %s missing in %s list.

Parameters:
1. attribute name (string).
2. object name (string).
Cause:
Event mapper is missing default definitions for vector values.

Effect:
Event mapper exits.

Action:
If you change the defaults file, be careful. No entries can be deleted. You may, however, change the values assigned to any of the attributes.

Message:
UNVW0025 default definition of %s vector missing

Parameters:
1. vector name (string).

Cause:
Event mapper is missing default definitions for the specified vector.

Effect:
Event mapper exits.

Action:
If you change the defaults file, be careful. No entries can be deleted. You may, however, change the values assigned to any of the attributes.

Message:
UNVW0026 list_append got NULL list

Cause:
Event mapper detected error trying to append to a non-existent list.

Effect:
Event mapper continues.

Action:
Contact Cabletron Support.

Message:
UNVW0027 unknown type in deallocate cell

Cause:
Event mapper detected an unknown data type when deallocating cell.
Effect:
Event mapper continues.

Action:
Contact Cabletron Support.

Message:
UNVW0028 unknown symbol in match_event_filter

Cause:
Event mapper detected an unknown symbol name when comparing event against filter.

Effect:
Event mapper continues.

Action:
Contact Cabletron Support.

Message:
UNVW0029 unknown symbol in alert builder

Cause:
Event mapper detected an unknown symbol name when building an SNA alert.

Effect:
Event mapper continues.

Action:
Contact Cabletron Support.

Message:
UNVW002A select error while waiting for an SNA response

Cause:
Event mapper encountered an error while executing the select system call.

Effect:
Event mapper continues.

Action:
Contact Cabletron Support.
Message:  
UNVW002B unable to open the brxpu2.1 interface

Cause:  
Event mapper could not connect to the Brixton BrxPU2.1 SNA Server.

Effect:  
Event mapper cannot send or receive any SNA messages until the connection to the Brixton BrxPU2.1 SNA Server is established. Event mapper continues to attempt to reconnect every 30 seconds.

Action:  
1. Start the Brixton BrxPU2.1 SNA Server.  
2. Check that remote Brixton BrxPU2.1 SNA Server on the remote host can be reached: try ping.  
3. Check that /etc/services file entries match for both systems (brxadmin_pu2 and brx_pu2_espd).

Message:  
UNVW002C select error while waiting for an open reply from the Brixton BrxPU2.1 SNA Server

Cause:  
Event mapper encountered an error while executing the select system call.

Effect:  
Event mapper continues.

Action:  
Contact Cabletron Support.

Message:  
UNVW002D the Brixton BrxPU2.1 SNA Server interface disconnected

Cause:  
Event mapper’s connection to the Brixton BrxPU2.1 SNA Server has terminated.

Effect:  
Event mapper cannot send or receive any SNA messages until the connection to the Brixton BrxPU2.1 SNA Server is established. Event mapper continues to attempt to reconnect every 30 seconds.
Action:
1. Start the Brixton BrxPU2.1 SNA Server.
2. Check that the Brixton BrxPU2.1 SNA Server on the remote host can be reached: try ping.

________________________________________________

Message:
UNVW002E Unavailable PU: %s, Port = %d

Parameters:
1. pu name (string).
2. port number (decimal number).

Cause:
Event mapper unable to connect to the specified PU. The port is already in use or does not exist.

Effect:
Event mapper cannot send or receive any SNA messages until the connection to the Brixton BrxPU2.1 SNA Server is established. Event mapper continues to attempt to reconnect every 30 seconds.

Action:
1. Ensure no other applications (including Event mapper) are using the PU port.
2. Check if PU exists in the Brixton BrxPU2.1 SNA Server configuration.
3. Specify another PU.

________________________________________________

Message:
UNVW002F Open PU, internal error = %d

Parameters:
1. error (decimal number).

Cause:
Event mapper unable to connect to the specified PU. Internal error reported.

Effect:
Event mapper cannot send or receive any SNA messages until the connection to the Brixton BrxPU2.1 SNA Server is established. Event mapper continues to attempt to reconnect every 30 seconds.
Action:
Contact Cabletron Support.

Message:
UNVW0030 PU not active on open reply

Cause:
Event mapper connected to Brixton BrxPU2.1 SNA Server, but, the SSCP-PU session is inactive.

