Solstice Cooperative Consoles Administration Guide

2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

SunSoft
A Sun Microsystems, Inc. Business

Please
Recycle

Contents

Figures

Tables

Preface

The *Solstice Cooperative Consoles Administration Guide* provides information on the functions and features of Cooperative Consoles 1.2 for Solstice Site/SunNet/Domain Manager.

Who Should Use This Book

This document is intended for network administrators who set up and configure Cooperative Consoles for information sharing between Site/SunNet/Domain Manager Console instances running on multiple hosts.

Installation Information

For information on installing Cooperative Consoles software, refer to the appropriate *Site/SunNet/Domain Manager Installation Guide.*

How This Book Is Organized

This document is organized as follows:

Chapter 1, "Introduction," provides an overview of the components of Cooperative Consoles.

Chapter 2, "Cooperative Consoles Configurations," describes a number of possible configurations in the information forwarding relationships among Site/SunNet/Domain Manager Consoles with Cooperative Consoles installed.

Chapter 3, “Cooperative Consoles Operation,” describes the function of Cooperative Consoles and its underlying architecture.

Chapter 4, “Using the Configuration Tool,” describes the use of the Configuration Tool to customize the sharing of information between management stations.

Chapter 5, “Cooperative Consoles Examples,” provides examples of filter file entries.

Appendix A, “Diagnosis,” provides some information on what you can do when encountering problems using CC.

Compatibility

See the *Site/SunNet/Domain Manager Release Notes* for compatibility information.

Conventions Used in This Book

Command Line Examples

All command line examples in this guide use the C-shell environment. If you use either the Bourne or Korn shells, refer to `sh`(1) and `ksh`(1) man pages for command equivalents to the C-shell.

What Typographic Changes and Symbols Mean

The following table describes the type changes and symbols used in this book.

Table P-1 Typographic Conventions

Typeface or Symbol	**Meaning**	**Example**
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `system% You have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`system% su` `Password:`
<AaBbCc123>	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *<filename>*`.`
AaBbCc123	Book titles, new words or terms, or words to be emphasized	These are called *class* options. You *must* be root to do this.
Code samples are included in boxes and may display the following:		
`%`	UNIX C shell prompt	`system%`
`$`	UNIX Bourne and Korn shell prompt	`system$`
`#`	Superuser prompt, all shells	`system#`

Mouse Conventions

This book assumes that you are using a standard Sun workstation three-button mouse. The mouse buttons are called SELECT (left), ADJUST (middle), and MENU (right).

Click means to press and quickly release a mouse button.

Press indicates you should hold the button down until an action is completed, such as a menu appearing.

Introduction 1

Cooperative Consoles 1.2 implements the sharing of information between multiple instances of the Site/SunNet/Domain Manager Console.

Geographically dispersed organizations with large networks often need a division to handle network management responsibilities among multiple management Consoles. For such network environments, Cooperative Consoles (CC) provides the ability to implement the forwarding of information about selected changes in the state of critical network devices or changes in selected aspects of network topology between multiple management stations running Site/SunNet/Domain Manager. CC provides the flexibility to implement a variety of possible information-sharing configurations between multiple Site/SunNet/Domain Manager consoles.

Note – CC assumes that version 2.3 of Site/SunNet/Domain Manager has been installed for management stations on the Solaris 1.x and Solaris 2.x environments.

Note – At this release, the CC_Receiver is supported with Domain Manager only. Site Manager and SunNet Manager can only function as a sending station.

1

1.1 Cooperative Consoles Operation

CC currently enables the forwarding of four types of information between Site/SunNet/Domain Manager consoles:

- **SNM Events** — These events are generated by agents in response to SNM event requests when conditions specified by the event request (such as a device being unreachable) are satisfied.
- **Traps** — These are unsolicited events, not generated in response to SNM event requests; an example is a Simple Network Management Protocol (SNMP) `linkDown` trap.
- **SNM Database Traps** — The Site/SunNet/Domain Manager Console generates traps when changes are made to the SNM database, such as the addition of a new element or loading of a background image for a view.
- **Glyph Traps** — Glyph Traps are generated when the glyph state is changed manually on the Console.

To implement information forwarding between Site/SunNet/Domain Manager consoles, CC uses three executable software modules:

- **Receiver Application** — A Receiver is installed on each Domain Manager Console machine that is to receive forwarded event and topology information from other Site/SunNet/Domain Manager consoles. A Domain Manager Console machine with a Receiver process running is a *receiving station*.
- **Sender Daemon** — A Sender daemon is installed on each Site/SunNet/Domain Manager host that is to forward event and topology information to remote Site/SunNet/Domain Manager consoles. A Site/SunNet/Domain Manager Console machine with a Sender process running is a *sending station*.
- **Configuration Tool** — This tool is the user interface for configuring operation of the Sender and Receiver processes on the local host.

The Receiver and Sender processes are described in Chapter 3, “Cooperative Consoles Operation.” The use of the Configuration Tool is explained in Chapter 4, “Using the Configuration Tool.”

A Site Manager and a SunNet Manager Console machine can function as a sending station only. A Domain Manager Console machine can function as both a sending station and a receiving station, or it can serve as only a sending

station, or as only a receiving station. The particular distribution of the Sender and Receiver on the Site/SunNet/Domain Manager Console machines in your network depends upon the desired network management configuration. Several sample configurations are discussed in Chapter 2, “Cooperative Consoles Configurations.”

In addition to the Receiver and Sender processes, the SNM Event Dispatcher (`na.event`) also plays an important role in the functioning of CC on both receiving and sending stations.

The information flow between Site/SunNet/Domain Manager consoles with CC installed is illustrated in Figure 1-1.

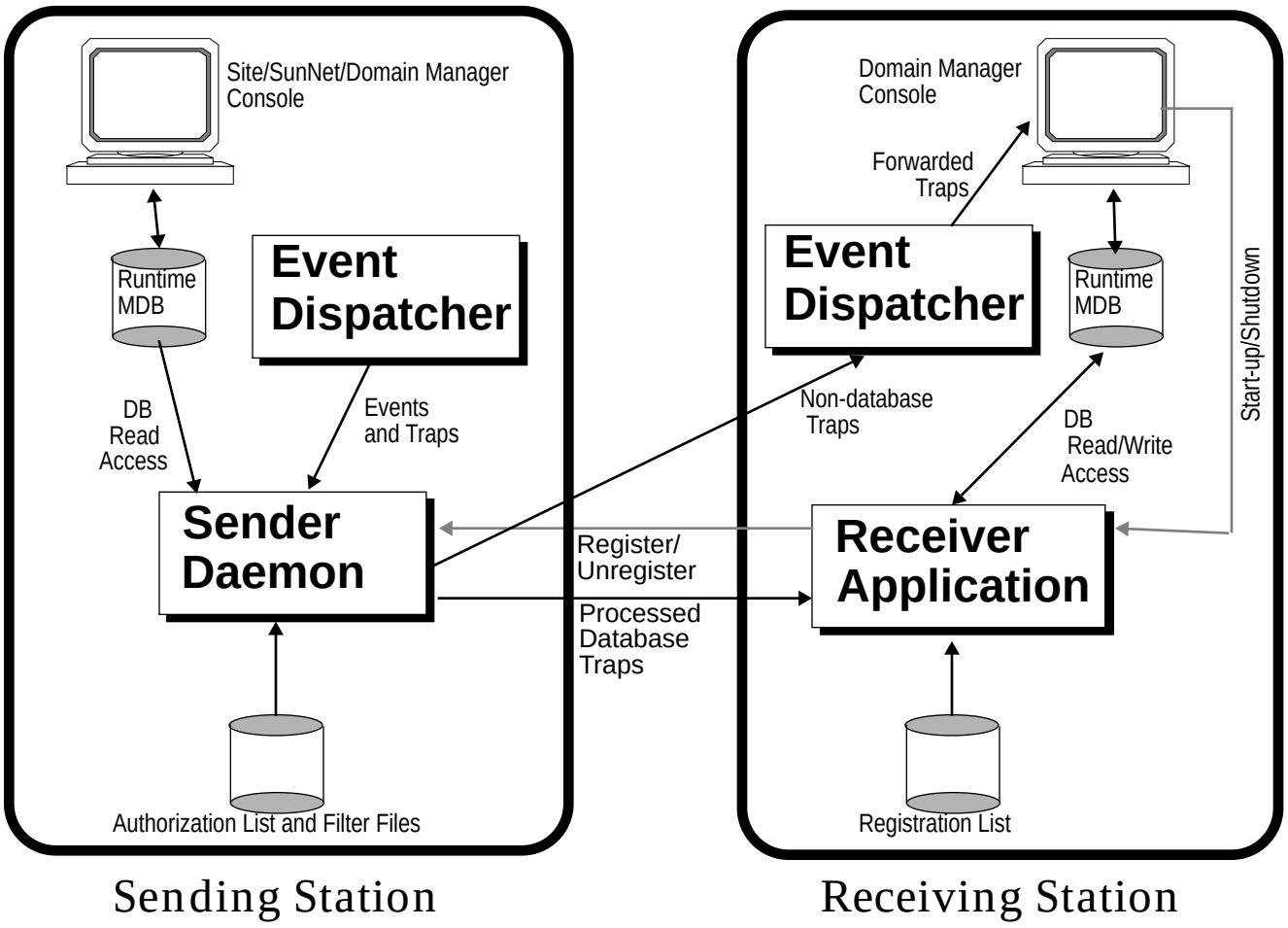

Figure 1-1 Cooperative Consoles Operation

1.2 Adding CC to the Site/SunNet/Domain Manager Console Tools Menu

After installing the CC software, you need to load the `cooptools.schema` file into Site/SunNet/Domain Manager in order to add the Cooperative Consoles Receiver (Domain Manager only) and Configuration Tool to the Site/SunNet/Domain Manager Console's Tools menu. The methods for loading the `cooptools.schema` file are:

- Restart Site/SunNet/Domain Manager with the `-i` option.

  If you start Site/SunNet/Domain Manager with the `-i` option after installing CC, this forces Site/SunNet/Domain Manager to reload all the schema files, including `cooptools.schema`.

Note – Starting Site/SunNet/Domain Manager with the `-i` option erases your runtime management database. To save your current runtime database, you need to use the Console File menu Save➤Management Database option to save your runtime database to an ASCII file. You can then reload this database using the Load➤Management Database option after restarting Site/SunNet/Domain Manager with the `-i` option.

- Load the `cooptools.schema` file from the Site/SunNet/Domain Manager Console's File menu, then restart Site/SunNet/Domain Manager.

  To use this method, pull down the Site/SunNet/Domain Manager Console's File menu and select the Load➤Management Database option. The `cooptools.schema` file is located in the following directory:

  - `/opt/SUNWconn/snm/struct` for Solaris 2.x environments
  - `/usr/snm/struct` for Solaris 1.x environments

  Select the `cooptools.schema` file, as shown in Figure 1-2, and then click SELECT on the Load button.

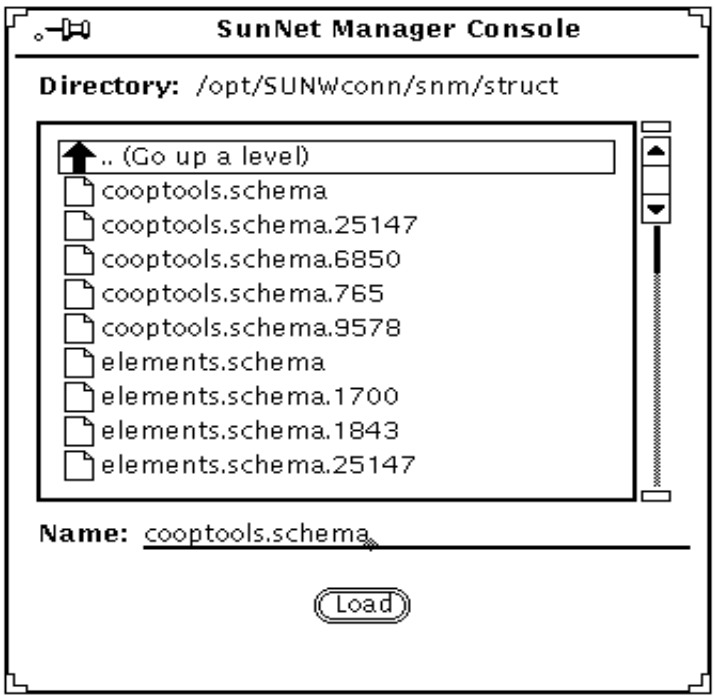

Figure 1-2 Loading the `cooptools.schema` file

After loading the `cooptools.schema` file, quit Site/SunNet/Domain Manager, then restart Site/SunNet/Domain Manager again. When you restart Site/SunNet/Domain Manager, you do not need to use the `-i` option.

1.3 Updating Schema Files

When a new schema file is added or when a record is added to an existing schema file, you need to run the utility program `modify_el` located in `/opt/SUNWconn/snm/bin` for Solaris 2.x and `/usr/snm/bin` for Solaris 1.x.

The default location will be used when `modify_el` is specified without an argument. If you used an install directory other than the default installation path, you need to run the program as follows:

% **modify_el** <*installation_path*>**/SUNWconn/snm**.

