CA Spectrum®

Southbound Gateway Toolkit Guide

Release 9.3

CA Technologies Product References

This document references the following CA Technologies products:

CA Spectrum® (CA Spectrum)

CA Spectrum® CA Spectrum Southbound Gateway Toolkit (Southbound Gateway)

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Overview 9

Centralize Network Management with Southbound Gateway 9
Prerequisites for Developers 11

Chapter 2: Southbound Gateway Architecture 13

Southbound Gateway Model Types 13
Southbound Gateway Model Type Support Files 14
Host Model Types 15
The Flow of Data through the Southbound Gateway 16

Chapter 3: Creating a Southbound Gateway Integration 17

How to Create a Southbound Gateway Integration 17
Configuring the Third-Party System 17
- About Integrating Alert Data from a Third-Party System in a Secure Domain 19
- About Host Agents and Southbound Gateway 19

Map SNMP Trap Data to a CA Spectrum Event 20
- AlertMap File Syntax 21

Map Non-SNMP Alert Data to a CA Spectrum Event 23
- Use XML to Create CA Spectrum Events 23
- Using CORBA to Create Events 27

The Event Data Template 27
- Event Data Template Fields 28
- Variable ID Fields 1-6 31
- Variable ID Field 7 (Target Name) 32
- Variable ID Field 8 (Target Address) 33
- Variable ID Field 10 (EventModel Name) 33
- Variable ID Field 11 (Model Class) 34
- Variable ID Field 13 (Network Address) 34
- Variable ID Field 14 (MAC Address) 34
- Variable ID Field 15 (Manufacturer) 34
- Variable ID Field 16 (Target Name Case Insensitive) 34
- Other Variable IDs (Any Additional Data) 34

Control Events and Alarm Creation 35
- EventAdmin EventDisp File 36
- EventModel EventDisp File 37

Presentation Format of Events 37

Add Value to Alarms 39

Chapter 4: Distributing a Southbound Gateway Integration 41

About Distributing a Southbound Gateway Integration 41
The Installation Script 41
The Part Description File 42
Create an Index File 42
    Sample Index File 43
mkmm and mkcd 44

Chapter 5: Using a Southbound Gateway Integration 45

About Using a Southbound Gateway Integration 45
Create an EventAdmin Model 45
    EventAdmin Model Status 47
    Move an EventAdmin Model 47
    View EventAdmin Model Information 49
Use a Host Model Type Instead of the EventAdmin Model Type 49
Create an EventModel Model 50
    EventModel Models 51
    About Moving an EventModel Model 51
    Put an EventModel Model into Maintenance Mode 52
Set Up a Fault-Tolerant Environment 53
    Archive Manager Failure 54

Chapter 6: Southbound Gateway Case Study 55

Getting Started 55
Step 1: The AlertMap File 55
Step 2: The EventAdmin EventDisp File 57
Step 3: The EventModel EventDisp File 57
Step 4: The Event Format File 57
Step 5: The Probable Cause File 58
Step 6: Package for Distribution 59

Chapter 7: Creating a Southbound Gateway Demonstration 61

Introduction to Creating a Southbound Gateway Integration 61
Step 1: Edit the EventAdmin AlertMap File 61
Step 2: Edit the EventAdmin EventDisp File 62
Step 3: Edit the EventModel EventDisp File 62
Step 4: Create an Event Format File 63

Step 5: Create a Probable Cause File 63
Step 6: Create an EventAdmin Model 64
Step 7: Send the Trap 65
Step 8: Show the Results in OneClick 66
The Alert is Received and Translated into a CA Spectrum Event 66
The Event is Processed by the EventAdmin 66
An Alarm is Asserted on the EventModel 66
View Data About the Alarm in OneClick 67

Index **69**

Chapter 1: Overview

This section contains the following topics:

Centralize Network Management with Southbound Gateway (see page 9)
Prerequisites for Developers (see page 11)

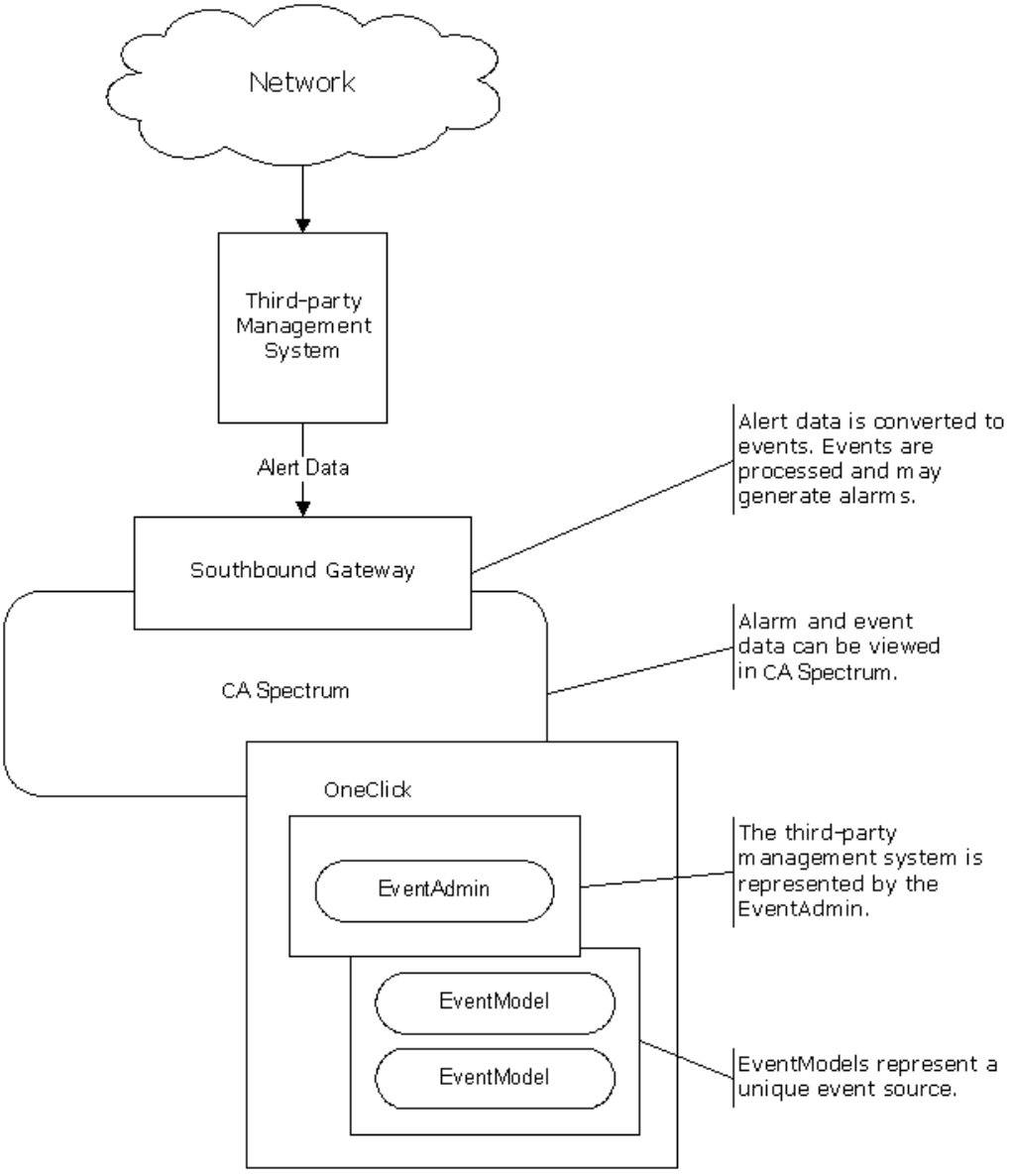
Centralize Network Management with Southbound Gateway

The Southbound Gateway integration point accepts alert data from third-party sources. Often these third-party sources are other network management applications that specialize in monitoring a specific aspect of a computing environment. Using Southbound Gateway, you can centralize network management, allowing CA Spectrum to capture and display data from other systems. Alert data is organized into CA Spectrum event and alarm data as appropriate and is displayed within OneClick.

Southbound Gateway can be used with any incoming alert data stream format. It provides a simple, non-programmatic integration point for systems that can generate SNMP traps. It is also very useful for managing non-SNMP environments. Southbound Gateway supplies an import tool that accepts XML-formatted alert data in case the system you are integrating cannot generate SNMP traps. A Document Type Definition (DTD) is provided to specify the XML elements used in this file.

Model types that represent the external management system and its components are built into CA Spectrum, eliminating the need to generate or derive new model types. The EventAdmin model type represents a third-party system that sends alerts to CA Spectrum. The EventAdmin can contain one or more EventModels. These EventModels represent a unique event source within the network management system. Using the Southbound Gateway toolkit, you create alert, event, and alarm related support information for these model types. When the integration is complete, you can monitor events and alarms from this third-party system using OneClick.

The following diagram illustrates the components of a Southbound Gateway integration and shows how these components work together to process third-party alert data.



Prerequisites for Developers

Before you use Southbound Gateway, verify that you have met the following requirements:

- Significant exposure to CA Spectrum
- Have read the *Concepts Guide* and are familiar with the underlying concepts of CA Spectrum
- Have a basic knowledge of MIBs and SNMP
- Have detailed knowledge of the system you are integrating
- (For non-SNMP integrations) Have a working knowledge of XML, C++, or Java
- Ability to use UNIX or Windows operating system to navigate the file system, copy and delete files, and create and edit text files
- Have a CA Developer ID

Note: The Developer ID is essential for creating some of the support files in a Southbound Gateway integration and is used when compiling the integration for distribution. For more information about Developer IDs and details about how to obtain one, see the *Concepts Guide*.

Chapter 2: Southbound Gateway Architecture

This section contains the following topics:

Southbound Gateway Model Types (see page 13)
Southbound Gateway Model Type Support Files (see page 14)
Host Model Types (see page 15)
The Flow of Data through the Southbound Gateway (see page 16)

Southbound Gateway Model Types

An EventAdmin model type represents a third-party system sending alerts to CA Spectrum. Each instantiated EventAdmin model represents an individual instance of a running external management application. Specific events from the third-party system are sent to the EventAdmin model. The EventAdmin model receives these events and transfers the event data to EventModels or device models depending on how the integration has been configured. Alarms can be created from this event data.

The EventModel is a model type that represents a unique source of event data on the system managed by the EventAdmin's application. A given EventAdmin model can contain one or many instantiated EventModels. Each event received through the Southbound Gateway contains information that uniquely identifies the source of that event. The EventAdmin receives the event, finds the unique event source, and passes the event to the target model that represents the unique source. If the integration has been configured to use a unique identifier to map events exclusively to EventModels and no EventModel exists for the source, an EventModel model is automatically created to represent it. All new EventModel models are placed in the corresponding EventAdmin container model. You can cut or copy EventModel models from an EventAdmin container model and paste them into other types of container models in the Topology, Location, or Organizational view. You can cut or copy the EventAdmin and paste it into the Topology, Location, or Organizational view. You can also put EventModels into maintenance mode as needed.