Effect:
Event mapper disconnects from the Brixton BrxPU2.1 SNA Server. Event mapper cannot send or receive any SNA messages until the connection to the Brixton BrxPU2.1 SNA Server is established and the SSCP-PU Session is active. Event mapper continues to attempt to reconnect every 30 seconds.

Action:
1. Request the SNA Host system programmer to activate the PU.
2. Refer to the Brixton BrxPU2.1 SNA Server troubleshooting documentation.

Message:
UNVW0031 error on write to the Brixton BrxPU2.1 SNA Server: errno=%d, brx_errno=%x

Parameters:
1. system error (decimal number).
2. brixton error (decimal number).

Cause:
Event mapper detected error when sending message to the Brixton BrxPU2.1 SNA Server.

Effect:
Event mapper does not send the message.

Action:
Restart Event mapper and the Brixton BrxPU2.1 SNA Server. Messages lost on the SSCP-PU session can cause the session to lock-up.

Message:
UNVW0032 error on closing connection to the Brixton BrxPU2.1 SNA Server: errno=%d, brx_errno=%x
Parameters:
1. system error (decimal number).
2. brixton error (decimal number).

Cause:
Event mapper detected error when closing the Brixton BrxPU2.1 SNA Server connection.

Effect:
Event mapper continues.

Action:
None. Event mapper only closes the connection during termination proceedings. The system will clean-up only outstanding resources.

Message:
UNVW0033 select error while waiting for close reply from the Brixton BrxPU2.1 SNA Server

Cause:
Event mapper encountered an error while executing the select system call.

Effect:
Event mapper continues.

Action:
Manually terminate Event mapper.

Message:
UNVW0034 error on sending an SNA response

Cause:
Event mapper detected error on sending an SNA response to the Brixton BrxPU2.1 SNA Server.

Effect:
Event mapper continues.

Action:
Restart Event mapper and the Brixton BrxPU2.1 SNA Server. Messages lost on the SSCP-PU session can cause the session to lock-up.

Message:
UNVW0035 can’t open file runcmd reply file %s
Parameters:
1. working file name (string).

Cause:
Event mapper cannot access a temporary working file in the current working directory.

Effect:
Event mapper continues. Runcmd replies are not transmitted back to NetView.

Action:
Change the permissions on the current working directory to give write permissions to the Event mapper application.

Message:
UNVW0036 error: unable to execute command %s

Parameters:
1. working file name (string).

Options:
NO_MESSAGE_NUMBER

Cause:
Event mapper cannot access a temporary working file in the current working directory.

Effect:
Event mapper continues. Runcmd replies are not transmitted back to NetView.

Action:
Change the permissions on the current working directory to give write permissions to the Event mapper application.
A
Alert mapping
- brxspec.defaults file 4-7
- defining subvector values 4-7
- process description 4-6
- structure records 4-6

B
Basic alert subvector
- values/type/description, table 5-2
- brxspec.config file
code example C-1
overview C-1
- brxspec.defaults file
code example B-1
overview B-1

C
Cause undetermined subvector
- fix alert condition 5-14
- values/type/description, table 5-14
Configurable mapping
- alert mapping record 1-4
- configuration instructions 1-4
- list of subvectors supported 1-4
Control Functionality
- components 1-3

D
Default mapping
- assignable values 1-4
- subvector values 1-4
Detailed data
- optional subfield 5-9
- product specific information 5-11
- values/type/description, table 5-9, 5-11

E
Emulated product identifier
- identifying product type 5-5
- values/type/description, table 5-5
Event conversion
- alert-to-conversion instructions 1-3
- reading configuration files 1-3
- writing to memory resident tables 1-3
Event filtering
- alert fields, list 4-5
- events forwarded to NetView 1-5
- factors for setting 1-5
- list of factors 1-5
- matching events, criteria 4-5
- two levels available 1-5
Event mapping
- default configuration 4-2
- overview 4-1
Expand mappings
- testing NVGW’s alarm filter 6-5
- x command line option 6-5

F
Failure causes subfield
- probable cause of alert 5-12
- values/type/description, table 5-12
Failure causes subvector
- probable failure causes 5-12
- values/type/description, table 5-12
**G**

Generic alert data
- generic alert data subvector, table 5-2
- subvector 5-2
- subvector descriptions 5-2

**H**

Hardware product common name
- identifying product name 5-5
- values/type/description, table 5-5