1.4 SNM Database/Glyph Trap Forwarding

To ensure that *all* SNM database/glyph traps are sent to the local Sender daemon, make sure that the attribute `snm.console.DBMgrTrapAlways` (for database traps) or `snm.console.sendGlyphTraps` (for glyph traps) is set to TRUE in your `.SNMdefaults` file. The default setting for this SNM attribute is FALSE. You can use `vi` or your favorite text editor to change the setting of this attribute. If no `.SNMdefaults` file exists in your home directory, you can force Site/SunNet/Domain Manager to create one by making a minor change to some Console attribute from the Console Properties menu.

Cooperative Consoles Configurations 2

Cooperative Consoles (CC) provides the flexibility to implement a variety of possible cooperative relationships between multiple Site/SunNet/Domain Manager Consoles. Three possible configurations are described in this chapter.

2.1 Peer-to-Peer Forwarding

A peer-to-peer relationship exists among Domain Manager consoles when the Domain Manager machines both send and receive event and topology information to each other. The peer-to-peer configuration implies that each Domain Manager Console machine has both a Sender and Receiver software installed, and thus functions as both a receiving and sending station.The following example illustrates peer-to-peer forwarding relationships.

Note – Because the CC_Receiver is supported with Domain Manager only, the Peer-to-Peer relationship solely exists between Domain Manager consoles.

Company PQR has separate Domain Manager consoles to manage regional networks. A network management station in San Francisco is responsible for managing a west coast region and a station in New York is responsible for an east coast region. The network manager in New York wants to know about all changes to PQR's backbone network (routers, WAN link) and critical servers (financial database server) in the west coast region. Events are filtered by the Sender on the San Francisco machine on the basis of host name (selecting the servers) and component type (selecting the routers) and forwarded to the Domain Manager machine in New York. The relevant elements reside in a view

on the San Francisco machine called “WestNet.” The Receiver process on the New York machine places forwarded topology information under a view also called “WestNet” on the New York machine.

The network management station in San Francisco also wants to receive this type of information from the network management console in New York. The Receiver process on the San Francisco machine places forwarded topology information under a view called “EastNet,” mirroring the view name on the New York machine. A Sender and a Receiver are therefore installed on both machines to implement this peer-to-peer relationship. Figure 2-1 illustrates this example.

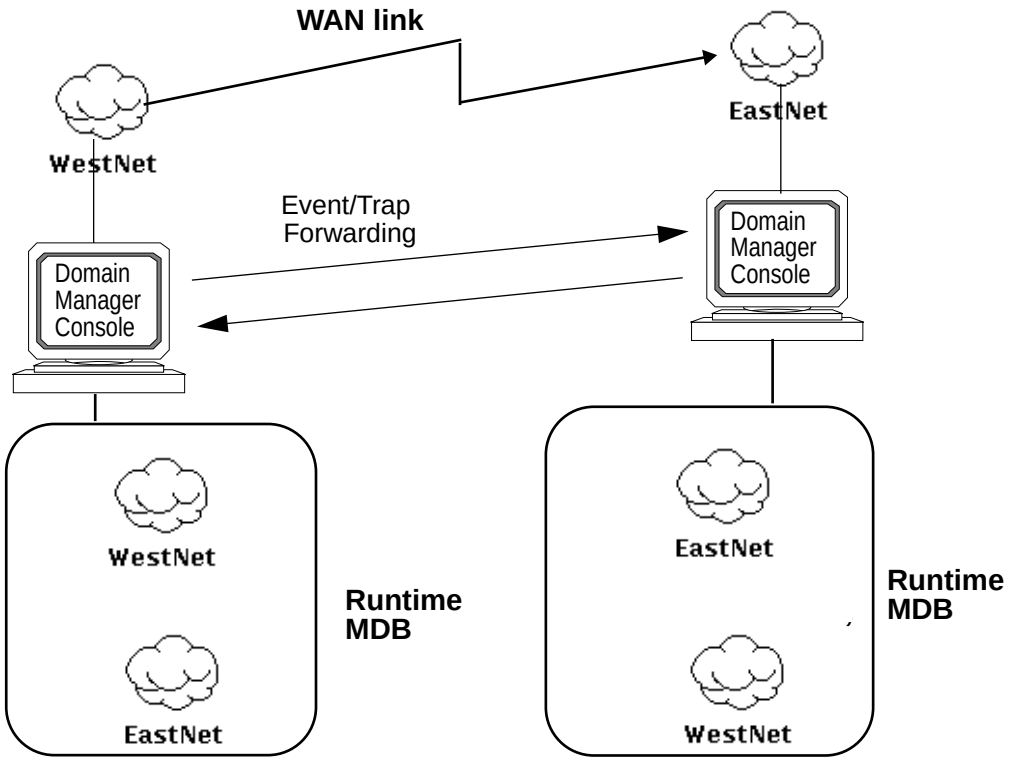

Figure 2-1 Example of Peer-to-Peer Relationship Between Regional Stations

Note – Peer-to-Peer configuration is not supported on the same machine.

2.2 Periphery-to-Center Forwarding

In a periphery-to-center configuration, the flow of forwarded alarm and topology information is from distributed Site / SunNet / Domain Manager consoles to a central Domain Manager Console. For example, an organization can have multiple Site / SunNet / Domain Manager consoles responsible for regional components of its network while also having a Central Domain Manager Console that needs to display and monitor the global network topology and state of critical devices. The flow of information, in this configuration, is one-directional, from the regional management stations to the central management station. This configuration is illustrated in Figure 2-2.

Note – Because the CC_Receiver is supported with Domain Manager only, the central console must be a Domain Manager Console machine so that it can receive information.

In this configuration, the Sender daemon needs to be installed only on the regional Site / SunNet / Domain Manager Console machines. The Receiver needs to be installed only on the central Domain Manager Console machine. Start-up and shutdown of CC is initiated from the central Domain Manager Console.

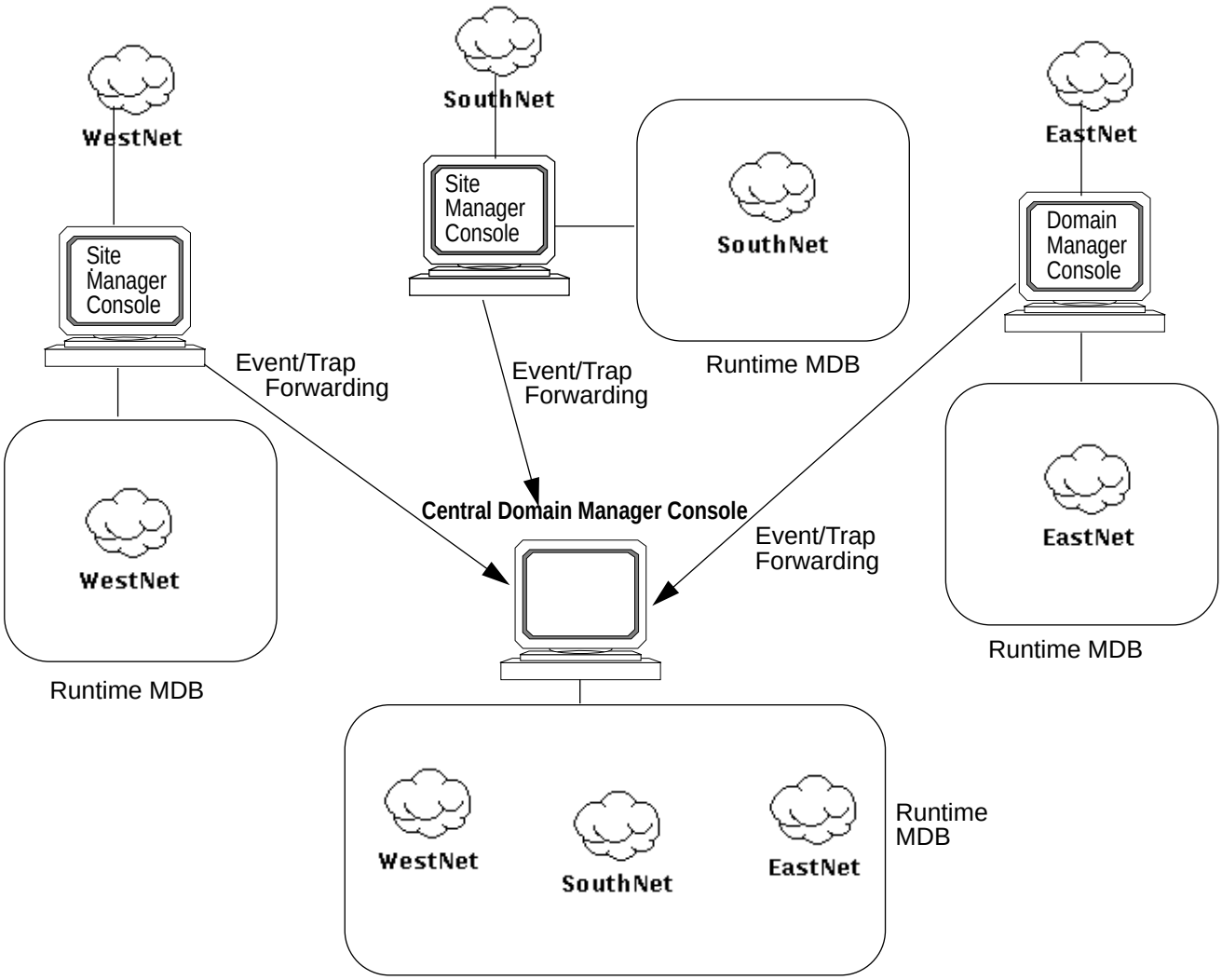

Figure 2-2 Forwarding of Information to Central Management Station

2.2.1 Distributed Management with Functional Specialization

A variation on periphery-to-center forwarding of information is a situation where regional Domain Manager consoles have management responsibility for a type of network link (for example, X.25, Frame Relay, ISDN) or type of network device (routers, database servers, T1 links) for the enterprise-wide network. In this configuration, illustrated in Figure 2-3, no central Domain Manager Console machine functions as a single network management center. Rather, each Domain Manager Console receives topology and event information from other Domain Manager consoles about devices of a particular type it is responsible for, in addition to managing its own regional network. Each Domain Manager Console functions as the *center* only for a particular type of network link or type of network device.

As in a peer-to-peer configuration, both a Sender and Receiver are needed on each Domain Manager Console machine in this configuration. The Sender daemon is required to forward information that pertains to the types of devices the other Domain Manager consoles are interested in. The Receiver application functions as a receiving station, to receive information about the particular device types the local Domain Manager Console manages for the enterprise-wide network.

In the example in Figure 2-3, the Receiver process on Domain Manager#2, which manages the western regional network, registers with the Sender daemons on each of the other Domain Manager Console machines to receive information about X.25 links. Each of the sending stations must have a filter table available that forwards only X.25-related event and topology information. The Receiver process on Domain Manager#2 passes the name of this filter file when it registers with the Sender daemons on the other stations. Domain Manager#2 functions as the *central* Domain Manager Console only in regard to the X.25 network links.

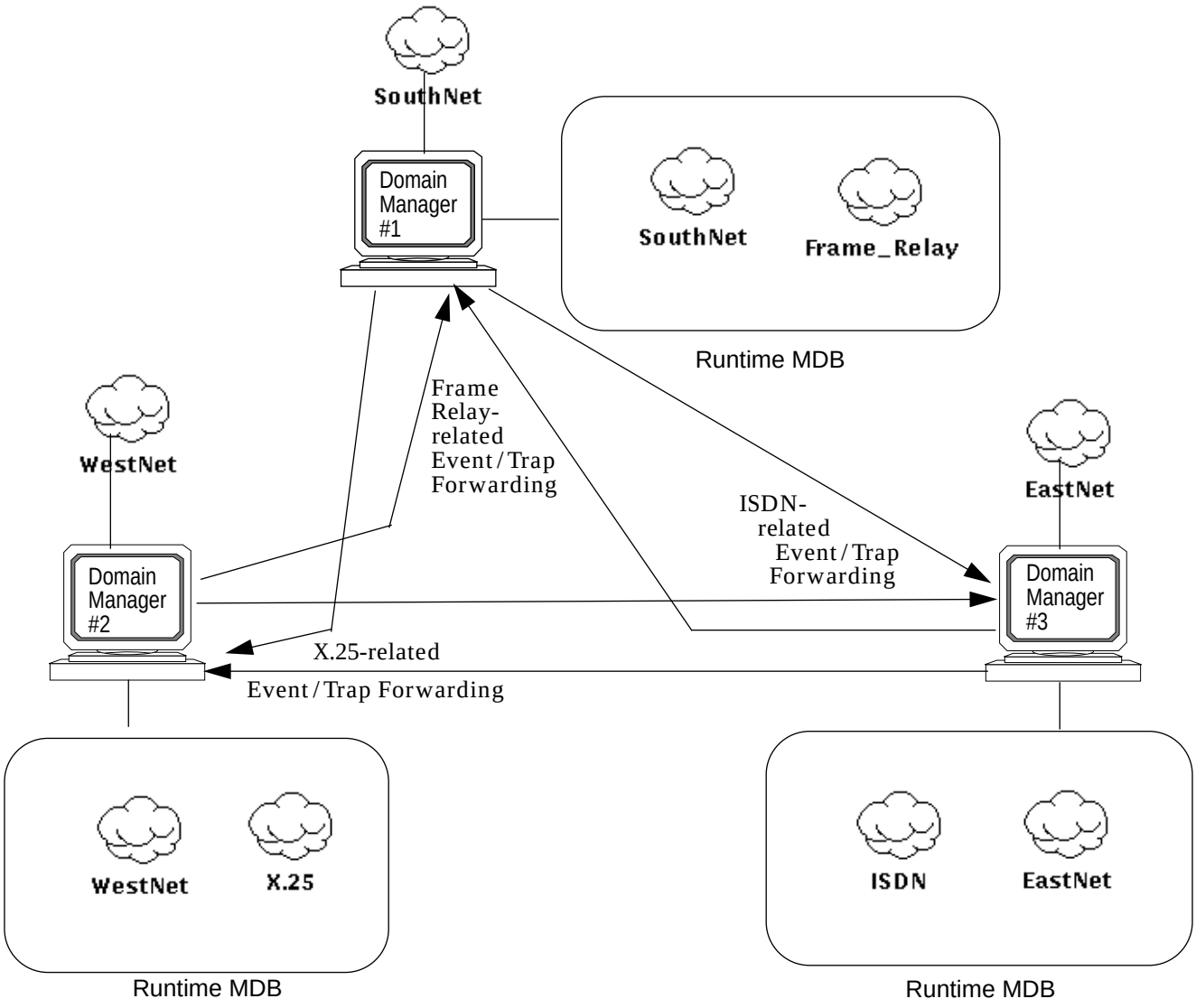

Figure 2-3 Functional Specialization in Multiple SNM Consoles

2.3 Center-to-Periphery Forwarding

In a center-to-periphery configuration, topology changes and alarms are propagated from one or more central Site/SunNet/Domain Manager consoles to distributed Domain Manager consoles to off-load responsibility for managing particular regions, type of network, or type of device. This scenario is illustrated in Figure 2-4.