More information:

About Moving an EventModel Model (see page 51)
Move an EventAdmin Model (see page 47)

Southbound Gateway Model Type Support Files

Support files supply information to the EventAdmin and EventModel models. Work with the following support files to create a Southbound Gateway integration:

- **AlertMap file:** If the alert source sends SNMP traps, create a text file to add data to the EventAdmin's AlertMap file. This lets CA Spectrum receive and parse the SNMP traps and convert them to CA Spectrum events.
- **XML file:** If the alert source does not send SNMP traps, create an XML file to send events to CA Spectrum. The syntax of the XML file must follow the Document Type Definition (DTD) provided with this toolkit. You can find a sample XML file named sbgw_event_sample.xml in the SS-Tools/GSADEMO directory. The Southbound Gateway import tool imports the data from the XML file into CA Spectrum.
- **EventDisp file:** Create these text files to add data to the EventDisp files that the EventAdmin and EventModel use. These additions define how the events are processed.
- **Event Format file:** Create Event Format files to provide additional, optional, or textual information and formatting for each event.
- **Probable Cause file:** Create Probable Cause files to give textual information about each alarm you create.

These support files organize the data into an event, process the event, and display information about the event. If the alert is SNMP-based, it is sent to the AlertMap file. If the data is non-SNMP based, it can be formatted into an XML file and processed by the Southbound Gateway import tool.

In both cases, the data is mapped from the alert to the CA Spectrum event. Variable data from the event is formatted, and a unique event source identifier is defined using the Southbound Gateway Event Data Template. The event is then passed to the EventAdmin model. The EventAdmin model checks its EventDisp file to verify that it is registered to receive this event. If the event is present in the EventDisp file, the EventAdmin examines the event to find the unique ID. The variable information sent with the event is mapped to determine a unique event source. Based on this unique identifier, the EventAdmin forwards the event to the appropriate EventModel. If this model does not yet exist, it is automatically created.

When the EventModel receives the event, it uses its EventDisp file to determine what to do with the event. The event can be logged, mapped to an alarm, used to clear an alarm, or used with other events to create an alarm. The EventModel uses the Event Format and Probable Cause files to display information about the event or alarm in the various Event and Alarm views or applications.

Host Model Types

CA Spectrum has several host model types that have the same functionality as the EventAdmin model type. These include Host_Compaq, Host_Dell, Host_IBMDirector, Host_Sun, and Host_systemEDGE. You can use any one of these host model types instead of the EventAdmin model type to represent the third-party system.

If you use one of these model types, all the integration instructions in this guide are still applicable. However, modify the AlertMap and EventDisp support files for the model type you have chosen. These files are located in the following directory:

<$SPECROOT>/SS/CsVendor/Ctron_Gen_HOST/*<model_type>*

<model_type>

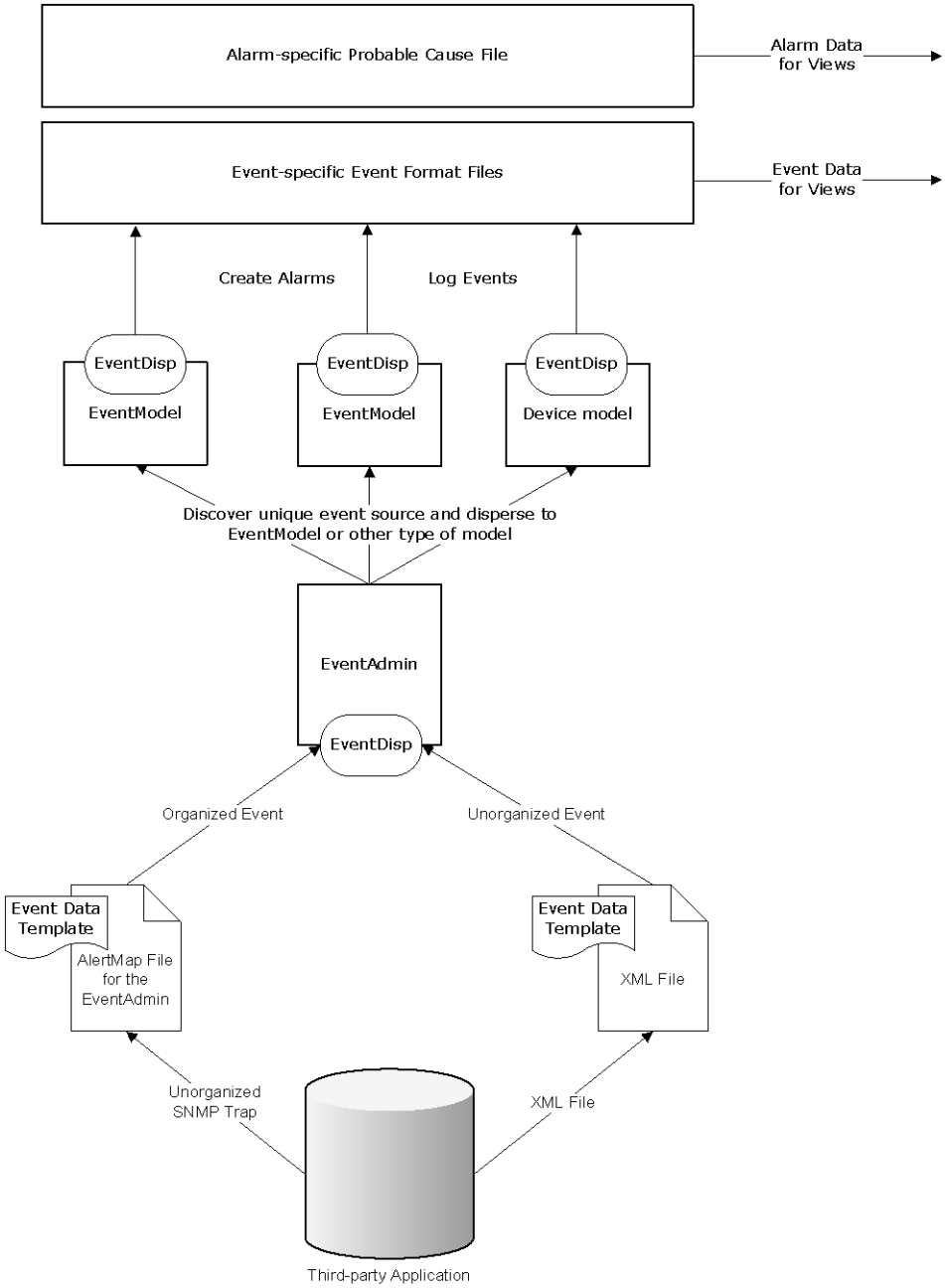
Indicates the host model type you have chosen.

Once you have configured the integration, use the setup instructions provided in the topic, Use a Host Model Type Instead of the EventAdmin Model Type (see page 49).

Note: If you use a host model type instead of the EventAdmin model type, you cannot distribute the integration.

The Flow of Data through the Southbound Gateway

The following diagram illustrates how data flows from the third-party system into CA Spectrum through the Southbound Gateway:



Chapter 3: Creating a Southbound Gateway Integration

This section contains the following topics:

How to Create a Southbound Gateway Integration (see page 17)
Configuring the Third-Party System (see page 17)
Map SNMP Trap Data to a CA Spectrum Event (see page 20)
Map Non-SNMP Alert Data to a CA Spectrum Event (see page 23)
The Event Data Template (see page 27)
Control Events and Alarm Creation (see page 35)
Presentation Format of Events (see page 37)
Add Value to Alarms (see page 39)

How to Create a Southbound Gateway Integration

Your primary job as the Southbound Gateway integrator is to create data for the support files that provide information to the EventAdmin and EventModel models representing the third-party system.

Integration steps include:

1. Configuring the third-party system to send alert data to CA Spectrum.
2. Mapping SNMP trap data to a CA Spectrum event if the third-party system can send SNMP traps.
3. Mapping non-SNMP alert data to a CA Spectrum event if the third-party system sends alerts that are not SNMP-based.
4. Controlling events and alarm creation by creating data for EventDisp files.
5. Defining the presentation format of events with Event Format files. These files provide event information to CA Spectrum applications.
6. Adding value to alarms by creating Probable Cause files. These files provide alarm information to CA Spectrum applications.

Configuring the Third-Party System

The methodology used to set up and configure a third-party system to send alert data to the Southbound Gateway is dependent on the functionality of the individual system.

If the third-party system can send SNMP traps to CA Spectrum, configure it to do so. The application must send the trap data to the host name and port that the SpectroSERVER is running on and listening on. By default, CA Spectrum listens to the standard SNMP trap port 162. You can modify this default by changing the snmp_trap_port parameter in the CA Spectrum .vnmrc file located in the SS directory. Once Southbound Gateway receives the alert data, this data must be mapped to a CA Spectrum event in an AlertMap file.

If the third-party system is sending non-SNMP alerts, you format the alert data using XML and import the data into CA Spectrum using the sbgwimport tool.

In both cases, Southbound Gateway determines the appropriate EventAdmin model to receive the alert data based on the IP address of the host computer sending the data. The IP address of the host computer should match the IP address used to create the EventAdmin model.

More information:

Map SNMP Trap Data to a CA Spectrum Event (see page 20)
Map Non-SNMP Alert Data to a CA Spectrum Event (see page 23)
Create an EventAdmin Model (see page 45)

About Integrating Alert Data from a Third-Party System in a Secure Domain

Consider the following before you integrate alert data from a third-party system in a secure domain:

- Set the SecureDomainAddress attribute (0x12d83) of the EventAdmin model to the IP address of the [assign the value for ssdc in your book] for the domain that includes the third-party system.
- When an EventModel is created, its secure domain context is set to the secure domain context of the EventAdmin that created it.
- An EventAdmin forwards events only to EventModels or device models that have the same secure domain context.
- By default, Southbound Gateway models only forward events to target models when the Southbound Gateway model and the target model have the same secure domain address, or if the Southbound Gateway model and the target model are both directly managed (the secure domain address is 0.0.0.0). Set the SBGW_Ignore_Secure_Domain_Address attribute (0x3dc001d) of the Southbound Gateway model to TRUE to forward events to all matching models, regardless of their secure domain addresses.

  If you want to forward events to remote models, we recommend that a *single* Southbound Gateway model forwards events to each remote target model. If you have multiple Southbound Gateway models forwarding events, all the Southbound Gateway models must have the same secure domain IP address and the SBGW_Ignore_Secure_Domain_Address attribute (0x3dc001d) must be set to the same value. CA Spectrum does not store all the information about a remote model locally, and CA Spectrum does not search for updated information about that remote model when a new Southbound Gateway model forwards events. Remote target models are found using the settings that are set on the Southbound Gateway model. Each Southbound Gateway model finds a remote target model from a cache of models, by name or target IP address, even if the secure domain address does not match the address of the Southbound Gateway model.

About Host Agents and Southbound Gateway

Southbound Gateway can forward syslog messages received as traps from log monitoring agents such as CA Unicenter NSM Agent, CA SystemEDGE Agent, or iAgent. The messages are forwarded to the originating device if it can be determined. The forwarding algorithm inspects the trap message for an IP Address or host name to determine the actual source. If a host name is found, CA Spectrum attempts to resolve the host name to an IP Address and locate the associated model.

For performance reasons, the Southbound Gateway caches resolved and unresolved host names for approximately six hours. Therefore, if you want to use newly created name service entries in the Southbound Gateway where the name was cached as an unsuccessful lookup you must either restart CA Spectrum or wait for six hours.

Southbound Gateway's log matching algorithm is a set of waterfall heuristics that make a best effort attempt to recognize and parse the inbound trap. While CA Spectrum supports over 10,000 known Cisco messages you can set up your own.