Hardware product identifier subvector
- identifying products 5-4

Hierarchy name list subfield
- list of resources in hierarchy 5-16
- values/type/description, table 5-16

Hierarchy resource list subvector
- identifies resources 5-16
- values/type/description, table 5-16

**I**

Install causes
- values/type/description, table 5-10

Install causes subfield
- probable cause of alert 5-10
- values/type/description, table 5-10

Install causes subvector
- install causes of alert 5-10

**L**

Logical data scope
- brxscope utility 6-3
- monitor utility, SNA traffic on DLC interfaces 6-3

**M**

Message files
- defining fields, list D-3

Messages
- action field D-5
- bmsg, CNT/Brixton message display utility, description D-2
- bmsg, on-line help facility D-1
- cause field D-5
- CNT/Brixton BMD message file, xxxx_msglib D-2
- CNT/Brixton message database (BMD) D-2
- CNT/Brixton message database (BMD), description D-1
- effect field D-5
- options field, list D-4
- overview D-1
- parameter insertion values, list D-4
- parameters field, defined D-4

**N**

NetView Alert
- checking display 2-6
- procedure to view alerts 2-6

Notice
- NVGW-Spectrum product support 2-9
- source code acknowledgement (MIT) 2-9

**O**

On-Line help
- bmsg, CNT/Brixton on-line help 6-2

Operator status
- brxop display, code example 6-3
- brxop, display local SNA resources 6-3

**P**

Probable causes subvector
- code indicating SNA alert cause 5-7
- values/type/description, table 5-7

Product Set ID
- defined 5-3
- values/type/description, table 5-3

Product Set ID Index Subfield
- alert subvector 5-11

Product set ID index subfield
- particular alert subvector 5-15
- particular product set ID 5-13
- specifying particular product 5-9
- values/type/description, table 5-9, 5-11, 5-13, 5-15
Recommended actions subfield
- alert probable causes 5-13
- fix alert condition 5-14
- list of actions to fix alert 5-11
- recommended alert fix 5-9
- values/type/description, table 5-9, 5-11, 5-13, 5-14

Record formats
- hexstring assignments 4-8
- numeric assignments 4-7
- records 4-8
- string assignments 4-8

Records
- subvectors, subfields, correspondence 4-8

Restricted Rights Notice ii

Run command messages
- message/cause/action, list 6-5

RUNCMD
- executing NetView command 2-6
- NetView display response 2-6
- trmm script, examples 2-14

RUNCMD scripts
- CLI support 2-8
- example of CLI usage 2-8
- invoking UNIX commands 2-7
- invoking user defined scripts or programs 2-7
- sending message to logged-in users 2-7
- sending UNIX mail 2-8

RUNCMD support
- adding commands 1-6
- list of supported commands and utilities 1-6
- passing commands to UNIX system 1-6

Self-defining text subvector
- transfer messages to NetView 5-17
- values/type/description, table 5-17

SNA configuration, sample
- example, illustration A-2
- NCP/VTAM macros A-1
- SNA resources/SNA communications controller A-1
- VTAM/NCP generation, code example A-2

SNA connectivity
- establishing 2-2
- using BrxPU2.1 to transfer alerts 2-2

SNMP commands
- snmpget 1-1
- snmpnext 1-1
- snmpset 1-1

snmpget
- RUNCMD command from host 2-9
- table of command parameters 2-9

snmpnext
- RUNCMD command from IBM host 2-11
- table of command parameters 2-11

snmpset
- RUNCMD command from IBM host 2-12
- table of command parameters 2-12

Software Product Common Level
- identifying software product version 5-6
- Software product common level values/type/description, table 5-6

Software product common name
- mandatory subview 5-6
- values/type/description, table 5-6

Software product identifier subvector
- describing one product 5-5
- describing unique product 5-5
- values/type/description, table 5-5

Software product program number
- identifying program numbers 5-6
- values/type/description, table 5-6

SPECTRUM/NV-S Gateway
- common problems, table 6-4
- converting alerts into SNA alerts 1-6
- definition 1-1
- implementing by NVGW-Spectrum process 1-6
- invoking manually, procedure 2-3
- invoking through SPECTRUM process 2-5
- SPECTRUM/NV-S Gateway defaults
- establishing values for defined event 1-7
- refer to brxspec.config file 1-7