Note – Because the CC_Receiver is supported with Domain Manager only, the peripheral consoles must be a Domain Manager Console machine so that it can receive information.

In this configuration, the central Site/SunNet/Domain Manager Console requires a Sender daemon but not a Receiver. Each peripheral Domain Manager console machine has a Receiver installed but does not require a Sender daemon.

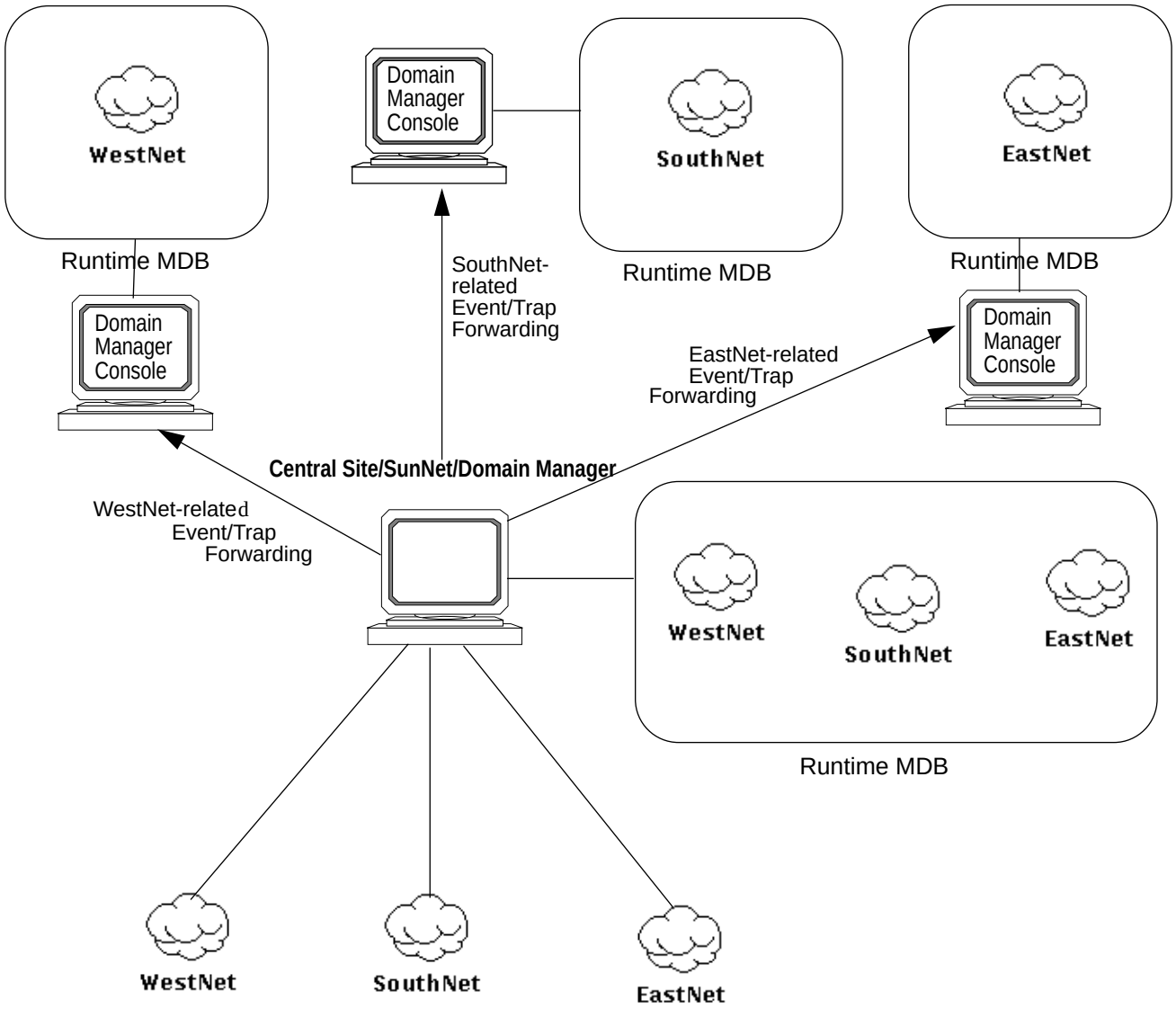

Figure 2-4 Forwarding of Information from Center to Periphery

Cooperative Consoles Operation 3

3.1 Cooperative Consoles Start-up

The Cooperative Consoles Receiver is the user entry point for starting the exchange of information between multiple Site/SunNet/Domain Manager Consoles using Cooperative Consoles (CC). Installation of the Cooperative Consoles Receiver on a management station adds the Receiver to the Domain Manager Console's Tools Menu on that machine, as shown in Figure 3-1.

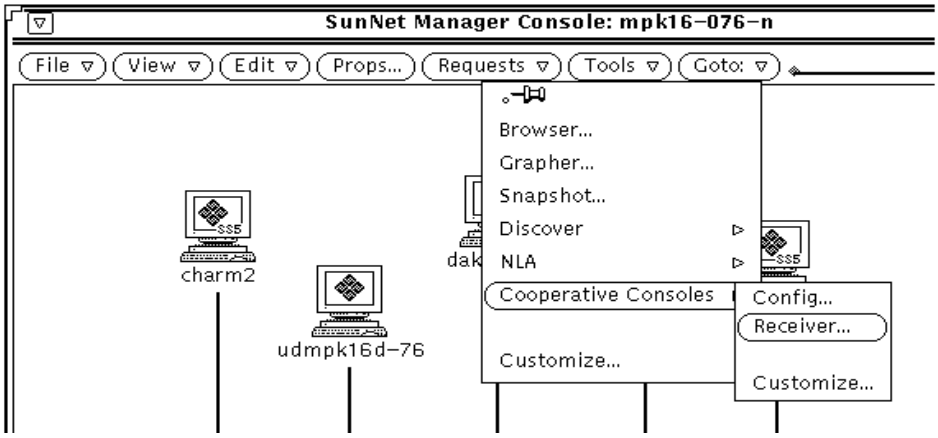

Figure 3-1 Starting the Cooperative Consoles Receiver

3

When you pull down the Domain Manager Console's Tools menu and select the Cooperative Consoles Receiver, the Receiver window appears, as shown in Figure 3-2. When launched, the Receiver process attempts to register with the Sender process on those Site/SunNet/Domain Manager Console machines on the Receiver's Registration List. The Registration List also specifies the specific database and event-forwarding criteria (Filter Table) to be used by the remote Sender processes. To build the Receiver's Registration List, see Chapter 4, "Using the Configuration Tool." The Receiver window provides information about the status of the connection with remote sending stations.

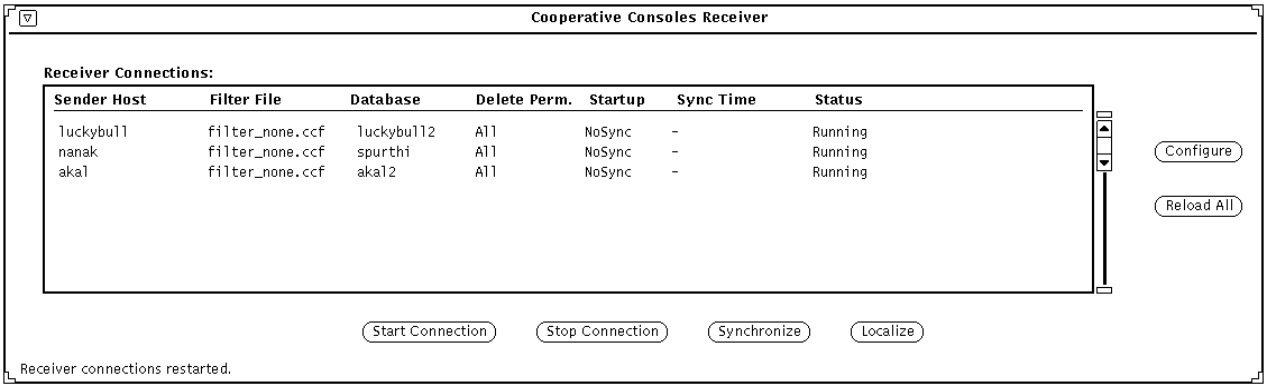

Figure 3-2 Receiver Window

The six buttons on the Receiver window have the following uses:

- **Start Connection** — This button enables you to reregister with selected sending stations. If you click SELECT on a target host entry, then click SELECT on the Start Connection button, the Receiver process attempts to register with the Sender process on the target host that you have selected.
- **Stop Connection** — This button allows you to shutdown forwarding of information from selected hosts to the local Domain Manager console. If you click SELECT on a target host entry, then click SELECT on the Stop Connection button, the Receiver process unregisters with the Sender on that remote host.

- **Synchronize** — This button allows you to manually synchronize the local runtime database with the selected sending station. If you click SELECT on a target host entry, then click SELECT on the Synchronize button, all database information for the selected connection is re-initialized in a two-step process:

  First, all existing elements in the database on the local machine that match the selected connection are deleted. The data for the selected connection is matched using the `Created by cc` field (<*hostname*>:<*filter-table*>:<*database-name*>). The `Created by cc` field is included in the properties sheet for that element; see Figure 3-3.

  Second, the Receiver sends a synchronization request to the target sending station. The Sender then sends all topology information that passes the pertinent filters for the selected entry.

- **Localize** — If you select one of the connections in the Receiver window, then click on the Localize button, every element in the local database that has a `Created by cc` field that matches the selected connection in its Created by cc field (<*hostname*>:<*filter-table*>:<*database-name*>) will have its `Created by cc` field blanked out, as if that element had been created by the local console rather than via the CC Receiver. This prevents the Receiver from deleting these elements when a synchronization is executed.

- **Configure** — Click SELECT on the Configure button to launch the Configuration Tool. For information on how to use the CC Configuration Tool, refer to Chapter 4, “Using the Configuration Tool.”

- **Reload All** — If you click SELECT on the Reload All button, the Receiver unregisters with all remote Senders, rereads the Receiver’s Registration List, then reregisters with the remote Senders on the Registration List. If you change the Receiver’s Registration List while the Receiver is running — for example, by adding new stations — use the Reload All button to force the Receiver to use the updated Registration List.

You can quit from the Receiver by pressing MENU over the control bar at the top of the window and selecting Quit from the control bar menu.

3.2 Receiver Operation

When you first start the Receiver, the Receiver process attempts to register with the Sender process on each machine specified on a list of target management stations. When registering with a Sender on a remote station, the Receiver can pass a user database name that the Sender process uses to select database information when forwarding database traps from that machine. Because multiple user databases can exist on the Site/SunNet/Domain Manager machine, a user database name is needed to specify which database the Sender process is to access. The Sender accesses the database file named

`db.`<*database-name*>

where <*database-name*> is passed by the Receiver at the time of registering with the Sender.

Note – Do not enter `db.` as part of the database name; the `db.` is generated automatically.

The Receiver process also passes to each Sender the name of the filter file to use for selecting event and topology information for forwarding to the receiving station.

When the Receiver is first launched from the Console Tools Menu, it attempts to register with the four target hosts, as shown in Figure 3-2. As the Receiver attempts to register with the Sender at `mgr.West.Sun.Com`, it will pass the value “west_coast” to indicate that the Sender should access the database file `db.west_coast`. The Receiver also passes the filter file name `criticalnodes.ccf`, that `mgr.West.Sun.Com` is to use for selecting information to forward.

Registration with a remote Site/SunNet/Domain Manager console machine is successful only if the requesting Receiver is on the target Sender’s list of stations authorized to receive forwarded information. (Use the Configuration Tool on the sending station to define the list of remote stations authorized to register with the local Sender process, as described in Chapter 4, “Using the Configuration Tool.”)

A Receiver’s attempt to register with a Sender process fails if the Receiver passes the name of a nonexistent filter file or passes a database name it is not authorized to access. Receivers running on different hosts can request use of different Filter files when registering with the same Sender daemon.