Southbound Gateway has debugging capabilities to assist you in resolving ParseMap and translations errors. You can enable this debugging by sending the action 0x3dc0001 to the Agent model. Output appears in the CA Spectrum Control Panel and is written to the VNM.OUT file.

Note: For more information about working with log monitoring agents in CA Spectrum, see the *Host System Resources Management User Guide*.

Map SNMP Trap Data to a CA Spectrum Event

If a third-party system is configured to send SNMP traps to CA Spectrum, map these traps to CA Spectrum events in an AlertMap file. To map traps, create data to be added to an existing AlertMap file. This data specifies the trap, the event it is mapped to, and the variable bindings from the trap. Understand the components of an SNMP trap to map the SNMP trap data in an AlertMap file.

Note: CA Spectrum supports the receipt and processing of SNMPv2 format traps and InformRequests as defined by RFC 2576. For more information about how to refer to SNMPv2 traps in the AlertMap file, see the *Event Configuration User Guide*.

For CA Spectrum to use trap information from a third-party system, associate the trap with a specific CA Spectrum event in an AlertMap file. The AlertMap file has three main functions:

- It indicates the SNMP traps that are received from the third-party system.
- It indicates the event code to be generated for a given trap.
- It maps the trap variable bindings to CA Spectrum event variable IDs.

The AlertMap file that is provided for Southbound Gateway is located in the following directory:

`$SPECROOT/SS/CsVendor/gen_app_gw/EventAdmin`

Do not modify the existing AlertMap file directly. The AlertMap file that is located in the specified directory can contain mapping information for pre-existing integrations. Create a text file that contains the alert map data to be added to this AlertMap file. When you distribute the Southbound Gateway integration, include the Southbound Gateway installation script that is provided with the toolkit. This script recreates the AlertMap file that is based on the text files that are supplied by all Southbound Gateway integrations. For more information, see the *Event Configuration User Guide*.

Use the following convention to rename the test file that you created for AlertMap data:

indexfilename.amp

indexfilename

Specifies the name of the index file that you create when using the CA Spectrum Extension Integration toolkit to distribute this Southbound Gateway integration.

More information:

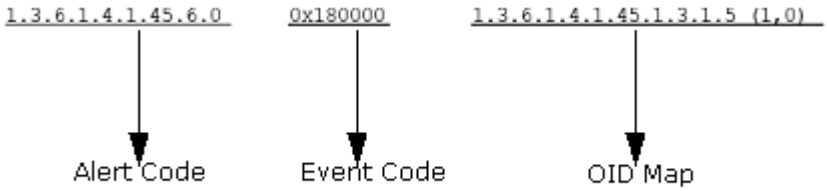
Create an Index File (see page 42)
The Installation Script (see page 41)

AlertMap File Syntax

Each trap mapped to a CA Spectrum event must have an entry in the AlertMap file. Each entry in the AlertMap file has three components:

- The alert code
- The event code
- The OID map

Example AlertMap file entry:



Alert Code

The alert code consists of several pieces of information from the trap: the Enterprise OID string, the generic trap identifier and the specific trap code.

- **1.3.6.1.4.1.45:** Enterprise OID indicating, in this case, a Synoptics device.
- **6:** The Generic Trap Identifier, in this case the 6 indicates an enterprise specific trap.
- **0:** Specific Trap Code indicating the specific enterprise trap.

Event Code

Note: To complete this process, you may need to obtain a Developer ID from CA. For further information about obtaining a Developer ID, see the *Concepts Guide*.

The event code is a hexadecimal number composed of the Developer ID and a number that uniquely identifies this particular event. The Developer ID makes up the first half of the event code. The second half of the event code is a unique number. Generate and manage this unique number to help ensure that an event code for your Developer ID is never repeated.

If the event code is 0, this alert is ignored.

The OID Map

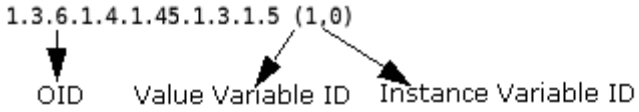
A trap can include one or more pieces of variable information, known as variable bindings. Each variable binding represents a piece of information about the trap; you can map these bindings so the information is used by CA Spectrum.

The variable bindings and the information about how CA Spectrum maps them are specified in the OID map section of the alert map entry. A single alert map entry can have multiple OID maps associated with it, which indicates that there were many variable bindings sent with the SNMP trap. Separate these OID maps using the '\' character and a new line as shown in the following example:

```
1.3.6.1.4.1.52.6.271 0x1060f 1.3.6.1.4.1.52.1.2.1.9.2.1.2(3,4)\
                             1.3.6.1.4.1.52.1.2.1.1.24(5,6)\
                             1.3.6.1.4.1.52.1.2.2.3.1.1(1,2)
```

The OID map can be broken down into three parts:

- The OID
- The value variable ID
- The instance variable ID



The OID identifies the specific variable sent with the trap. For example, the previous OID references the s3ChassisPsStatus variable in the Synoptics Trap MIB.

The value variable ID stores the value of the variable sent in the variable binding. The instance variable ID stores the instance portion of the OID. If your variable binding identifies a particular object from a table variable within the trap MIB, it likely includes an instance ID.

You determine the integer values used for the value variable ID and the instance variable ID by referring to the Southbound Gateway Event Data Template. The values you choose depend on the content provided by the variable binding. The following section outlines the Event Data Template and how to choose the appropriate number, depending on the content of the variable binding.

Map Non-SNMP Alert Data to a CA Spectrum Event

Two different approaches are available for mapping non-SNMP alert data to a CA Spectrum event. The first method is to use XML to create CA Spectrum events. We recommend this method, as it is both efficient and easy to implement.

You can programmatically send events to CA Spectrum. This method requires a more in-depth understanding of the CA Spectrum application programming interfaces.

More information:

The Event Data Template (see page 27)

Use XML to Create CA Spectrum Events

Southbound Gateway provides the sbgwimport tool to create CA Spectrum events based on data from an XML file. You mark-up the data contained in the alert using the XML elements provided in the Southbound Gateway document type definition file, .sbgwimport.dtd. Once you have created the XML file, you run the sbgwimport tool. The tool creates events from the data in the XML file and sends the events to the EventAdmin model that represents the third-party system. Southbound Gateway identifies the EventAdmin model that represents the third-party system based on the IP address of the host running the sbgwimport tool. If an EventAdmin model to represent the third-party system does not yet exist, a new EventAdmin model will automatically be created.

Note: You should have a working knowledge of XML and how a DTD functions before attempting to use this tool.

Create an XML File to Format the Alert Data

Southbound Gateway provides the .sbgwimport.dtd file located in the SS-Tools directory. The DTD defines the elements that can be used to create the XML input file. Two elements are defined in the DTD: the Import element and the Event element.

The Import element is the root element. XML syntax rules specify that the root element is the outermost element and denotes the beginning and ending of the XML file. Therefore, the Import element is the element that surrounds the rest of the XML elements used in the input file. It has one child element, the Event element. Any number of Event elements can exist within the Import element.

Attributes

The Event element is used to format alert data using a series of attributes. These attributes define the event that is sent to CA Spectrum. All these attributes correspond with specific fields in the Event Data Template with the exception of the eventType and the eventAdminName attributes. The attributes that correspond to the Event Data Template takes on the event variable ID specified in the Event Data Template. The event variable ID can then be used in an Event Format file. The following section defines all the available attributes.

eventType

Event Data Template Variable ID: N/A

This is a required attribute. The value for this attribute should be the hexadecimal event code assigned to the event. The event code is a hexadecimal number composed of the Developer ID and a number uniquely identifying this particular event. Your Developer ID makes up the first half of the event code. The second half of the event code is simply a unique number. Generate and manage this unique number to help ensure that an event code for your Developer ID is never repeated.

uniqueId1- uniqueId6

Event Data Template Variable ID: 1-6

targetName

Event Data Template Variable ID: 7

Note: You can use the attribute targetNameIgnoreCase if you do not want CA Spectrum to consider case when matching the attribute's value to the model name.

targetAddress

Event Data Template Variable ID: 8

eventModelName

Event Data Template Variable ID: 10

eventAdminName

Event Data Template Variable ID: N/A

This attribute is only used if an EventAdmin model representing the system sending the events does not yet exist. If this EventAdmin model does not exist, CA Spectrum creates an EventAdmin model and uses the value of this attribute as the model name.

If this value is not specified when a new EventAdmin model is created, *EventAdmin_<HostName>* is used as the model name, where *<HostName>* is the name of the host computer running the sbgwimport tool.

If an EventAdmin model exists, CA Spectrum ignores the value for the eventAdminName attribute. After you run the sbgwimport tool, a message is shown indicating that the value for the eventAdminName attribute has not been used because the EventAdmin model representing this system exists.

networkAddress

Event Data Template Variable ID: 13

macAddress

Event Data Template Variable ID: 14

Manufacturer

Event Data Template Variable ID: 15

This attribute can be set with the manufacturer of the alert source. If defined, the value of this attribute can be referenced in the Event Format file.

targetNameIgnoreCase

Event Data Template Variable ID: 16

userDefined_101- userDefined_120

Event Data Template Variable ID: 101-120

These attributes can be used to define user-specific data. If defined, each of these values can be referenced in the Event Format file. More user-defined attributes can be added to the DTD if they are necessary for the integration. If added they must use an integer value > 120.

More information:

Variable ID Field 10 (EventModel Name) (see page 33)
Variable ID Field 8 (Target Address) (see page 33)
Variable ID Field 14 (MAC Address) (see page 34)
Variable ID Field 13 (Network Address) (see page 34)
Variable ID Field 15 (Manufacturer) (see page 34)
Variable ID Field 16 (Target Name Case Insensitive) (see page 34)
Variable ID Fields 1-6 (see page 31)
Variable ID Field 7 (Target Name) (see page 32)

Sample Alert Data XML File

The following is a sample XML file that creates three events:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE Import SYSTEM ".sbgwimport.dtd">

<Import>
    <Event eventType = "0x3c80000" uniqueId1 = "xyzLocation"
        uniqueId2 = "abcPortNumber" eventAdminName = "MyTestAdmin"
        modelClass = "Router"
        userDefined_101 = "Test Application Name" />
    <Event eventType = "0x3c80004" uniqueId1 = "xyzLocation"
        uniqueId2 = "defPortNumber"
        eventAdminName = "MyTestAdmin" />
    <Event eventType = "0x3c80004" targetAddress = "10.253.29.1"
        eventAdminName = "MyTestAdmin" />
</Import>
```

The first event has an event code of 0x3c80000 defined by the eventType attribute. The attributes uniqueId1 and uniqueId2 are used to identify the source of this event. These attributes are concatenated together to create xyzLocation_abcPortNumber as the unique identifier for the alert source. The EventAdmin model to receive the event is named MyTestAdmin. The device sending the event is identified as a router using the modelClass attribute. The user_Defined_101 attribute is used to send data not described by one of the other fields.

The second event has an event code of 0x3c80004. The attributes ‘uniqueId1’ and ‘uniqueId2’ are used to identify the source of this event. These attributes are concatenated together to create ‘xyzLocation_defPortNumber_ClientID0x56FF’ as the unique identifier for the alert source. The EventAdmin model to receive the event is named ‘MyTestAdmin.’

The third event also has an event code of 0x3c80004. The attribute targetAddress is supplied to direct the event to a pre-existing CA Spectrum device model that has an IP address of 10.253.29.1. The EventAdmin model to receive the event is named ‘MyTestAdmin.’

Run the sbgwimport Tool

Once you have created the XML file, you are ready to run the sbgwimport tool.

Note: You may want to run the sbgwimport tool from the third-party host computer because Southbound Gateway identifies the EventAdmin model that represents the third-party system based on the IP address of the host computer that runs the sbgwimport tool. To run the sbgwimport tool from the third-party host, move the sbgwimport tool and all of its support files to the third-party host. For details about how to move the sbgwimport tool, see the *Distributed SpectroSERVER Administrator Guide*.

To run the sbgwimport tool, use the following syntax:

```
sbgwimport –vnm v_name inputfile
```

v_name

Specifies the name of the CA Spectrum server.

inputfile

Specifies the name of the XML file.

Note: Under some circumstances, such as on a Linux computer, the sbgwimport tool returns the localhost address (127.0.0.1) instead of the actual IP address. If this occurs, you may have to specify the IP address the sbgwimport tool uses to identify the desired EventAdmin model.

To run the sbgwimport tool using a specific IP address, use the following syntax:

```
sbgwimport -vnm v_name input_file -localIP IP address
```

IP address

Specifies the specific IP address.

Using CORBA to Create Events

You can use the CORBA API to create and send events into CA Spectrum. Use a very small portion of the functionality of this API to connect to the appropriate EventAdmin model, format the event data from the third-party system, and then send this data in the form of an event to the EventAdmin model. CA Spectrum provides you with sample code that sends events to an EventAdmin model. The files that make up this example can be found in the SS-Tools/GSADEMO directory. The sample code is in a file named SBGWImport.java.

You can use the command line interface (CLI) to generate these events; however, for performance reasons, we recommend that you use the CORBA API. This toolkit only provides examples using the SSORB interface.

The Event Data Template

The Southbound Gateway Event Data Template is used to define the meaning of alert variable data. This template is used with both SNMP and non-SNMP integrations. For SNMP integrations, the Event Data Template variable IDs are used in the AlertMap file to select the appropriate number for the value variable and instance variable IDs.

Correctly choosing the Event Data Template variable ID values is important because they enable essential functions of the Southbound Gateway integration. The values you choose determine how a target model for the incoming event is identified. You can control whether a new EventModel is created to handle an incoming event. You also have the option of specifying variable ID values which specify whether to use a network address or model name as the event target search criteria. This allows for mapping events to any type of model, including devices or ports. Other values allow the variable data to be used as attribute values for the EventModel the event is sent to. All the variables can be referenced in the Event Format files. A zero value for either the value variable or instance variable IDs indicates that the value is not stored.

The Event Data Template shows the name, type, and description of each variable ID that has meaning to the Southbound Gateway mechanism. When selecting values for the value variable ID for an integration, choose appropriate variable IDs that describe the content of the value of the variable binding. If it is applicable to store the instance variable ID, it is stored as one of the unique identifier fields.

Specify one of the following variable ID values to have a target model identified:

- One of the Unique ID variables (1-6)
- Target Name (7) or Target Name Case Insensitive (16)
- Target Address (8)

More information:

Map Non-SNMP Alert Data to a CA Spectrum Event (see page 23)

Event Data Template Fields

The following table describes the fields in the Event Data Template. Each event must contain the required vardata items to be processed by the Southbound Gateway.

Note: If the event received does not conform to the Event Data Template, an event with an ID 0x03dc0000 is generated against the EventAdmin model.

Variable ID	Name	Type	Description	Required/Optional
1	Unique ID 1	String/Integer	The Unique ID variables are used to identify a target EventModel by a Unique Identifier string. The Unique Identifier is a composite of up to six variable data items (one to six). The final unique identifier string is composed as follows:	
2	Unique ID 2	String/Integer		
3	Unique ID 3	String/Integer		
4	Unique ID 4	String/Integer		

Variable ID	Name	Type	Description	Required/Optional
5	Unique ID 5	String/Integer	<1>_<2>_<3>_<4>_ <5>_<6> in that exact order. If one of the unique identifier components is not provided, it is not included within the composite unique identifier. If an existing EventModel with the Unique Identifier string cannot be found one is created. Either one unique identifier or a Target Name (field 7) or Target Address (field 8) is required.	
6	Unique ID 6	String/Integer		
7	Target Name	String	This field lets you specify a target model by model name. **Note:** An EventModel will not be created when a target model cannot be found when using the Target Name event variable. This field is case sensitive. If you do not want Southbound Gateway to consider case when identifying the model by model name, use the Target Name Case Insensitive field defined by Variable ID Field 16.	
8	Target Address	Octet/Text String	This field lets you specify a target model by IP address. Use this field if you want to send the event to a model other than an EventModel.	
9	Reserved for future CA use			

Variable ID	Name	Type	Description	Required/Optional
10	EventModel Name	String/Integer	This gives the user the ability to give a model name that is different than the unique identifier. If this data is not provided, the composite unique identifier becomes the model name.	Optional
11	Model Class	Integer	Populates the Model Class attribute of the target EventModel with the value specified in this event variable.	Optional
12	Reserved for future CA use			
13	Network Address	Octet/Text String	Populates the Network Address attribute of the target EventModel with the value specified in this event variable.	Optional
14	MAC Address	Octet/Text String	Populates the MAC Address attribute of the target EventModel with the value specified in this event variable.	Optional
15	Manufacturer	String	Populates the Manufacturer attribute of the target EventModel with the value specified in this event variable.	Optional

Variable ID	Name	Type	Description	Required/Optional
16	Target Name Case Insensitive	String	This field lets you specify a target model by model name. **Note:** An EventModel will not be created when a target model cannot be found when using the Target Name Case Insensitive event variable. This field is case insensitive. If you want Southbound Gateway to consider case when identifying the model by model name, use the Target Name field defined by Variable ID Field 7 (Target Name) (see page 32), which is case sensitive.	
17-99	Reserved for future CA use			
Any other variable ID greater than or equal to 100.	Any Data	Any Type	This data is forwarded to the EventModel model unchanged. The data type and data are preserved. This data can be viewed within an event message.	Optional

Variable ID Fields 1-6

The EventModel unique identifier is composed of the first six fields within the Event Data Template. Southbound Gateway integrations often require more than one piece of data to identify the source of an alert. The Event Data Template provides a way to create an EventModel unique identifier that is composed of the elements from up to six different fields. The EventModel unique identifier is composed as follows:

<1>_<2>_<3>_<4>_<5>_<6>

For example, a location and port number can be used to represent a router as the EventModel unique identifier. The router's location would be put in Event Data Template field 1 and the port number would be put in Event Data Template field 2. The EventModel unique identifier would then appear as follows: Location_PortNumber.

Variable ID Field 7 (Target Name)

This field lets you specify the model name of the model you would like the event data to be sent to. For example, if you specify the model name *MyRouter* in the Target Name field, Southbound Gateway searches the existing models to find a model with this model name. If this model is found, Southbound Gateway sends the event data to this model for processing.

Important! If no model is found, an EventModel is not created.

This field has the same functionality as Variable ID Field 16 (Target Name Case Insensitive) (see page 34), except that it is case sensitive. If this field had the value MyRouter, and CA Spectrum had two models, one named MyRouter and one named myrouter, MyRouter would receive the event and myrouter would not. If instead you used the Target Name defined in field 16, both models (MyRouter and myrouter) would receive the event. If for some reason you used both of these fields (7 and 16) and each had the value MyRouter, the model with the model name MyRouter would receive two events (sent as a result of field 7 and 16) and the model with the model name myrouter would receive one event (sent as a result of field 16).

If you specify both a unique ID (using a combination of fields 1 through 6) and a Target Name (using fields 7 or 16), Southbound Gateway sends the event to both the EventModel specified by the unique ID and the model specified by the Target Name.

If you specify a unique ID, a Target Name using fields 7 or 16, and a Target Address, and the Target Name and Target Address are for different models, the Southbound Gateway sends the event to all three models.

If the Target Name is not found, the EventAdmin model generates an Event that indicates that CA Spectrum has no models with the name <Target Name>. If a unique ID has also been specified, the event is sent to an EventModel.

If you specify a Target Name using fields 7 or 16 and a Target Address and one or both is not found, the appropriate error message is generated, and the event is not sent to any model. If a unique ID has also been specified, the event is sent to an EventModel.

More information:

Event Data Template Fields (see page 28)

Variable ID Field 8 (Target Address)

This field lets you specify the IP address of the model you would like the event data to be sent to. Use this field if you want to send data to a model other than the EventModel. For example, if you specify the IP address 10.253.97.2 in the Target Address field, Southbound Gateway searches the existing models to find a model with this IP address. If Southbound Gateway finds this model, it sends the event data to this model for processing.

If you specify both a unique ID, using a combination of fields 1 through 6, and a Target Address, Southbound Gateway sends the event to both the EventModel specified by the unique ID and the model specified by the Target Address.

If you specify a unique ID, a Target Name, and a Target Address, and the Target Name and Target Address are for different models, the Southbound Gateway sends the event to all three models.

If the Target Address is not found, the EventAdmin model generates an Event that indicates that CA Spectrum has no models with the network address <Target Address>. If a unique ID has also been specified, the event is sent to an EventModel.

If you specify a Target Name and a Target Address and one or both is not found, the appropriate error message is generated, and the event is not sent to any model. If a unique ID has also been specified, the event is sent to an EventModel.

Variable ID Field 10 (EventModel Name)

This is an optional name that can be given to the EventModel model. If supplied, it appears beneath the EventModel icon in the OneClick topology views. If you define an EventModel prefix when creating the EventAdmin model, the EventModel name is concatenated onto the EventModel prefix, separated by an underscore. If you do not define the EventModel prefix and the EventModel name is defined, the EventModel name becomes the model name. If the EventModel Name is not provided within Field 10, the unique identifier is used as the EventModel model name.

More information:

Create an EventAdmin Model (see page 45)

Variable ID Field 11 (Model Class)

If provided, this variable indicates the value with which to populate the model class attribute on the EventModel. The model class defines the type of device the model represents. The entry in Field 11 should be an integer that identifies the model class. A list of model classes and their respective integer identifiers can be found in the *Concepts Guide*.

Variable ID Field 13 (Network Address)

If provided, this variable indicates the value with which to populate the network address attribute on the EventModel.

Variable ID Field 14 (MAC Address)

If provided, this variable indicates the value with which to populate the MAC address attribute on the EventModel.

Variable ID Field 15 (Manufacturer)

If provided, this variable indicates the value with which to populate the manufacturer attribute on the EventModel.

Variable ID Field 16 (Target Name Case Insensitive)

This field lets you specify the model name of the model you would like the event data to be sent to. This field has the same functionality as Variable ID Field 7 (Target Name) (see page 32), except this field is case insensitive.

Other Variable IDs (Any Additional Data)

Any other integration-specific data can be placed in any CA Spectrum variable ID greater than or equal to 100. This data is not set as attributes on the EventModel model, but is viewable from within the Event Format file.

The following list shows additional data that can be captured for display in an event message:

- application name: 100
- server name: 101
- event status: 102

- ticket status: 103
- ticket type: 104
- comments: 105

Control Events and Alarm Creation

EventDisp files define how events are processed. When alert information has been converted to an event, CA Spectrum identifies the model that the event has occurred in and then locates the EventDisp file that is associated with that model's model type. CA Spectrum searches the file for the appropriate event code and carries out the action specified for that event code.

Because there are two different model types that play a part in handling events for Southbound Gateway, create data for two different EventDisp files. There is one EventDisp file for the EventAdmin model type that defines how the EventAdmin models process the event and there is another EventDisp file for the EventModel model type that defines how the EventModel models process the event.

The EventAdmin EventDisp file is located in the following CA Spectrum directory:

<$SPECROOT>/SS/CSVendor/gen_app_gw/EventAdmin

The EventModel EventDisp file is located in the following CA Spectrum directory:

*<$SPECROOT>*SS/CSVendor/gen_app_gw

Do not modify the existing EventDisp files directly. The EventDisp files currently located in these directories can contain information for pre-existing integrations. Instead, create text files containing the EventDisp data that should be added to each of the files. When you distribute the Southbound Gateway integration, you include the Southbound Gateway installation script provided with the toolkit, which recreates each of the EventDisp files based on the text files supplied by all installed Southbound Gateway integrations.

The text files that you create for your EventDisp data must each use the following naming convention. This gives you two files with the same name. We recommended that you manage these files by keeping them in separate directories while you are working with them.

<indexfilename>.evd

<indexfilename>

Specifies the name of the index file that you create when using the CA Spectrum Extension Integration toolkit to distribute this Southbound Gateway integration.

More information:

Create an Index File (see page 42)
The Installation Script (see page 41)

EventAdmin EventDisp File

The purpose of the EventAdmin EventDisp file is to define what events are received from the external alert/trap source. To do this, the EventDisp file must list the event codes of the events that are received. These event codes are defined in the AlertMap file, or in the XML import file.

Note: This EventDisp file must list those events that are forwarded to other CA Spectrum models through the use of Target Name or Target Address, and events that are forwarded to EventModel models.

For example, an EventDisp file expecting to receive three events with the codes 0x03dc003, 0x03dc004, and 0x03dc005 would have the following content:

```
0x03dc003
0x03dc004
0x03dc005
```

You can define more options for the event within this EventDisp file but it is not recommended. The intention of this file is to indicate what events the EventAdmin model should process. Once the EventAdmin receives the event, it forwards the event to the proper EventModel. It is at the EventModel level where further definition of the event processing takes place.

Note: This file is not optional. If your integration sends an event code that is not referenced in the EventAdmin EventDisp file, CA Spectrum simply drops that event, which prevents it from being forwarded to the appropriate EventModel model.

More information:

Step 2: Edit the EventAdmin EventDisp File (see page 62)

EventModel EventDisp File

Although the EventDisp for the EventModel is not required, it is highly recommended. In the EventAdmin EventDisp file, all the events that the EventAdmin could receive are listed. At the EventModel level you define how these events are processed. In addition to receiving events, EventDisp files can define a number of event behaviors and characteristics including:

- If the event should be logged
- The severity of the event
- If the event should clear an alarm
- If the event should generate an alarm
- If the alarm generated can be cleared by a user
- The severity of any alarm to be generated
- The probable cause of any alarm to be generated
- The event or series of events that should generate another event

Using the EventCondition Event Rule, you can create an event disposition entry that allows events to be created conditionally based on the value of variable bindings from a trap or based on the value of CA Spectrum attributes. This can be useful if the third-party application sending traps to CA Spectrum through the Southbound Gateway sends only one trap OID. This trap may have different meanings based on the value of one or more variable bindings. Using the EventCondition rule, you can generate different events depending on the value of these variable bindings. See the *Event Configuration User Guide* for complete information about the EventConditon Event Rule.

In general, if you are forwarding events to a model other than an EventModel model, you define the instructions for event processing in the EventModel's EventDisp file. This EventDisp file processes events for all appropriate model types. However, if you are using the EventCondition Event Rule as outlined in the previous paragraph, you can create this entry in the model type EventDisp file for the model to which you are forwarding events. This forces the condition parsing work to be done by that model type rather than by the EventModel.

The *Event Configuration User Guide* provides reference information about EventDisp files. See the syntax instructions in this guide when creating your EventDisp syntax.

Presentation Format of Events

Event format files determine the contents of an event message when the event is displayed in OneClick. Each event has a separate event format file. Event messages appear in the Events tab and the Alarm Details tab in OneClick.

Each event that requires a corresponding event message must have an associated event format file. Keep in mind that most of the information a user receives about an event or alarm is through the message text that is affiliated with that alarm or event. The index file that you create for distribution unpacks the event format files into the correct location (SG-Support/CsEvFormat) on the user's SpectroSERVER.

Unlike the AlertMap and EventDisp files, the data in the text files is not added to other files. Therefore, it is important to use the naming convention outlined in this section to identify your Event Format files.

The event format file must be named as Eventxxxxxxxx where xxxxxxxx equals the event code given to the event in the AlertMap file and listed in the EventDisp file. The event code must be written in hexadecimal format with a full eight digits, including leading zeros.

The event format file can contain plain text, but can also contain variables that reference specifics about the instance of the individual event. For information about possible variable references, see the *Event Configuration User Guide*.

The following is an example of the event format file:

```
{d "%w- %d %m-, %Y - %T"} - ASX DS3 LOS DETECTED - This trap indicates that the specified
DS3 port has detected incoming LOS (Loss of Signal) Alarm. - Model Name: {m};
PortName: {S 1},
PortBoard: {I 2},
PortModule: {I 3},
PortNumber: {I 4}.
(event [{e}])
```

In this example, {d "%w- %d %m-, %Y - %T"} places the date and time into the event message, m inserts the model name, and e inserts the event code. The other variables reference values from variable bindings and their corresponding data types, S for string and I for integer.

Use the variable IDs that you have used from the Event Data Template to construct any type of message; however, verify that the variables in the message are the correct data type.

Add Value to Alarms

Alarms that are generated as a result of the EventDisp file usually contain a probable cause as part of the alarm message. The text of this probable cause is defined in the Probable Cause file associated with the alarm. Create a separate Probable Cause file for each of the alarms that you generate. Each Probable Cause file contains a textual explanation of the cause of the alarm with optional advice about the likely cause and common troubleshooting steps. The index file that you create for installation unpacks the Probable Cause files into the correct location (SG-Support/CsPCause) on the user's SpectroSERVER. Unlike the AlertMap and EventDisp files, the data in the text files is not added to other files. Therefore, it is important to use the following naming convention to identify your Probable Cause files.

Each Probable Cause file must be named Probxxxxxxxx where xxxxxxxx is the event code that generated the alarm through the EventDisp file. Write the event code in hexadecimal format with a full eight digits, including leading zeros.

The Probable Cause file contains text only. The first line states the cause of the alarm in uppercase letters. Following the cause are three separate sections, each with a title in uppercase letters: SYMPTOMS, PROBABLE CAUSES, and RECOMMENDED ACTIONS. Under each of these titles, list the appropriate data.

If there are multiple symptoms, probable causes, or recommended actions, format the data under those categories as a numbered list.

The following is an example of a Probable Cause file showing the appropriate syntax:

```
ALL DEVICE CONNECTIONS ARE UNREACHABLE
SYMPTOMS:
The VNM is unable to contact this device because all of the devices that this one is
connected to are unreachable.
PROBABLE CAUSES:
1) A router connected to this device has gone down.
2) A network cable has become unplugged.
RECOMMENDED ACTIONS:
1) Check the status of the devices that this one is connected to for problems.
2) Check for a loose or unplugged network cable.
```

Chapter 4: Distributing a Southbound Gateway Integration

This section contains the following topics:

[About Distributing a Southbound Gateway Integration](#) (see page 41)
[The Installation Script](#) (see page 41)
[The Part Description File](#) (see page 42)
[Create an Index File](#) (see page 42)
[mkmm and mkcd](#) (see page 44)

About Distributing a Southbound Gateway Integration

After creating the proper files to support your Southbound Gateway integration, the next step is to package the files for distribution to other CA Spectrum users.

To do this, you use some of the tools in the CA Spectrum Extension Integration toolkit. This toolkit provides the ability to create a package of files called a VCD (Virtual CD-ROM) that the CA Spectrum installer opens and expands. This package of files can include a number of different components; at a minimum yours includes the Southbound Gateway installation script, an index file, a Part Description file, and the support files you have created.

The Installation Script

The Southbound Gateway installation script is a pre-built script included with the Southbound Gateway toolkit. It is named MergeVendorFiles.cus and is located in the SS-Tools/GSADEMO directory. This script is responsible for taking your EventDisp and AlertMap data contained in the text files you created in the integration process, and recreating the existing AlertMap and EventDisp files to include that data. Add the following line to all Southbound Gateway index files to verify that the installation script is included with your integration.

```
file: SS cus /<$SPECROOT>/SS-Tools/GSADEMO/MergeVendorFiles.cus ? ?
```

More information:

[Map SNMP Trap Data to a CA Spectrum Event](#) (see page 20)

The Part Description File

The part description file is a text file that contains a brief synopsis of the integration being installed. The user can view the text in this file during the install process. Pay close attention to the directory structure outlined in the Sample Index File (see page 43). For the integration to work properly, the support files must be correctly placed on the computer.

Create an Index File

The index file maps all the files included in the VCD, indicates the files needed for installation, where these files are located on your computer, and where the files should be unpacked on the customer's computer. For a description of the possible syntax for an index file, see the *CA Spectrum Extension Integration (SEI) Developer Guide*. The syntax outlined in Sample Index File (see page 43) describes a typical Southbound Gateway integration installation. In your index file, include references to the following files:

- Southbound Gateway Installation Script
- AlertMap file (if applicable)
- Part Description file
- EventDisp files
- Event Format files
- Probable Cause files

Create your index file using your preferred text editor and then save the index file in the SG-Tools directory. The naming convention for the index file is as follows:

3P-*<ABCxxxx>*.i

3P

Specifies a third-party integration.

<ABC>

Specifies the first three letters of your developer name.

<xxxx>

Specifies any four-digit integer that uniquely identifies this index file from any other index file you have created.

More information:

Map SNMP Trap Data to a CA Spectrum Event (see page 20)

Sample Index File

The following sample index file shows the components of an integration that includes the Southbound Gateway installation script, a Part Description file, an Event Format file, a Probable Cause file, two EventDisp files, and an AlertMap file. At install time, all files are unpacked into their proper directory and the installer program runs the Southbound Gateway installation script.