3.2.1 Receipt of Topology Information

After a Receiver process successfully registers with a Sender process on a remote station, topology type information is forwarded from the Sender process to the Receiver process according to user-configurable criteria. (Use the Configuration Tool on the sending station to define the Sender's processing of information to be forwarded.) Multiple Receivers on various hosts can register with the Sender on a given Site / SunNet / Domain Manager Console machine and the criteria used for forwarding can be configured separately for each receiving station.

The Receiver process receives only topology data (SNM database traps) from the Sender.

Note – For other information like the SNMP traps, SNM events, and the SNM glyph traps, the sender forwards this information to the Event Dispatcher process directly on the receiving station.

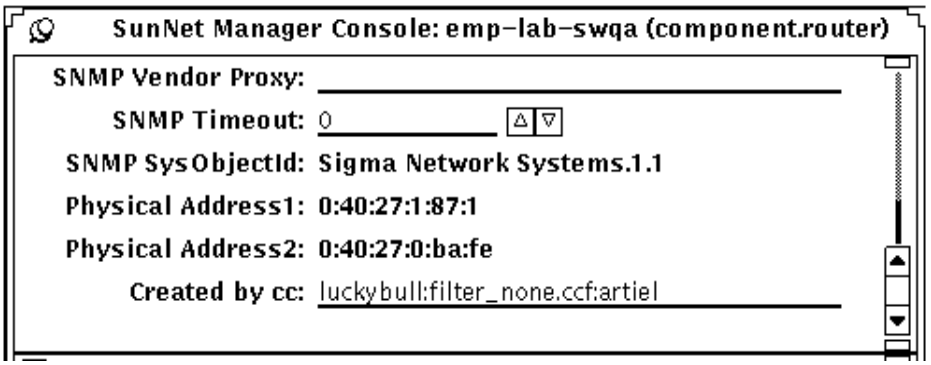

Figure 3-3 `Created by cc` Field in an Element's Properties Sheet

As topology traps are received by the Receiver process, the Receiver uses the database Application Programming Interface (API) of the local Domain Manager console to update the local database to reflect the topology information received from sending stations. The local Receiver is able to track the source of the elements that it adds to the local database due to the `Created by cc` field. This field is blank for all elements created by the local Site / SunNet / Domain Manager console.

Whenever the Receiver creates elements in the local database, the `Created by cc` field is filled in, as shown in Figure 3-3, to indicate the connection that was the source of that topology information. The `Created by cc` field matches the connection on hostname, filter table, and database name.

User-configurable filters are used at the sending station to determine what topology information should be forwarded. Forwarded information can include view memberships, agents and proxy system names on the element's "properties sheet," glyph color, screen position (X,Y coordinates), and attributes. See Chapter 4, "Using the Configuration Tool" for instructions on how to configure the types of information to be forwarded.

When the Receiver process receives a topology trap from a sending station, it reads the local SNM database to determine if this element already exists. If the element does not exist, it is added to the local database. If an element already exists in the local database, the Receiver process determines whether the forwarded features of the element match those already attributed to the element. If the element's characteristics in the local database already match the features reported in the trap, the trap is ignored. If the element's characteristics in the database differ from the forwarded characteristics, the element information in the local database is changed to match the information forwarded from the sending station.

Note – If an element forwarded from the sending station already exists in the local runtime database and the existing record does not match the information passed from the remote sending station, the Receiver *overwrites* the existing record with the information forwarded from the Sender process on the remote station. Only the view membership information in the existing record is retained. If you want to protect local database records from being overwritten, you should configure the Sender daemons to not forward database traps to the local Receiver for those elements.

3.2.1.1 Synchronization of Shared Topology Information

The filters on the sending station define the information to be shared with remote receiving stations. When the databases exactly match in this targeted information, they are said to be "synchronized." If a Receiver is down for a period of time, the database information on the sending machine can change

during that interval. “Synchronization” is the process of updating the topology information on the receiving station so the two stations have exactly the same picture of the segment of network information to be shared between them.

You can manually execute synchronization by clicking on the Synchronize button (see Figure 3-2) or by configuring the Receiver to request synchronization automatically at start-up or at scheduled times. The Receiver initiates synchronization by sending a synchronization request to a selected Sender. In response to a synchronization request from an authorized Receiver, the Sender reads through its local database and forwards to the Receiver all topology data that passes the specified filters.

Whenever the Receiver adds elements to the local database to reflect information passed by remote Senders, it indicates in the element’s `Created by cc` field (as shown in Figure 3-3) the hostname, filter table name, and database name that was the source for that element.

Whenever a synchronization is initiated for a selected sending station, the Receiver removes all the existing records in the database with a `Created by cc` field that matches the target host, filter, and database names. Replacement records are then added to the local database as new information is received from the sending station in response to the synchronization request. For more information about database synchronization, see Section 3.5, “Database Synchronization.”

3.2.1.2 Holding Area Views

By default, a “holding area” view is created at the receiving console for each connection to a remote sending station. Holding area views created by the Receiver are indicated in the Home view by the CC icon, as shown in Figure 3-4. The name of a “holding area” view has the following form:

`snm`:<*host-name*>:<*database-name*>

where <*host-name*> is the name of the sending station and <*database-name*> is the database name used by the Sender daemon at that station.

In Figure 3-4, CC holding area views for sending machines named `luckybull` and `nanak` are present in the Home view.

When information about an element’s view memberships is passed from a sending station, the Receiver adds the element to the specified views, if they already exist. The “holding area” is used to store elements whose view location

is unknown to the Receiver. This situation can occur if elements are created at the sending station between synchronizations. If the element is a member of <*viewname*>, where <*viewname*> is not a view that already exists on the receiving station, the Receiver places the new element in the “holding area” view that corresponds to that sending station.

Topology information passed to a receiving station during synchronization never goes into holding area views, however. During synchronization viewname information is always passed before element information, so the views required for the elements already exist in the database at the receiving station. Thus, the holding area views are typically empty during normal operation.

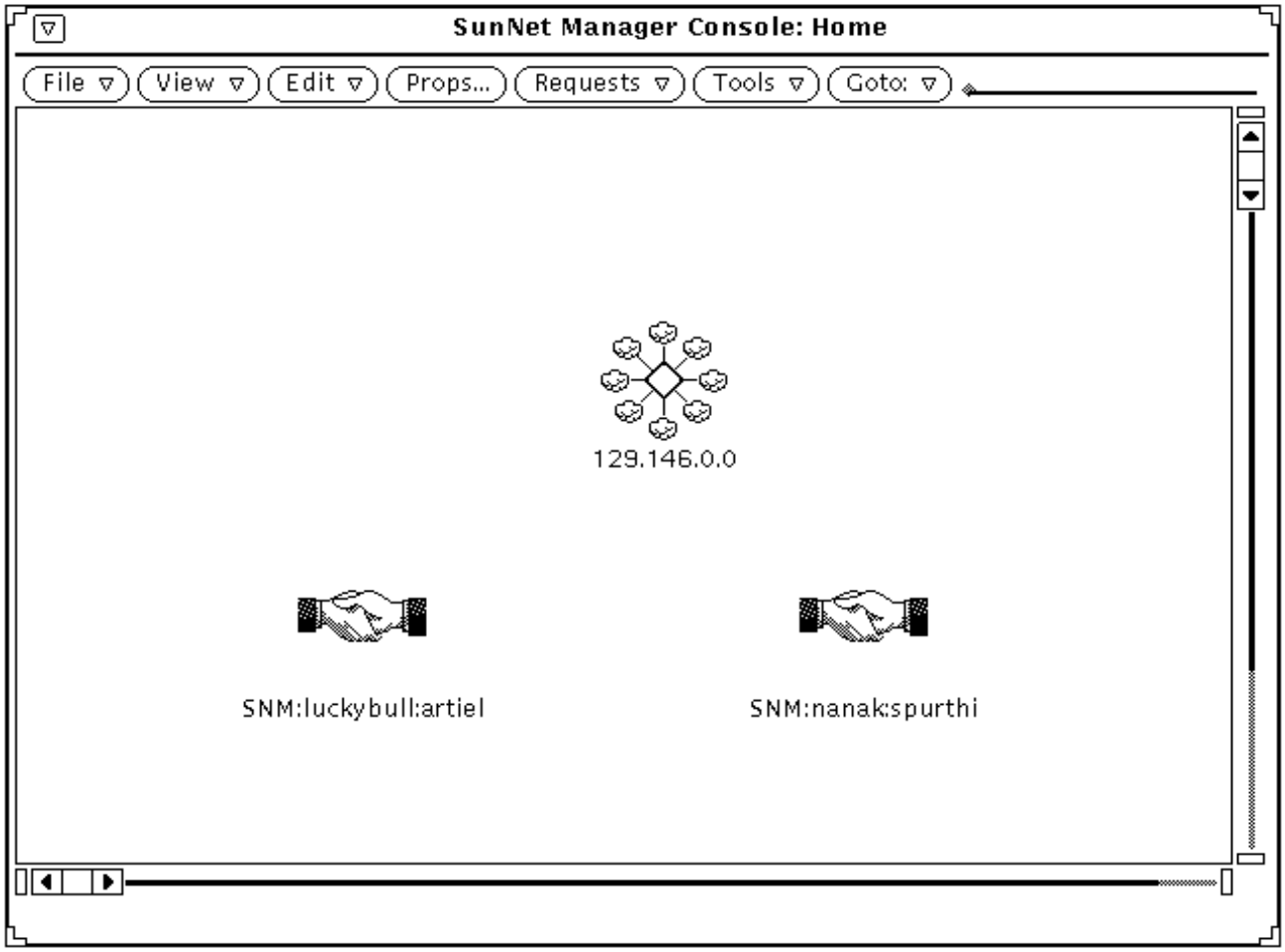

Figure 3-4 Holding Area Views in the Console Home View

3.2.1.3 *Using the Receiver* `-h` *Option*

If you want the Receiver to leave elements (such as components, views) that it adds to the local database in the “holding area” view, the Receiver’s `-h` use the runtime option to implement it. If you specify the `-h` option, any views and elements added to the local runtime database by the Receiver are held in the “holding area” view. This approach is useful if the local console is receiving forwarded topology information from a number of sending stations. The topology information forwarded from each machine is grouped under its respective holding area view in the console’s Home view.

The following two examples illustrate how the database from different sending stations are grouped in the holding area views using the Receiver `-h` option.

In Figure 3-5, two different views, `Spain` from the sending station `luckybull` with database `artie`, and `India` from the sending station `nanak` with database `raju`, are sent to the receiving station `charm2` with database `tylie`. When `charm2` receives information from the two sending stations, `luckybull` and `nanak`, it groups the topology data from the view `Spain` and `India` into the two respective holding area views in database `tylie`.

However, if both `luckybull` and `nanak` have the same view `Spain`, then both holding area views in database `tylie` on `charm2` will contain all the topology data associated with database `artie` and database `raju` as illustrated in Figure 3-6.

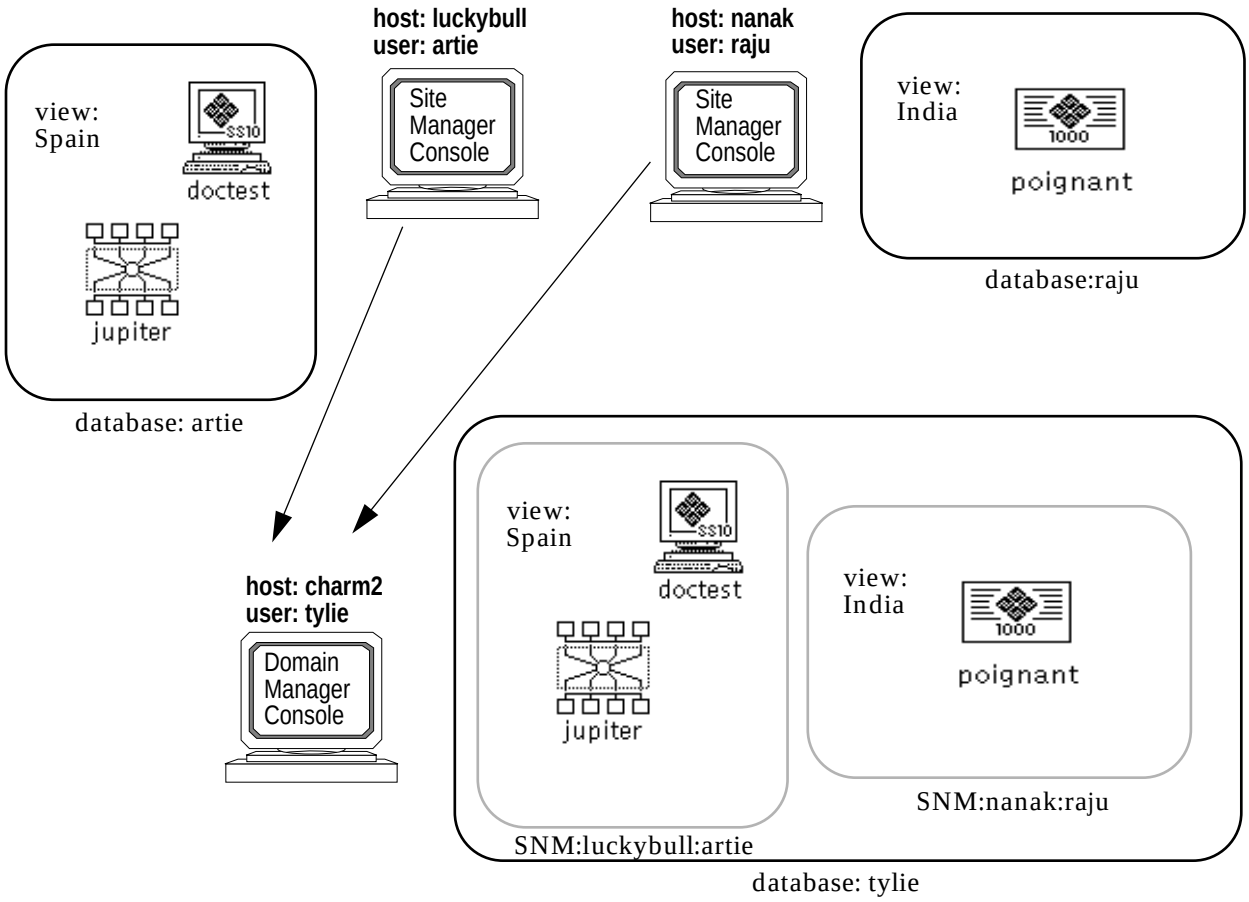

Figure 3-5 Multiple Holding Area Views

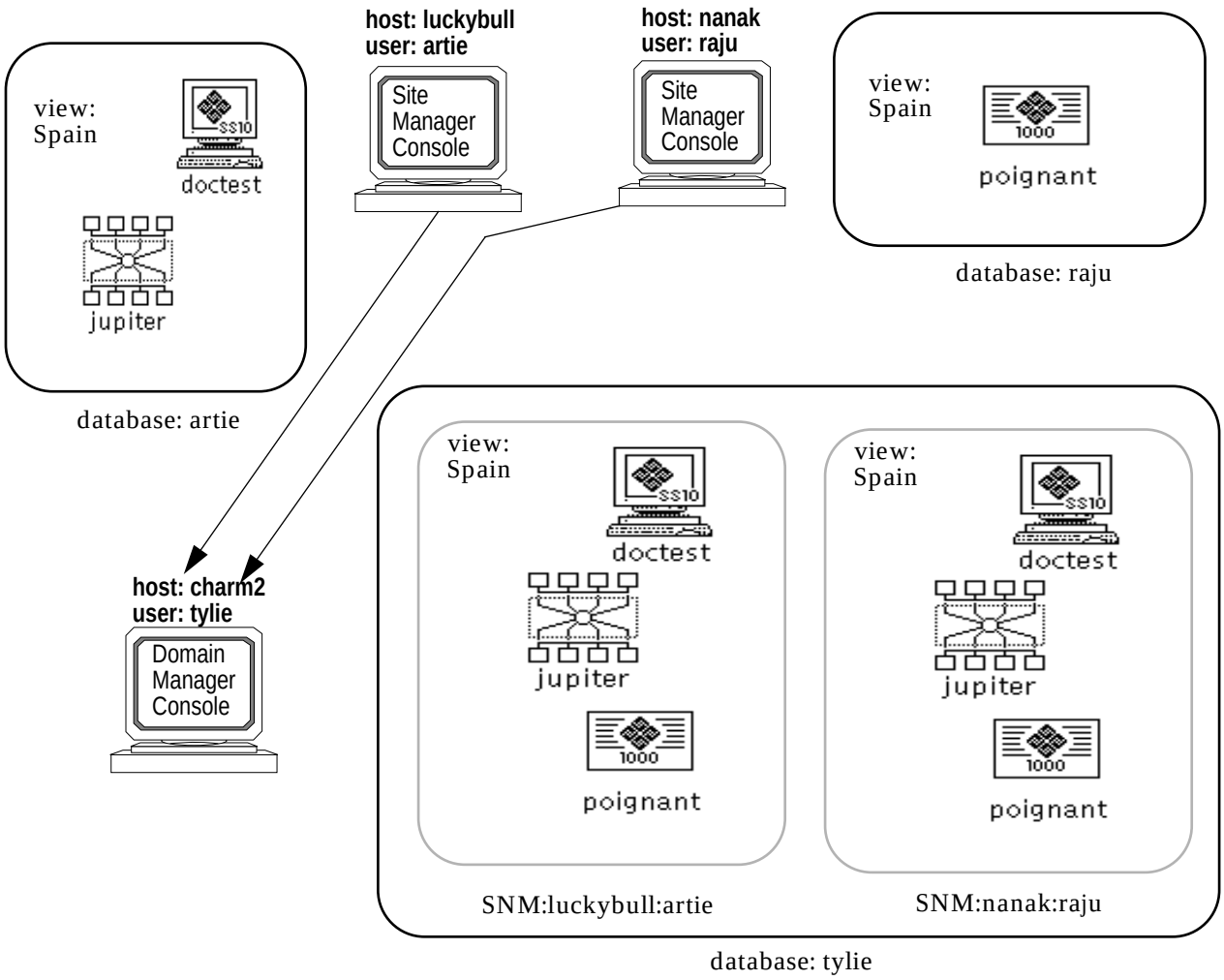

Figure 3-6 Multiple Holding Area Views containing the Same View

To implement the `-h` option, select Customize from the Domain Manager Console Tools menu, which invokes the Custom Tools window, as shown in Figure 3-7. Select the Cooperative Consoles Receiver and type in the `-h` runtime option, as shown below. Click SELECT on the Change button, then click SELECT on the Apply button to save the change.

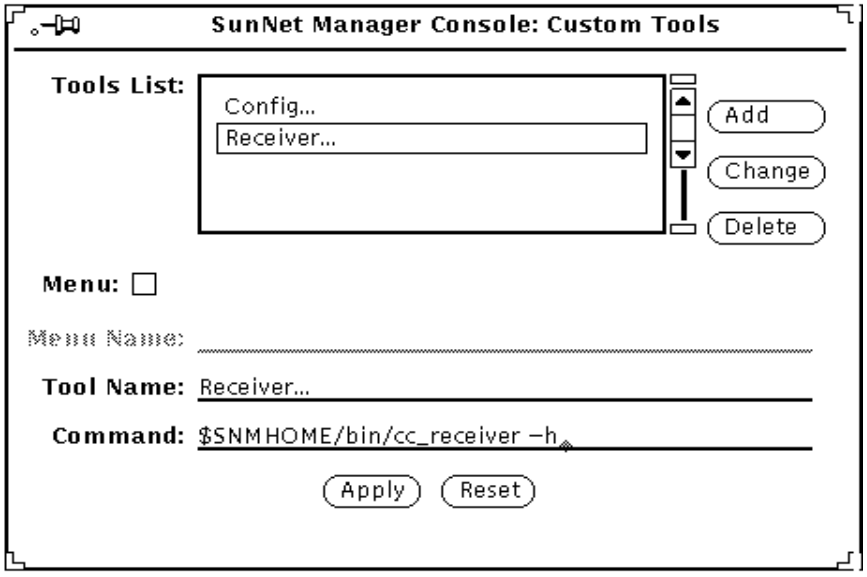

Figure 3-7 Adding the Receiver -h Option

Note – Using the -h option does not prevent the Receiver from overwriting element records if information for that same element is forwarded from a sending station and it does not match the existing record. The -h option provides a convenient way of grouping forwarded information by sending machine. If you want to prevent overwriting of certain element records in the local SNM database, you need to configure the Sender daemons to not forward topology information about those elements.

3.3 The Role of the SNM Event Dispatcher

A building block used by CC is the facility for forwarding events and traps, provided by the SNM Event Dispatcher (`na.event`). The Event Dispatcher permits network management applications to register with it to receive SNMP traps, SNM events, SNM database traps, and Glyph traps. Each instance of the Site/SunNet/Domain Manager console running on that machine is itself a network management application that registers with the Event Dispatcher to receive all traps and events. The Event Dispatcher spawns a child process for each network management application that registers with it.

When the Sender daemon on a given host is first launched, it registers with the Event Dispatcher on that machine in order to receive all SNMP traps, Glyph traps, SNM events, and SNM database traps. The Event Dispatcher forwards this information to the Sender daemon on the local machine for filtering and forwarding to receiving stations.

On each receiving station, the local Event Dispatcher receives SNMP traps, SNM events, and Glyph traps from remote Sender processes. The Event Dispatcher then forwards this information to the Domain Manager Console on the local machine. The Event Dispatcher on the receiving station does not receive SNM database traps forwarded from remote Sender daemons — topology information is sent to the Receiver process on the receiving station.

3.4 Sender Daemon Operation

The Sender process on a Site/SunNet/Domain Manager console machine is launched in response to the first registration request from a remote Receiver process. When the Sender process is first created, it registers with the local SNM Event Dispatcher to receive all SNMP traps, Glyph traps, SNM events, and SNM database (topology) traps.

A separate child Sender process is spawned for each receiving station that registers with the Sender. A Receiver process unregisters with the Sender if the user selects Quit from the Receiver window control menu on the receiving station. The Sender process unregisters with the Event Dispatcher and exits if every Receiver that had registered with it has unregistered.

3.4.1 Access Control

Access of remote receiving stations to local event and topology information is controlled by the Sender daemon. A remote Receiver process can successfully register with the Sender, and access a given local database, only if that receiving station is authorized to register with this Sender and access the specified database. To make this determination, the Sender daemon uses an authorization list that you define using the Configuration Tool on the sending station. This authorization list provides you with the ability to prevent unauthorized eavesdropping and discourage unnecessary forwarding of network event information.

The authorization list allows you to specify a list of authorized databases for each receiving station. See Section 4.2.1, “Setting Up the Authorization List” for further discussions on Authorization List.

3.4.2 Information Forwarding

As previously discussed, the Event Dispatcher on the receiving station receives SNMP traps, Glyph traps, and SNM events. All these events and traps are forwarded by the sender daemon.

The Sender daemon reformats all SNMP traps, Glyph traps, and SNM events into SNM traps before forwarding them to the Event Dispatcher on the receiving station. The Event Dispatcher on the receiving station passes the traps to the local Site/SunNet/Domain Manager Console. An SNM event is converted into a trap before being sent to a remote Domain Manager Console. This console ignores an SNM event that it cannot match to one of its own event requests.

3.4.2.1 Event-Forwarding Example

The example in Figure 3-8 shows an SNM event as viewed in the Event/Trap Reports window on the Site/SunNet/Domain Manager console running on the machine `montecarlo`. As indicated in the Console footer, the event has been generated by the SNM `diskinfo` agent on the router `emtv14a-41` in response to an event request “diskinfo.checkiffull” launched from this console machine. The condition that defined the occurrence of this event was any file system on `emtv14a-41` having more than 90% of its capacity used. The glyph representing `emtv14a-41` has dimmed to indicate an alarm has occurred.

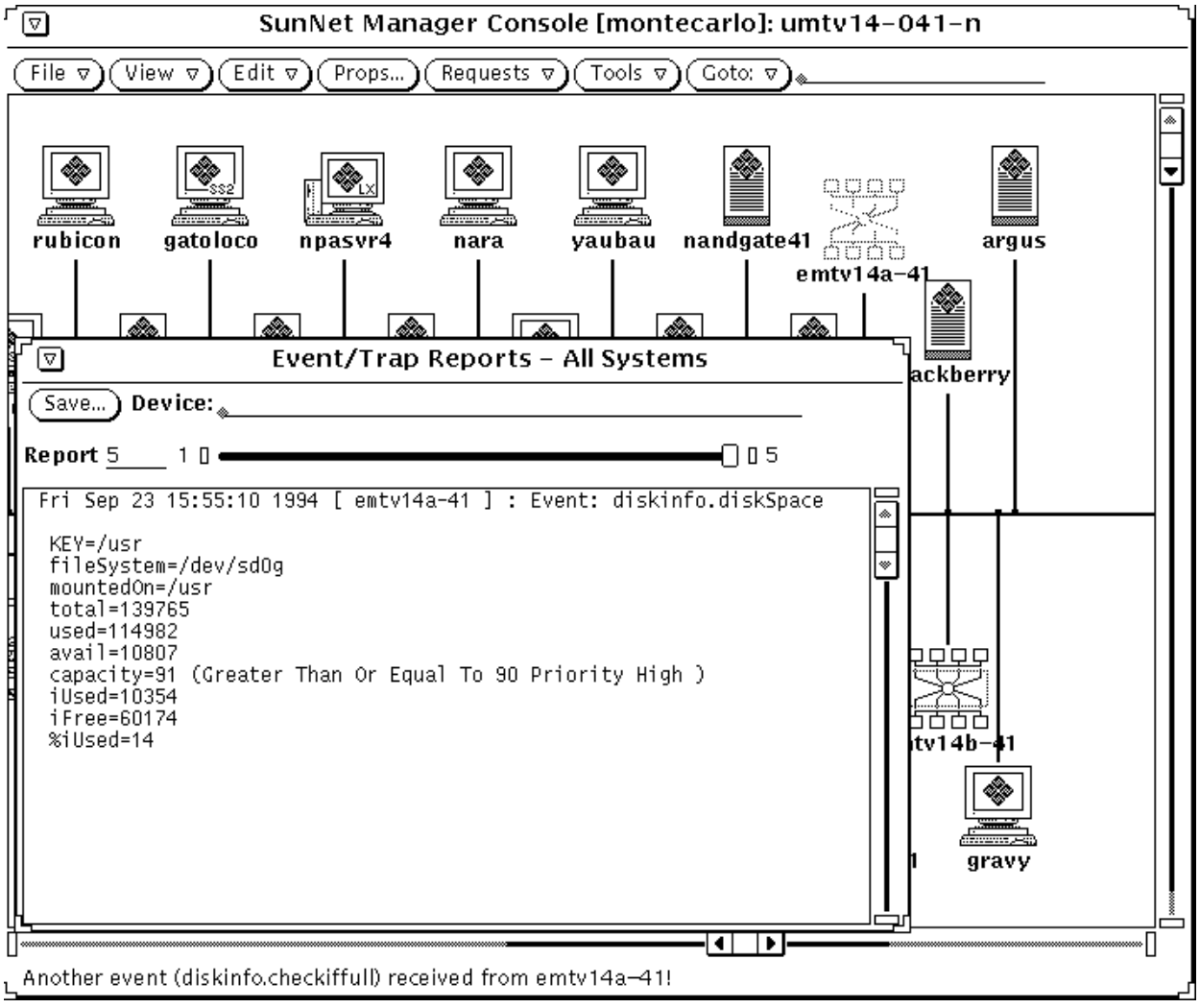

Figure 3-8 Example of Event as Viewed on Sending Station

The Sender daemon on `montecarlo` reformats this event as a trap, then forwards it to the appropriate receiving stations. Figure 3-9 shows this trap as viewed in the Event/Trap Reports window on the receiving station.