```
level: 1
mm:     CA Spectrum Southbound Gateway Integration Demonstration
irev:   06.50.00.000
rev: 6.5
pprep:  Sample
vend:   gen_app_gw
file: SS cus /SpectrumRootDirectoryName/SS-Tools/GSADEMO/MergeVendorFiles.cus ? ?
file:   Install mmdesc ? Demo.mmd ? ?
file:   SG other ? ../../SG-Support/CsEvFormat/Event03e50000 ? ?
file:   SG other ? ../../SG-Support/CsPCause/Prob03e50000 ? ?
file:            SS vfile ? ../SS/CsVendor/gen_app_gw/components/3P-ABC0001.evd
SS/CsVendor/gen_app_gw/components 3P-ABC0001.evd
file:            SS vfile ? ../SS/CsVendor/gen_app_gw/EventAdmin/components/3P-
ABC0001.evd SS/CsVendor/gen_app_gw/EventAdmin/components 3P-ABC0001.evd
file:   SS vfile ?
../../SS/CsVendor/gen_app_gw/EventAdmin/AM_components/3P-ABC0001.amp ?
3P-ABC0001.amp
```

The index file can be divided into the following two portions:

- The first portion is the extension module description entries. These entries include specific information about the integration, such as name and release level. For more information, see the *CA Spectrum Extension Integration (SEI) Developer Guide*.
- The second portion of the index file contains the file distribution entries. These entries specify the files that can be included with the installation. The first line in this section has the following syntax:

  ```
  file: SS cus /SpectrumRootDirectoryName/SS-Tools/GSADEMO/MergeVendorFiles.cus ?
  ?
  ```

Include this line in all the index files that are created to distribute a Southbound Gateway integration. This line is responsible for activating the Southbound Gateway installation script. The rest of the index file references the Event Format file, the Probable Cause file, the two EventDisp files, and the AlertMap file for this particular installation. For more information, see the *CA Spectrum Extension Integration (SEI) Developer Guide.*

More information:

Create an Index File (see page 42)
The Part Description File (see page 42)

mkmm and mkcd

To complete the package, run the mkmm and the mkcd tool. See the *CA Spectrum Extension Integration (SEI) Developer Guide* for more information.

Chapter 5: Using a Southbound Gateway Integration

This section contains the following topics:

About Using a Southbound Gateway Integration (see page 45)
Create an EventAdmin Model (see page 45)
Use a Host Model Type Instead of the EventAdmin Model Type (see page 49)
Create an EventModel Model (see page 50)
Set Up a Fault-Tolerant Environment (see page 53)

About Using a Southbound Gateway Integration

CA Spectrum provides different model types that represent components of a third-party system, the EventAdmin, and the EventModel. These model types work together to manage alert information sent to CA Spectrum. A unique EventAdmin exists for each instance of a third-party system managed by CA Spectrum. The EventAdmin is a container model type and includes models of the type EventModel. EventModels represent unique sources of event information that a third-party system manages. The EventAdmin receives an event from the third-party system and dispatches it to an EventModel.

When you represent this system within CA Spectrum, you are representing the management application, not the host computer that the application is running on. To manage this host computer, create a separate model of the appropriate model type.

For both SNMP and non-SNMP integrations, the EventAdmin's network address is stored in the attribute named trapIPAddress, instead of in the usual NetworkAddress attribute. This prevents the EventAdmin from interfering with another CA Spectrum model that is managing the device containing that network address, that is, the model that is managing the host computer.

Create an EventAdmin Model

Before attempting to represent your third-party system within CA Spectrum, verify that the management application is running. Once the application is running properly, the first step is to model the application in CA Spectrum using the EventAdmin model type. In most situations, you add the EventAdmin icon into the same topology view as the VNM icon.

You can define several attributes when you create this model. Most of these attributes carry over to become attributes of EventModel models created for this EventAdmin. There are several applications within CA Spectrum that display the EventModel models' attributes, events, and alarms. To make this information as useful as possible, it is a good idea to provide as much information as you can about those EventAdmin and EventModel models.

To create the EventAdmin model

1. From the Topology tab that you want to create the model in, click the Create new model by type button.

   The Select Model Type dialog appears.

2. In the Containers tab, click EventAdmin.

3. Click OK.

   The Create Model of Type EventAdmin dialog appears.

4. Configure the following parameters:

   Name

   Optional. Defines the EventAdmin's model name. This model name appears in the field at the top of the EventAdmin icon.

   Network Address

   Required for all integrations that are based on SNMP traps. Specifies the network address of the event source's host computer. If you are unsure if your third-party application sends SNMP traps, consult with the documentation for that product.

   Security String

   (Optional) Defines who can view and edit this model.

   Manager Name

   When this attribute is set on the EventAdmin, all EventModels contained within this EventAdmin also have this attribute. If the name of the third-party application does not appear on this list, select Default from the drop-down list.

   EventModel Prefix

   This field is prepended to the EventModel Name for all the EventModels contained by this EventAdmin. This helps ensure that there are consistent naming prefixes for all EventModels associated with a particular EventAdmin. It is also useful for sorting and filtering.

5. After you enter the appropriate information into these fields, click OK.

   This generates an EventAdmin model. A default EventModel is also created and is contained in the EventAdmin model. This model is used for fault tolerance functionality.

Note: You can configure the EventAdmin model to forward events to models on remote landscapes by setting the SBG_AlertForwardingEnabled (0x3dc000c) attribute for the EventAdmin model to TRUE.

More information:

Set Up a Fault-Tolerant Environment (see page 53)
Step 6: Create an EventAdmin Model (see page 64)

EventAdmin Model Status

Once you have instantiated an EventAdmin model, its status becomes green (Normal) even if there are not yet any EventModel models associated with it. This status changes to reflect alarms asserted on the EventAdmin model.

Note: CA Spectrum does not monitor the status of the application that the EventAdmin represents, so the status of the model does not represent the state of the third-party application, nor does it represent the state of the third-party application's host computer.

Move an EventAdmin Model

An EventAdmin model can be cut or copied and then pasted into other topologies as needed.

The following is a list of containers that an EventAdmin can be placed in:

- Universe:
  - Broadband_Net
  - LAN
  - Network
  - Universe
  - WAN
  - GnChasCont
  - LAN_802_3

  - LAN_802_5
  - FDDI
- World:
  - Panel
  - Rack
  - Room
  - Site
- TopOrg:
  - Org_Owns
  - Service_Owns

You move an EventAdmin model in the same way you would move any other model. However, it is not possible for an EventAdmin model to contain another container model. The EventAdmin model can only contain EventModel models.

To move an EventAdmin model

1. Open the topology view that contains the EventAdmin model that you want to move.
2. Right-click the model that you would like to move and select Copy or Cut.
3. Go to topology view where you want to place the EventAdmin model and then right-click and select Paste.

   You have now successfully moved the model.

Note: The EventAdmin model represents the third-party management system and is required for proper operation of the Southbound Gateway integration. If you delete the EventAdmin model to remove support for the third-party system, delete all EventModel models associated with the EventAdmin.

View EventAdmin Model Information

You can review details about the EventAdmin model from the tabs in the Contents panel and the Component Detail panel.

You can use the Information tab to review and edit the model name, network address, manager name, and the EventModel prefix of the EventAdmin model.

From the Topology tab, you can navigate to all the EventModels contained by the EventAdmin.

The EventAdmin model manages the third-party application and the alerts it sends to CA Spectrum, the EventAdmin does not manage the host computer that is running the third-party application.

Use a Host Model Type Instead of the EventAdmin Model Type

You can use a CA Spectrum host model type instead of the EventAdmin model type to represent the third-party system. After making the necessary support file modifications outlined in the previous chapters, you can configure the host model.

To configure the host model

1. Click the create new model button in the view you want to add the host model to.

   The Select Model Type dialog appears.

2. Choose the appropriate host model type:

   - Host_Dell
   - Host_Compaq
   - Host_IBMDirector
   - Host_Sun
   - Host_systemEDGE

   and click OK.

   The Create Model of Type dialog appears.

3. Fill in the fields appropriately, using the IP address of the third-party server for the Network Address field and then click OK.

   CA Spectrum creates the model.

4. For EventModels to be created, set the Enable_SouthboundGateway (0x116296e) attribute on the host model to TRUE. Do this using the Attribute Editor.

   Note: For more information about using the Attribute Editor, see the *Modeling and Managing Your IT Infrastructure Administrator Guide*.

5. Specify where the EventModels created by the host model are to be placed when they are created. They can be placed in any container model available in the Topology view with the exception of any SW_Link, WA_Link, and VNM container models. To specify the appropriate container model, set the host model's EventModelContainerHandle attribute (0x3dc000a) equal to the model handle of the container where you would like the EventModels to be created. Do this using the OneClick Attribute Editor.

   If you would like the EventModels to be placed directly in the Universe view, use the model handle of the Universe view. You can retrieve the model handle of the Universe view by using the Attributes tab in the OneClick Component Detail panel tab to read the model_handle attribute on the Universe model.

   The host model is now configured to receive and process traps.

More information:

Host Model Types (see page 15)
About Host Agents and Southbound Gateway (see page 19)

Create an EventModel Model

The EventModels that are managed by the EventAdmin are created automatically. When an alert is sent from the third-party application into CA Spectrum, the alert is converted into an event. Each event has a unique ID string. This unique ID string is set up during the integration development process to determine a unique source of alert information within the third-party system. The EventAdmin examines this unique ID and looks for an EventModel with a matching unique ID. If this EventModel exists, the event is sent to that EventModel. If an EventModel with this unique ID string does not exist, the EventModel is automatically created.

Note: A default EventModel model is created when the EventAdmin model is created. This is for use in a fault-tolerant environment. EventModel models can also be created using the Modeling Gateway. For more information, see the *Modeling Gateway Toolkit Guide.*

More information:

Set Up a Fault-Tolerant Environment (see page 53)

EventModel Models

EventModels for a particular EventAdmin are automatically placed in a topology view that can be accessed by drilling down from the EventAdmin. These icons do not show any connectivity with one another because they represent an event source, not necessarily a physical device or component.

About Moving an EventModel Model

EventModel models can be cut or copied from the EventAdmin container model and pasted into various types of container models within the Topology, Location, or Organizational views.

The following is a list of containers that an EventModel can be placed in:

- Topology view:
  - Broadband_Net
  - LAN
  - Network
  - Universe
  - WAN
  - GnChasCont
  - LAN_802_3
  - LAN_802_5
  - FDDI
- Location view:
  - Panel
  - Rack
  - Room
  - Site
- Organization view:
  - Org_Owns
  - Service_Owns

You move an EventModel model in the same way you would move any other model.

An EventAdmin model cannot contain another container model. The EventAdmin model can only contain EventModel models.

Even if you have moved all the EventModel models out of the EventAdmin model, it is important that you do not delete the EventAdmin model. This model represents the third-party management system and is required for proper operation of the Southbound Gateway integration. Any new EventModel models that are created appear in this EventAdmin model by default.

If you delete the EventAdmin model to remove support for the third-party system, delete all EventModel models associated with the EventAdmin also.

More information:

Move an EventAdmin Model (see page 47)

Attributes

The EventModel icon displays the model name and the model type. The incoming alert information can assign the model name. If the model name is not assigned, the unique ID string that identifies the EventModel is visible within the Model Name field.

If you defined an EventModel prefix when creating the EventAdmin, this prefix prepends the EventModel name for all the EventModels contained by this EventAdmin. This provides a way to further uniquely identify the EventModel for a particular integration.

If you have chosen a specific Manager Name when creating the EventAdmin, the information is also available to the EventModel.

All these parameters can be changed from the Attributes tab in the EventAdmin's Component Detail view. If such changes are made, existing EventModels are updated with the new attribute values.

More information:

Create an EventAdmin Model (see page 45)

Put an EventModel Model into Maintenance Mode

You can put EventModel models into maintenance mode from the Information tab in the Component Detail view. Putting EventModel models into maintenance mode suspends management traffic to the EventModel model and its components and prevents the generation of any events or alarms on its behalf. When an EventModel is put into maintenance mode, its icon displays a device condition color of brown.

To put an EventModel model into maintenance mode

1. Select the EventModel model you want to put into maintenance mode, click the Information tab in the Component Detail view, and then expand the General Information node.
2. Click set in the In Maintenance field, and then select Yes from the drop-down list.

   The EventModel model is now in maintenance mode; its icon color changes to brown.

Set Up a Fault-Tolerant Environment

Fault tolerance is provided by having more than one SpectroSERVER available for managing a given landscape. If the primary SpectroSERVER fails, the secondary SpectroSERVER takes over.

To set up a fault-tolerant environment with Southbound Gateway

1. Set up the fault-tolerant environment as described in the *Distributed SpectroSERVER Administrator Guide*.
2. Start the primary SpectroSERVER and then start the management application being integrated through the Southbound Gateway.

   The primary SpectroSERVER creates an EventAdmin model automatically if you have used the sbgwimport tool to create your integration, or if you have used the CA Spectrum CORBA API and specified programmatically that the EventAdmin be created automatically. Otherwise, the EventAdmin model is not created automatically and creates it on the primary SpectroSERVER.

   A default EventModel model is created automatically once the EventAdmin model is activated.
3. Synchronize the primary and secondary SpectroSERVER databases using the CA Spectrum Online Backup feature.

   Note: See the *Distributed SpectroSERVER Administrator Guide* and the *Database Management Guide* for instructions.

If the primary SpectroSERVER goes down and the secondary SpectroSERVER takes over, the default EventModel model is used to process incoming events.

When the secondary SpectroSERVER is monitoring the network and events are received by the EventAdmin model, new EventModel models are not created. First the EventAdmin checks to see if there is an appropriate existing EventModel to send the events to, if not, instead of creating an EventModel, the events are sent to the default EventModel model.

Allowing the secondary SpectroSERVER to create new EventModel models would cause disparate model handle information between the primary and secondary SpectroSERVERs. This would cause the Archive Manager to store the event and statistical information for a single event source in different places, as if the information were actually from different event sources. The default EventModel prevents this from happening.

Note: If either the EventAdmin or default EventModel is destroyed on the primary, secondary, or tertiary server, the previous steps must be repeated. Steps 1 to 3 must be repeated if the models are destroyed on the primary SpectroSERVER. Steps 2 to 3 must be repeated if the models are destroyed on the secondary or tertiary SpectroSERVER.

More information:

Create an EventAdmin Model (see page 45)

Archive Manager Failure

If events are being sent to the EventAdmin model through the sbgwimport tool, additional fault tolerance has been implemented in case of an Archive Manager failure. If the Archive Manager goes down, alarms triggered by incoming events are created on the default event model. When the Archive Manager is running again, the events are sent to it and can be viewed in the event log.

Chapter 6: Southbound Gateway Case Study

This section contains the following topics:

Getting Started (see page 55)
Step 1: The AlertMap File (see page 55)
Step 2: The EventAdmin EventDisp File (see page 57)
Step 3: The EventModel EventDisp File (see page 57)
Step 4: The Event Format File (see page 57)
Step 5: The Probable Cause File (see page 58)
Step 6: Package for Distribution (see page 59)

Getting Started

This chapter walks you through an integration with a typical third-party management software that can send SNMP traps to CA Spectrum.

The traps sent by this software have been directed to the SpectroSERVER and are received by Southbound Gateway. For the sake of simplicity, this chapter assumes that this application only sends one type of trap to the Southbound Gateway. The trap is an enterprise-specific trap with a number of variable bindings.

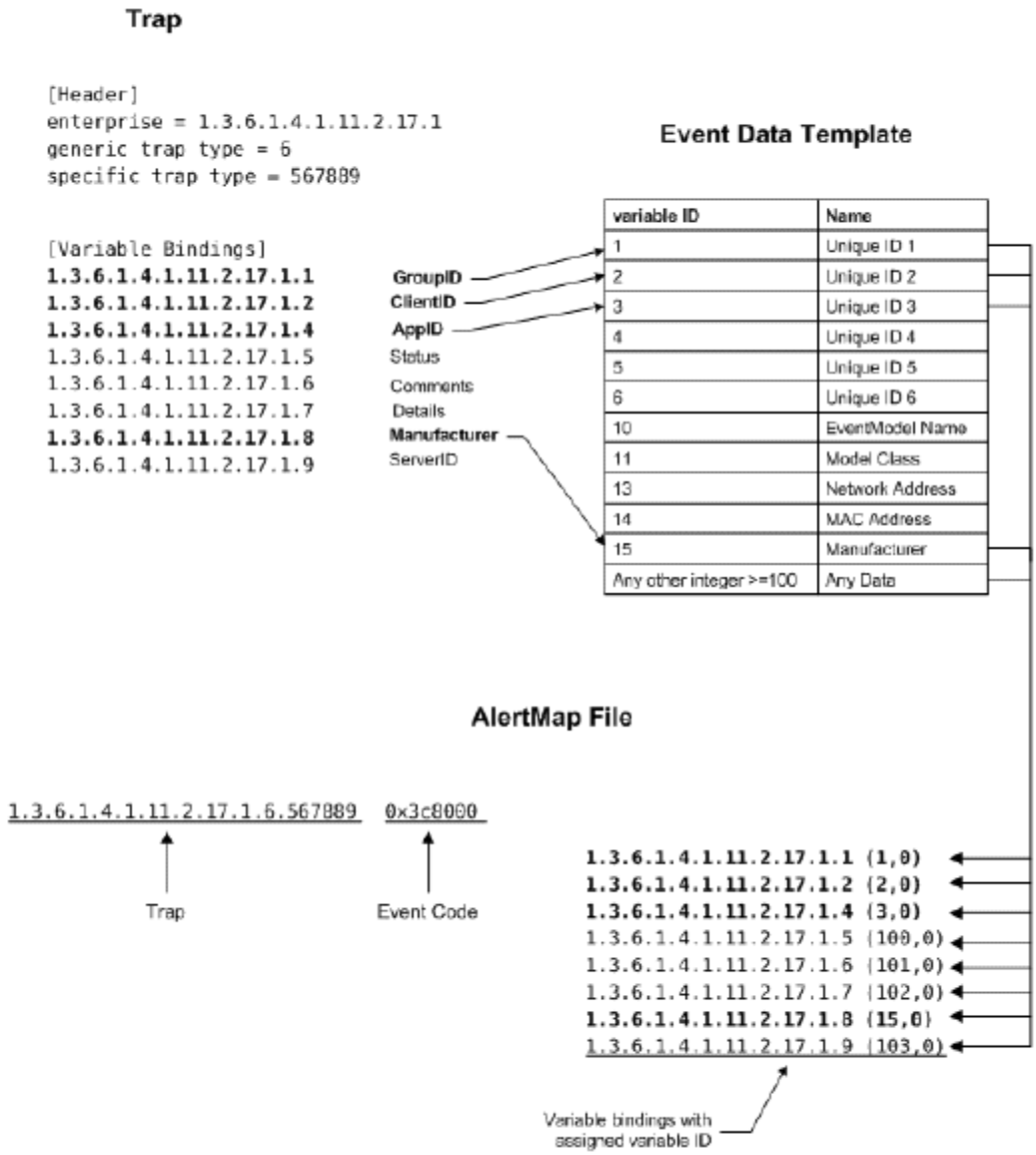
```
[Header]
enterprise = 1.3.6.1.4.1.11.2.17.1
generic trap type = 6
specific trap type = 567889