The line “coop_forwarded_by=montecarlo” in the Event/Trap Reports window in Figure 3-9 indicates that this trap is an event or trap that is forwarded by the Cooperative Consoles Sender process located on the Site/SunNet/Domain Manager console machine named `montecarlo`. Also, notice that the priority of the event has been changed by the Sender from high to low, as indicated in the last line in the Event/Trap Report window in Figure 3-9. This action is configured by the user when defining the Filter Table used by the Sender for forwarding events to this receiving station.

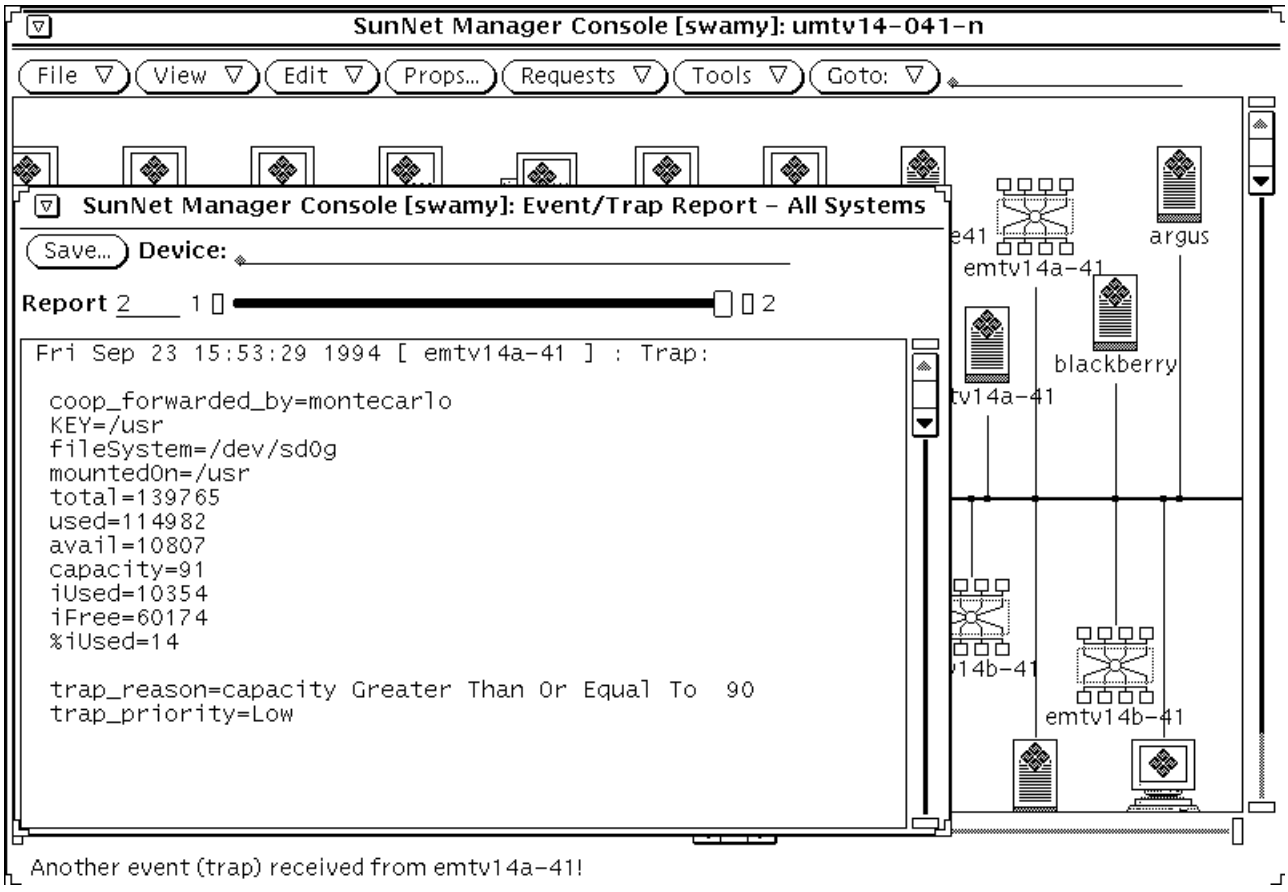

Figure 3-9 Event as Viewed on Receiving Station

3.4.2.2 Filtering Events and Traps for Forwarding

Note – Filtering is the most powerful and complex part of Cooperative Consoles. Please read this section thoroughly to understand the filtering process before you attempt to configure the filter files.

The Sender process uses one or more filter files in processing SNM events, SNMP traps, Glyph traps, and database traps for forwarding to the Event Dispatcher or Receiver Application on the receiving stations. The information that needed to be forwarded or dropped to the receiving station via database traps is specified in database template files. For information on the types of database traps and the information that can be forwarded, see Section 4.2.3.3, “Adding a Database Trap Type Entry.”

Multiple receiving stations can use the same filter file if the same criteria are to be used in processing events for forwarding, or separate filters can be defined for different receiving stations.

Each filter file (filter table) contains one or many filter entries that define the selection criteria (whether it’s a match) and action (whether to pass or drop) to be taken for a given SNM event, SNMP trap, Glyph trap, or database trap.

For events, SNMP traps, and Glyph traps, the selection criteria in each filter entry is based on the following:

- Filter type (for example, hostname, component, view name, view type, or default)
- Name (host name, component type, view name, or view type)
- Events or Traps type (for example, events, SNMP traps, or Glyph traps)
- Trap Selection Template (only if “traps” is selected and incoming traps are of SNMP trap type; SNMP traps can be further selected based on this template)

For database traps, the selection criteria in each filter entry is based on the following:

- Filter type
- Name

Any given trap or event that matches the selection criteria above is selected to pass or drop. Otherwise, the next filter entry (by precedence rule) is considered for that event or trap. The order of precedence is as follows:

- hostname
- component type
- view name
- view type
- default

When more than one filter entry is eligible for being a match, the first entry encountered is considered for the match. The filter entries are checked until a match is found or until the filter file is exhausted. Once a match is found, the action specified in the matching filter entry is performed after checking against the priority field and making the necessary priority changes. Any unmatched events or traps are dropped.

An event or trap that is considered a high priority event for one management station might have a lower priority for another management station. For this reason, you can also use the filters to change priorities of events or traps when forwarded from a sending station to a receiving station.

3.4.2.3 Forwarding of Topology Information

When the Sender daemon receives SNM database traps from the local Event Dispatcher, it uses the filter file specified for the target receiving station to determine which database template file to use in processing the database trap. The DB Template field contains the template file name.

The name of the element is the minimum information passed when a database trap is forwarded to the Receiver process on the receiving station. The database template file allows you to specify additional topology information that should be forwarded.

Because the filter file allows you to specify different DB Template files in each filter, you can use the filter selection criteria to specify different types of topology information to forward for different devices (by host name or element type), if you define different DB template files. Refer to Chapter 4, “Using the Configuration Tool” for information about configuring the filter table and the DB Template files.

3.5 Database Synchronization

The runtime databases on the sending and receiving stations are said to be "synchronized" when they are in agreement on the area of network information that they are intended to share. For example, if Consoles on machines RegionWest and RegionEast are both intended to contain information about routers on Net_A, then the RegionWest and RegionEast consoles are synchronized when they have the same information about routers in Net_A. Synchronization is therefore always in regard to some defined set of data that two or more runtime databases are intended to share. This information can include everything in the respective runtime databases or only some of that information. The intended area of shared data is determined by the filters on the pertinent sending stations.

Synchronization is an action initiated by the Receiver. When synchronization is initiated, the Receiver sends a synchronization request to the selected sending station on its Registration List, that is, the list of target stations from which the Receiver requests event and topology information.

When a Sender daemon receives a synchronization request from a receiving station on its list of authorized Receivers, the Sender reads through the database and sends all topology information in the DB Template files indicated for the target receiving station. As this information is received by the Receiver requesting the updated information, it is added to the local runtime database.

Whenever the Receiver adds elements to the local database to reflect information passed by remote Senders, it indicates in the element's `Created by cc` field (as shown in Figure 3-10) the hostname, filter table name, and database name that were the source for that element.

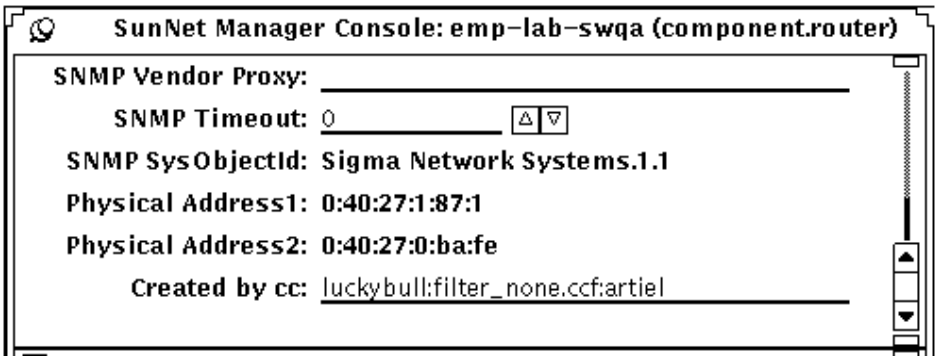

Figure 3-10 `Created by cc` Field in an Element's Properties Sheet

Whenever a synchronization is repeated for a target-sending station, already existing elements with a `Created by cc` attribute that matches the target hostname, filter, and database name are deleted and replaced with the updated information received from the sending station. In this way, the Receiver ensures that the local database reflects the latest information from the remote sending station. Because not all of the information in the local database may be shared information, the `Created by cc` attribute enables the Receiver to identify which portions of the local runtime database are shared with the remote sending stations.

3.5.1 Ways to Initiate Synchronization

A Receiver can initiate synchronization with its target sending stations in several ways:

- **Manually** — Select the target station on the Receiver's Registration List and click SELECT on the Synchronize button.
- **Automatically at Receiver start-up** — The Configuration Tool can configure a local Receiver to request synchronization with a target sending station on any occasion when the Receiver starts up. If the runtime database on the sending station has changed since the Receiver was last running, the receiving station has its database updated automatically at start-up to reflect the correct picture of the area of information shared between the sending and receiving station.

- **Automatically at scheduled intervals** — Use the Configuration Tool to schedule synchronization with a target sending station to occur automatically at a specified time of day on either a daily basis or weekly on a specified day of the week.

Refer to Chapter 4, “Using the Configuration Tool” for information about configuring the CC Receiver for automatic synchronization.

3.5.2 Multiple Sources of Information

Some scenarios include deriving an element in your local SNM database from a remote Site/SunNet/Domain Manager console on host A but not all of the attribute information for that element has come from host A. An example is illustrated by the periphery-to-center configuration shown in Figure 3-11. In this scenario, information about the device `doctest` can be received at Domain Manager Console `central` from either Console `site1` or Console `site2`.

In the situation depicted in Figure 3-11, the source of the instance `doctest` on `central` was `site1`, as indicated by the `Created by cc` field (shown beneath the elements in this example). In a case such as this, the source is determined chronologically — the first station to send information about an element is considered to be the source. But modifications to the properties sheet for `doctest` on `site2` is also reflected in the instance of `doctest` on `central`.

If the Receiver on `central` initiates a synchronization with `snm1`, the elements `jupiter` and `doctest` on central are deleted, then replaced by updated elements that match the current information on `site1`. If additional information about `doctest` has `site2` as its source, then it is necessary, with this configuration, to do a synchronization with `site2` to update this information as well. When information about specific elements has its source in multiple connections, it is necessary to initiate a synchronization with all of these connections to update information about that element.

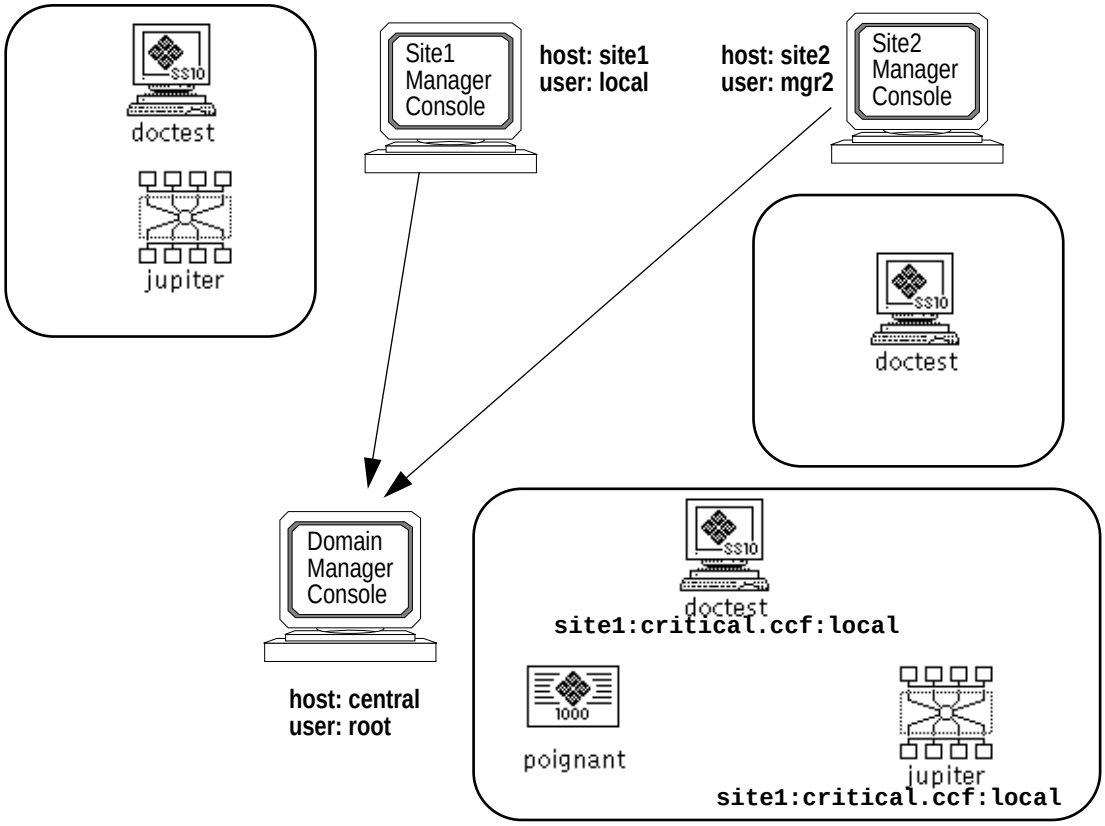

Figure 3-11 Information for Same Element from Multiple Sources

3.6 Delete Permission

The Receiver's Delete Permission option allows you to control the effect of deleting an element when multiple Site/SunNet/Domain Manager Consoles are linked by CC. The `Created by cc` field on each element in a local database indicates to the local Receiver process the *source* of that element. If the `Created by cc` field is blank, this indicates that the local Domain Manager console created that element — for example, by running Discover to build a view of its local network.

On the other hand, if the element was added to the local SNM runtime database by the Receiver, the `Created by cc` field indicates that the particular connection with a remote sending station was the source of that element. The source information in the `Created by cc` field is in the following form:

<hostname>:<filter-table>:<database-name>

In the sample configuration in Figure 3-12, the arrows indicate the direction of information flow from sending stations to receiving stations. The two elements `doctest` and `jupiter` were originally created on the station `domain1`. The `Created by cc` fields that result from this configuration are shown beneath each element.

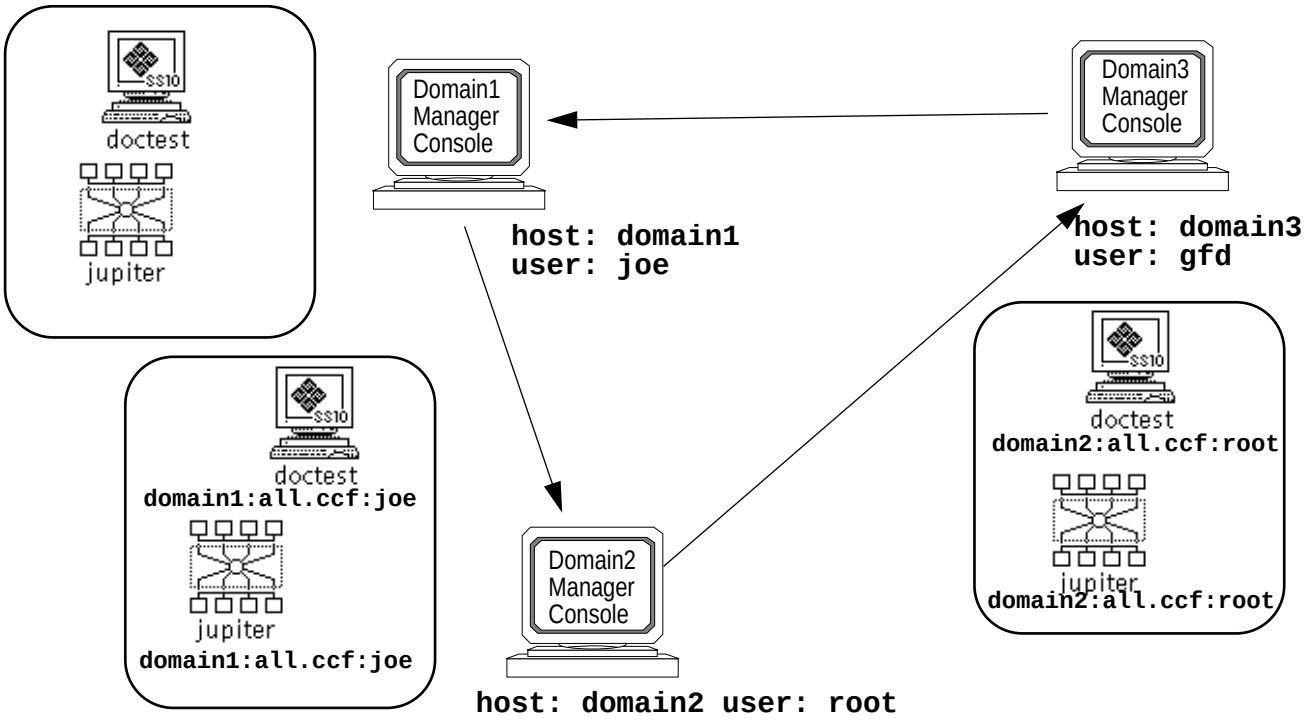

Figure 3-12 Sample Configuration Showing `Created by cc` Fields

When an element is deleted on a Site/SunNet/Domain Manager console, the Sender transmits a delete trap to registered Receivers if that element falls in the area of network information to be shared between them (as defined by the filters). If you have configured the Receiver's connection to that sending station with a Delete Permission of All, then the Receiver deletes the indicated element, no matter whether that remote sending station was the source of the element or not. If the Delete Permission for that connection is set to Source, however, then only elements that had that connection as their source could be deleted as the result of a deletion of an element on that remote station.

In the case of the configuration in Figure 3-12, if the Receiver on `domain1` has its connection with `domain3` set to a Delete Permission of All, and a user on `domain2` or `domain3` deletes the element, this results in its deletion on `domain1` as well. On the other hand, if domain1's connection to `domain3` has a Delete Permission set to Source, then the deletion of `jupiter` on either `domain2` or `domain3` would not result in the deletion of `jupiter` on `domain1`, since `domain1` is itself the source of that element on that console. Setting the Delete Permission to Source for a connection means that only an element whose `Created by cc` field matches that connection can be deleted as the result of a delete trap received from that connection.

3.7 Localizing Elements Received from Remote Stations

If Domain Manager console A has received an element from another Site/SunNet/Domain Manager console, B, then the `Created by cc` field for the instance of that element on Console B is filled in on the element's properties sheet to indicate that host A was the source for that information. For example, Figure 3-13 shows part of the properties sheet for a router element named "emp-lab-swqa." The `Created by cc` field on this properties sheet indicates that this element was received from the host `luckybull`.

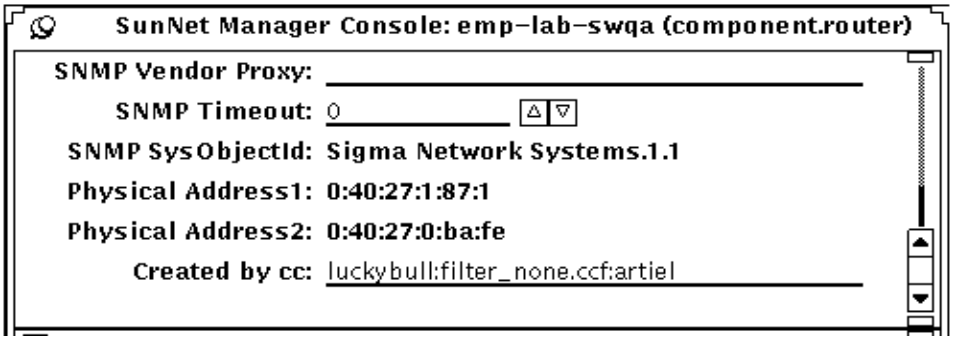

Figure 3-13 `Created by cc` Field in an Element's Properties Sheet

Elements created locally on B (for example, by running Discover), on the other hand, have a blank `Created by cc` field. An element with a blank `Created by cc` field is said to be *local* to that Console. Elements that are not local are deleted by the Receiver when a synchronization is initiated with the sending station that was the source for that element.

3.7.1 Localizing Selected Elements

Situations can arise where you do not want an element that was received by CC from a remote station to be deleted when a synchronization is performed. You can accomplish this by blanking out the `Created by cc` field in the properties sheet for that element.

To do this, select Properties... from the element's icon pulldown menu to invoke the element's properties sheet. Delete the information in the `Created by cc` field, then select Apply. An element that has had its `Created by cc` field blanked out this way is said to be "localized." You can localize selected elements manually, as just described. But the CC Receiver also provides a facility for doing a global localization of all elements received from a selected connection.

3.7.2 Global Localization

Clicking on the Localize button in the Receiver window clears the `Created by cc` field for all elements that had the selected connection as their source. The CC Receiver then regards these elements as having the local Domain Manager console as their source — as if they had been created locally by running Discover, for example.

The Localize button should be used only in certain special situations. If parts of the local network topology on your console have been built up by receipt of topology information from remote sending stations, in certain occasions you might not want this information to be wiped out by a new synchronization. You can localize the information received from selected stations so that a new synchronization with those stations does not erase that previously acquired topology data.

An example of such a situation is illustrated in Figure 3-14. Figure 3-14 shows two Domain Manager consoles arranged in a peer-to-peer configuration. (The peer-to-peer configuration is defined in Chapter 2, “Cooperative Consoles Configurations.”) The host `domainA` is the source of the database elements `jupiter` and `doctest` on `domainB`. If the database on `domainA` should be cleared (for example, by starting Domain Manager with the `-i` option), the database could be recovered if `domainA` were to synchronize with `domainB`. However, the database information that is passed from `domainB` to domainA now indicates `domainB` as its source, as shown in Figure 3-15.