[Variable Bindings]
1.3.6.1.4.1.11.2.17.1.1
1.3.6.1.4.1.11.2.17.1.2
1.3.6.1.4.1.11.2.17.1.4
1.3.6.1.4.1.11.2.17.1.5
1.3.6.1.4.1.11.2.17.1.6
1.3.6.1.4.1.11.2.17.1.7
1.3.6.1.4.1.11.2.17.1.8
1.3.6.1.4.1.11.2.17.1.9
```

Step 1: The AlertMap File

The trap information must be mapped to an event; therefore, create an alert map entry for this trap. The most important thing to specify is the string that makes up the unique identifier for the EventModel. This unique ID can be comprised of up to six variable bindings using the variable IDs one through six. This data can then be viewed within the event message.

The following example shows the trap being mapped to an event in the AlertMap file. The variable bindings are being assigned a variable ID from the Event Data Template. All non-bold variable bindings have been mapped to the “Any Data” category of the Event Data Template.

These variable bindings have been given a variable ID that has not been assigned to a specific category in the Event Data Template.



Step 2: The EventAdmin EventDisp File

The following is a sample of the EventAdmin EventDisp file. This EventDisp file lists all the events generated by the third-party system. Listing them here lets them be received by the EventAdmin model.

Example: The EventAdmin EventDisp File

```
0x3c8000
```

The events should be logged at the EventModel level rather than the EventAdmin level. For this reason, the only thing listed in this file is the event code. This syntax indicates that the event should be received, but not logged. After the EventAdmin model receives the event, the EventAdmin model forwards it to the appropriate EventModel model.

Step 3: The EventModel EventDisp File

This EventDisp file gives the event received at the EventModel its properties. This file is optional, but highly recommended. These events are the same as those in the EventAdmin EventDisp file.

Example: The EventModel EventDisp File

```
0x3c8000 E 50 A 1,0x3c8000
```

Clearly this file has more data than the EventAdmin EventDisp file. The E indicates that the event is logged. The event severity is indicated by the number 50; event severity ranges from 1 to 100, 100 being the most severe. The A indicates that the event generates an alarm. The number 1 indicates the severity of the alarm. Valid values for the alarm severity are 1 through 6. The hexadecimal number 0x3c8000 is the alarm code.

There is also an optional ‘S’ flag available for use with EventDisp files which specifies whether an event should be registered for Southbound Gateway processing. It applies to modeltype-specific entries only and only those modeltypes derived from the southbound modeltype fragment. For more information about this flag and the rest of the EventDisp file syntax, see the *Event Configuration User Guide*.

Step 4: The Event Format File

The Event Format files define the event message seen in both the Events tab and the Alarm Details tab in OneClick.

Example: Event Format File

```
{d "%w- %d%m-, %Y - %T"} - Device {m} of
type {t}

The user's SMTP/POP3 mail transaction
failed with error code 554. The event
code is {e}.

Other information:
Group {S 1}
Client {S 2}
Application {S 3}
Manufacturer: {S 15}
Server: {S 103}
Status: {S 100}
Comments: {S 101}
Details: {S 102}
```

The items in parenthesis indicate that a variable value is placed in the event text message. The S indicates the data type (String) and the numeric value is the variable ID assigned in the AlertMap file representing the value that is placed there.

Step 5: The Probable Cause File

The Probable Cause files define the alarm title, symptoms, probable causes, and recommended actions.

Example: Probable Cause File

```
MAIL TRANSACTION ERROR RECEIVED

SYMPTOMS:
The client's attempt to send mail failed.

PROBABLE CAUSES:
1. Network connectivity.
2. The mail server.

RECOMMENDED ACTIONS:
1. Check the network connectivity.
2. Check the mail server operation.
```

Step 6: Package for Distribution

Once you have created all the files necessary for the integration, you need to package these files so that they can be installed on a SpectroSERVER. See the *CA Spectrum Extension Integration (SEI) Developer Guide* for specific instructions and examples.

Chapter 7: Creating a Southbound Gateway Demonstration

This section contains the following topics:

Introduction to Creating a Southbound Gateway Integration (see page 61)
Step 1: Edit the EventAdmin AlertMap File (see page 61)
Step 2: Edit the EventAdmin EventDisp File (see page 62)
Step 3: Edit the EventModel EventDisp File (see page 62)
Step 4: Create an Event Format File (see page 63)
Step 5: Create a Probable Cause File (see page 63)
Step 6: Create an EventAdmin Model (see page 64)
Step 7: Send the Trap (see page 65)
Step 8: Show the Results in OneClick (see page 66)

Introduction to Creating a Southbound Gateway Integration

This chapter helps you demonstrate the functionality of a Southbound Gateway integration without completing all the steps to create a fully functional integration.

In this demonstration, you are sending a fictitious trap to the EventAdmin model using a trap-generating software. The model receives the trap, translates the trap into a CA Spectrum event, creates an EventModel based on the event information, and forwards the event to an EventModel model. The event generates an alarm on this EventModel model and the text of the event format and probable cause files is shown in the Alarm tab and the Alarm Details tab in OneClick.

Important! Any changes made directly to AlertMap or EventDisp files are overwritten when a new version of CA Spectrum is installed or when an additional Southbound Gateway integration is installed. For this reason, use the steps in this chapter for demonstration purposes only and not as a replacement for the complete integration steps outlined in the chapters of this guide.

Step 1: Edit the EventAdmin AlertMap File

To allow the EventAdmin to receive an SNMP trap from a third-party system, the trap must be translated into a CA Spectrum event in the EventAdmin AlertMap file.

To edit the EventAdmin AlertMap file

1. Open the AlertMap file located in the following directory:

   `<$SPECROOT>/SS/CsVendor/gen_app_gw/EventAdmin`

2. Add the following line to the AlertMap file:

   ```
   1.3.6.1.4.1.1850.6.1 0x5990001 1.3.6.1.4.1.1850.1.0.0.1(1,0)
   ```

   This line indicates that the trap, 1.3.6.1.4.1.1850.6.1, is translated into event 0x5990001. The value variable ID 1 (indicated in parentheses) shows that the Event Data Template variable of 1 is being used to represent the value of the variable binding, 1.3.6.1.4.1.1850.1.0.0.1. Because the Event Data Template variable 1 is one of the unique identifier values, the value of the variable binding is used as a unique identifier for the event model.

3. Save and exit the AlertMap file.

More information:

AlertMap File Syntax (see page 21)
The Event is Processed by the EventAdmin (see page 66)

Step 2: Edit the EventAdmin EventDisp File

For the event created in the AlertMap file to be passed along to the EventModel for processing, add the event code to the EventAdmin EventDisp file.

To edit the EventAdmin EventDisp file

1. Open the EventAdmin EventDisp file located in the following directory:

   ```
   SS/CsVendor/gen_app_gw/EventAdmin
   ```

2. Add the following line to this EventDisp file:

   ```
   0x5990001
   ```

3. Save and exit the EventDisp file.

More information:

EventAdmin EventDisp File (see page 36)

Step 3: Edit the EventModel EventDisp File

For the EventModel model to process the event, create processing instructions in the EventModel model EventDisp file.

To edit the EventModel EventDisp file

1. Open this file from the following directory:

   <$SPECROOT>/SS/CsVendor/gen_app_gw

2. Add the following line to this file:

   ```
   0x5990001 E 50 A 1, 0x5990001
   ```

   This line indicates that the event should create a minor alarm on the EventAdmin model.

More information:

EventModel EventDisp File (see page 37)
An Alarm is Asserted on the EventModel (see page 66)

Step 4: Create an Event Format File

The event format file lets users get information about the event in the Events tab in the Component Detail view in OneClick.

To create the event format file for this event

1. Create a text file with the following content:

   ```
   {d "%w- %d %m-, %Y - %T"} - SAMPLE - This is a sample event format file for the
   Southbound Gateway demonstration.
   The event processed is : event {e}.
   The variable binding data sent with the trap is: {S 1}.
   ```

2. Save this text file as *Event05990001* in the following directory:

   <$SPECROOT>/SG-Support/CsEvFormat

3. Verify that your text editor does not add a file extension to this text file and then close the text file.

More information:

Presentation Format of Events (see page 37)
View Data About the Alarm in OneClick (see page 67)

Step 5: Create a Probable Cause File

The text of the probable cause file is shown in the Alarm Details tab of the Component Detail view and provides information about the causes and resolutions for the alarm.

To create the probable cause file for this alarm

1. Create a text file with the following content:

```
SAMPLE PROBABLE CAUSE FILE FOR THE SOUTHBOUND GATEWAY DEMONSTRATION
SYMPTOMS:
The symptoms of the problem are listed here
PROBABLE CAUSES:
1) Cause 1.
2) Cause 2.
RECOMMENDED ACTIONS:
1) Action 1.
2) Action 2.
```

2. Save this text file as *Prob05990001* in the following directory:

   <$SPECROOT>/SG-Support/CsPCause

3. Verify that your text editor does not add a file extension to this text file and then close the text file.

More information:

Add Value to Alarms (see page 39)
View Data About the Alarm in OneClick (see page 67)

Step 6: Create an EventAdmin Model

Complete this procedure to create an EventAdmin model for the purposes of this demonstration.

To create an EventAdmin model in the Universe topology

1. Click the Create new model by type button in the Topology tab.

   The Select Model Type dialog appears.

2. In the Containers tab, click EventAdmin and click OK.

   The Create Model of Type EventAdmin dialog appears.

3. Configure the following parameters:

   a. Enter Test in the Name field.

   b. Enter the network address of the computer from which you launch the trap in the Network Address field.

      If you are using a trap generation program for the purposes of this demonstration, enter the network address of the computer running the trap generation program.

   c. Leave the Security String field blank.

d. Leave the Manager Name field as Default.

e. Leave the EventModel Prefix field blank.

f. Click OK.

The EventAdmin model is now created and appears in the Universe topology.

More information:

Create an EventAdmin Model (see page 45)

Step 7: Send the Trap

After you have finished Steps 1 through 6, shut down and restart the SpectroSERVER to activate the changes you have made.

After you have activated your changes, send a trap to the Southbound Gateway to show how this trap is handled.

You can use a trap generating software such as Trap Generator from Network Computing Technologies to send the trap to the SpectroSERVER. The following procedure uses Trap Generator to illustrate how to send a trap to Southbound Gateway.

Note: The Trap Generator shareware is being used in this example to demonstrate a process, but is not directly supported by CA. For information about downloading, licensing, and using Trap Generator, see the Network Computing Technologies web site: http://www.ncomtech.com/trapgen.html (www.ncomtech.com/download.htm).

To send a trap to Southbound Gateway using Trap Generator

1. If you are using Trap Generator, use the following syntax in your parameter text file:

```
-d <IP Address of SpectroSERVER>:162
-c public
-o 1.3.6.1.4.1.1850
-g 6
-s 1
-v 1.3.6.1.4.1.1850.1.0.0.1 STRING "TestEventModel"
```

2. After you have created this text file, launch the trap by typing:

```
trapgen -f <name of parameter text file>
```

This syntax directs the trap to port 162 on the SpectroSERVER and defines a variable binding, data type, and value.

Step 8: Show the Results in OneClick

The following process shows the results in OneClick:

1. The Alert is Received and Translated into a CA Spectrum Event (see page 66).
2. The Event is Processed by the EventAdmin (see page 66).
3. An Alarm is Asserted on the EventModel (see page 66).
4. View Data About the Alarm in OneClick (see page 67).

The Alert is Received and Translated into a CA Spectrum Event

When you launch the trap from the trap generation program, the EventAdmin model receives and processes the event.

More information:

Step 8: Show the Results in OneClick (see page 66)

The Event is Processed by the EventAdmin

When the EventAdmin model processes the event, it uses the variable binding used in Step 1: Edit the EventAdmin AlertMap File (see page 61) as the unique identifier to specify which EventModel to forward the event to. Since this EventModel does not yet exist, a new one is created. To view the EventModel, from the Universe Topology, drill down into the EventAdmin container.

More information:

Step 8: Show the Results in OneClick (see page 66)

An Alarm is Asserted on the EventModel

In Step 3: Edit the EventModel EventDisp File (see page 62), an entry was added to the EventModel's EventDisp file to indicate how the event should be processed. This entry indicated that the event should create a minor alarm.

The EventModel model has a status of yellow indicating a minor alarm.

More information:

Step 8: Show the Results in OneClick (see page 66)

View Data About the Alarm in OneClick

You can view data about the alarm from the Alarm Details tab in OneClick.

To view information about this alarm

1. Select the Event Model in the Explorer tab in the Navigation panel.
2. Click the Events tab in the Contents panel to view the contents of the event format file created in Step 4: Create an Event Format File (see page 63).
3. Click the Alarms tab in the Contents panel, and then click the Alarm Details tab in the Component Detail view to review the contents of the probable cause file created in Step 5: Create a Probable Cause File (see page 63).

More information:

Step 8: Show the Results in OneClick (see page 66)

Index

.

.sbgwimport.dtd • 23

A

alarms • 39, 67
alert code • 21
alert data xml file • 26
AlertMap file
  about • 55
  defined • 14
  editing • 61
  location • 20
  syntax • 21
  use • 20
Archive Manager, failure • 54

C

CA Spectrum Extension Integration toolkit • 41
CA Unicenter NSM Agent • 19
CLI • 27
CORBA • 27

D

data flow • 16
distribution • 41, 59
Documenr Type Definition (DTD) • 9

E

error 0x3dc0000 • 28
event code • 21
Event Data Template • 14, 27, 31, 37
event format file • 14, 37, 57, 63
event forwarding • 45
event messages • 34
EventAdmin
  about • 13, 17, 49, 61
  creating • 45, 64
  moving • 47
  status • 47
  viewing • 49
eventAdminName • 23
EventCondition Event Rule • 37
EventDisp • 14, 17, 35, 36, 37, 57, 62
EventModel
  about • 13, 17, 51, 61
  alarms on • 66
  attributes • 33, 34, 52
  creating • 50
  moving • 51
  Name • 33
  prefix • 45, 52
eventModelName attribute • 23
EventModels in • 52
events • 17, 23, 66
eventType • 23
extension integration toolkit • 41

F

fault-tolerant environment • 53

H

host agents • 19
host model types
  about • 15
  using • 49

I

iAgent • 19
index file
  and AlertMap files • 20
  creating • 42
InformRequests • 20
installation script • 41
instance variable ID • 21, 27

M

MAC address attribute • 34
maintenance mode • 52
Manager Name • 45
manufacturer attribute • 34
MergeVendorFiles.cus • 41
mkcd • 44
mkmm • 44
model class attribute • 34
model name
  case insensitive • 28
  case sensitive • 28

N

Name • 45
Network Address • 45
network address attribute • 34
network management • 9
non-SNMP environments • 9, 23

O

OID • 21
OID map • 21

P

ParseMaps • 19
part description file • 42
prerequisites • 11
probable cause file • 14, 39, 58, 63

R

RFC 2576 • 20

S

sbgwimport • 23, 26
SBGWImport.java • 27
secure domain • 19
Security String • 45
SNMP traps • 17, 55
snmp_trap_port parameter • 17
SNMPv2 • 20
Southbound Gateway import tool • 14
SystemEDGE • 19

T

Target Address field • 33
Target Name Case Insensitive field • 34
Target Name field • 32
targetAddress • 23
targetName • 23
targetNameIgnoreCase • 23
trapIPAddress • 45
traps • 65

U

uniqueId • 23

V

value variable ID • 21, 27
variable bindings • 21

X

xml • 23