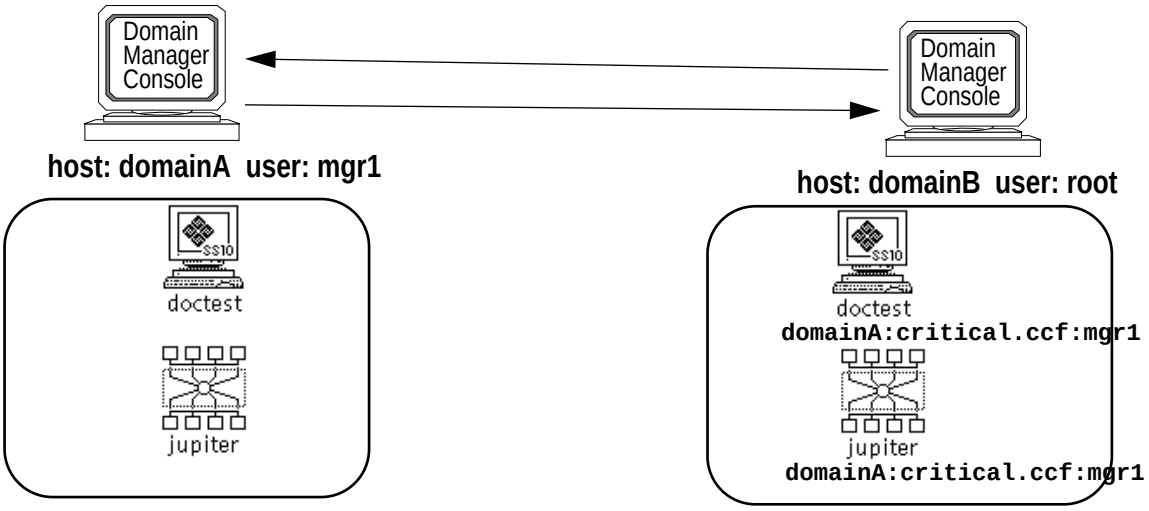

Figure 3-14 Peer-to-Peer Configuration Example Showing `Created by` `cc` Fields

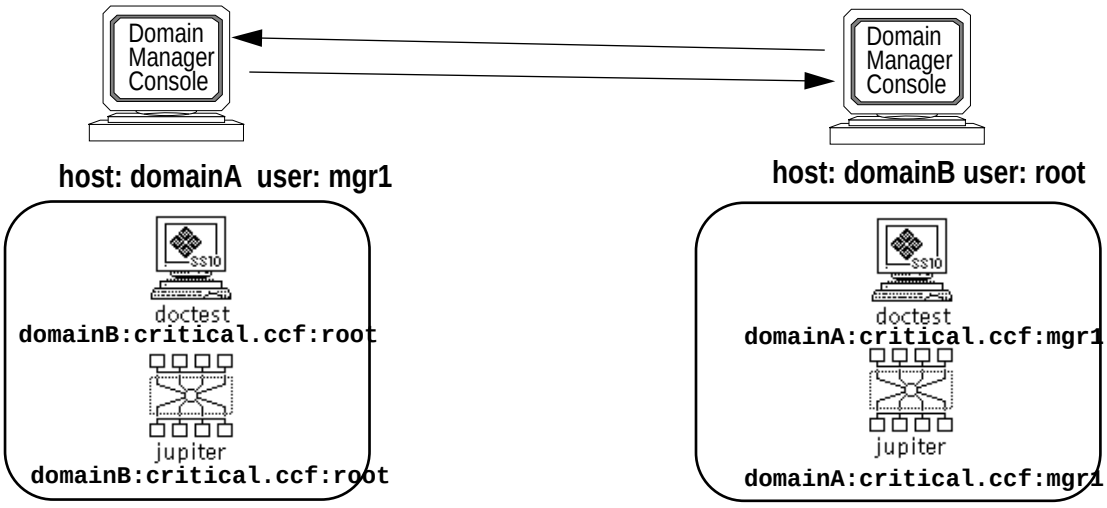

Figure 3-15 Peer-to-Peer Configuration with No Underived Elements

Note – In the situation depicted in Figure 3-15, the area of shared information is derivative on both machines — there is no underived database source for this information. In this situation it is recommended that you recreate a base or underived source for the data by localizing one of the copies of the information. Otherwise, a synchronization on either `domainA` or `domainB` for this connection will result in the data being deleted on both machines since there is no underived data source from which the information can be updated. Localizing the connection at the Receiver for `domainA` allows you to designate the local Console `domainA` as the source for the data, thus restoring the situation depicted in Figure 3-14.

3.8 Shutting Down Cooperative Consoles

The Receiver window provides the method for shutting down operation of CC. If you press the MENU button over the Receiver window control panel, the control panel popup menu appears. If you select Quit, the local Receiver process unregisters with the Sender processes at each of its target sending stations.

When all of the Receiver processes that have registered with the Sender process on a given Site/SunNet/Domain Manager machine unregister, that Sender process on that machine unregisters with the local Event Dispatcher and exits. Thus, all CC operations — Sender and Receiver processes — cease after Quit has been selected for the Receiver process at *each* receiving station.

3.9 Cooperative Consoles File Locations

The Cooperative Consoles executable binaries must be installed under the same path as the SNM executables ($SNMHOME). The default location for the executable binary files for CC is:

- `/usr/snm` in the Solaris 1.x environment
- `/opt/SUNWconn/snm` in the Solaris 2.x environment

These files can be relocated or installed on a server and shared across a network. The file structure for the Cooperative Consoles product is shown in Table 3-1. SNMHOME here refers to the path where Site/SunNet/Domain Manager has been installed.

Table 3-1 Cooperative Consoles File Structure

Directory	Files
`$SNMHOME/agents`	`cc_sender`
`$SNMHOME/bin`	`cc_config`, `cc_receiver`
`$SNMHOME/filter`	Sample filter files and DB template files
`$SNMHOME/help`	CC online help files
`$SNMHOME/icons`	Two files added for CC
`$SNMHOME/lib`	Dynamic library for `libcoop`
`$SNMHOME/man`	CC `man` pages
`$SNMHOME/struct`	`cooptools.schema`

The contents of the filter directory are copied to:

- `/var/adm/snm/cc_files` on Solaris 1.x environments
- `/var/opt/SUNWconn/snm/cc_files` on Solaris 2.x environment

`libcoop` is a directory for all libraries required to operate CC.

By convention, the following extensions are used. However, you can specify your own extensions.

- `.ccf` for Filter files
- `.cct` for DB template files
- `.cce` for Trap Selection template files

Using the Configuration Tool 4

The Cooperative Consoles Configuration Tool allows you to configure the CC Sender and Receiver on the local Site/SunNet/Domain Manager Console host. The Configuration Tool allows you to define the following:

- The list of remote stations authorized to register with the local Sender daemon and the databases they are authorized to access
- Filter files, database templates, and trap templates that determine the events, traps and topology information forwarded by the Sender daemon to remote receiving stations
- The list of remote management stations the local Receiver process attempts to register with and the database instance and filter file it requests at remote sending stations
- Schedule times for receiving stations to synchronize shared information with remote sending stations or configure a Receiver for automatic synchronization at start-up

You invoke the Configuration Tool from the Site/SunNet/Domain Manager Console's Tools Menu, as shown in Figure 4-1.

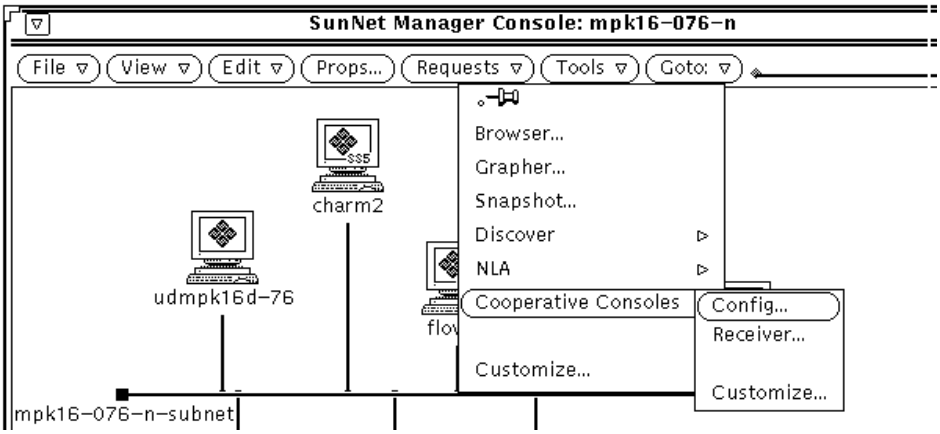

Figure 4-1 Selecting Configuration Tool from the Console Tools Menu

Invoking the Configuration Tool displays the main menu as shown in Figure 4-2.

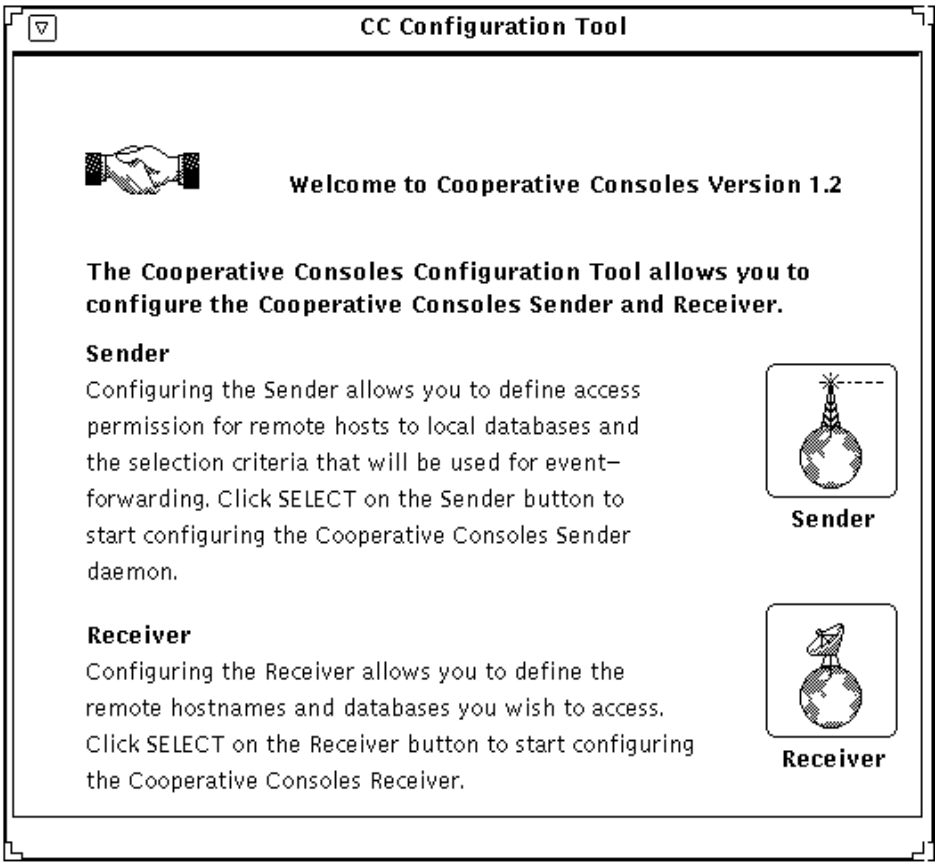

Figure 4-2 CC Configuration Tool Main Menu

To configure the action of the local Sender daemon, click SELECT on the Sender button. To configure the Receiver application on the local host, click SELECT on the Receiver button.

4

4.1 Configuring the Cooperative Consoles Receiver

If you select the Receiver button from the CC Configuration main menu, the Receiver Configuration window appears, as shown in Figure 4-3.

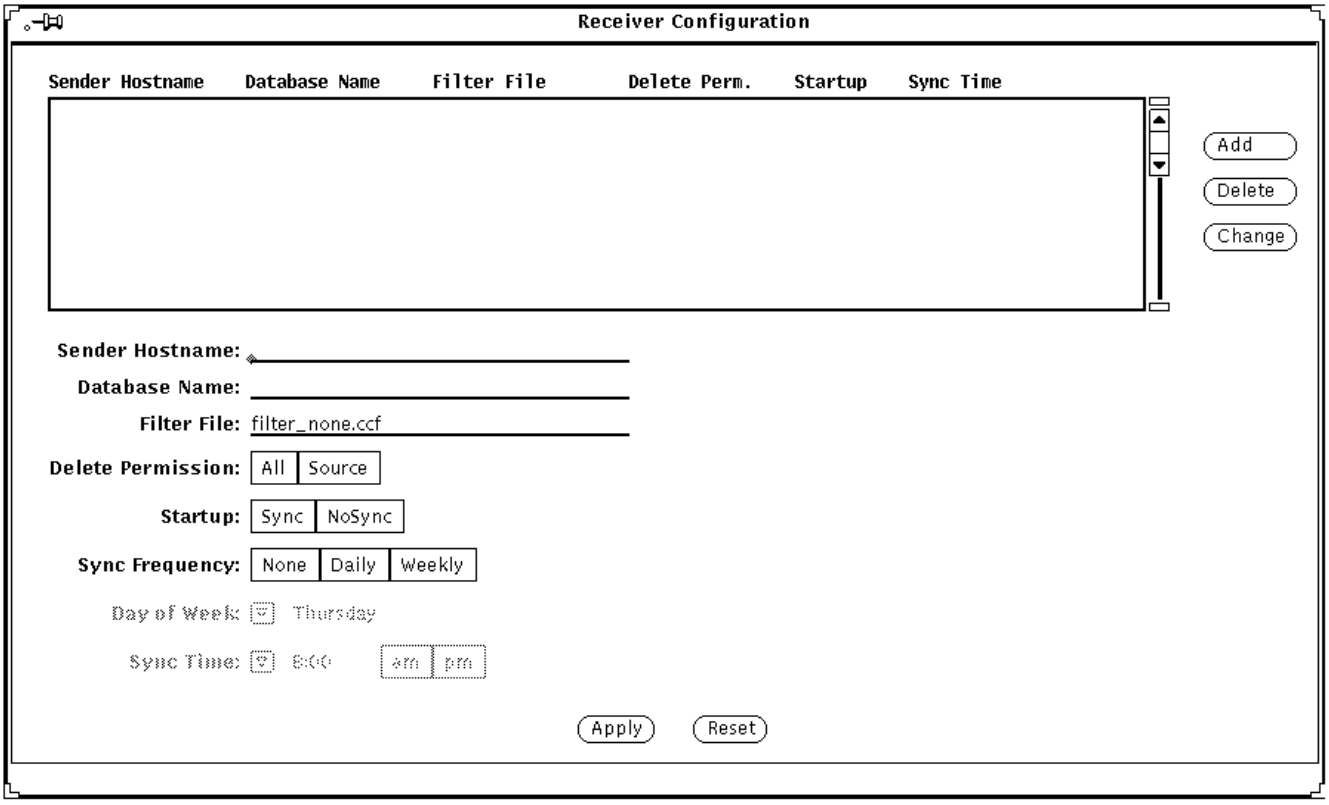

Figure 4-3 Receiver Configuration Window

The Receiver Configuration window allows you to build the registration list that determines the remote management stations that the local Receiver process will attempt to register with when it is launched. You can also specify the manner of automatic synchronization of the receiving station with remote sending stations.

4.1.1 Adding a New Entry to the Registration List

To add a new entry to the Registration List, you need to enter the appropriate information in the three fields at the bottom of the list window:

- **Sender Hostname** — This name is the host name of a remote Site/SunNet/Domain Manager console machine where a Sender daemon has been installed.
- **Database Name** — This name is the instance name of the runtime database on the remote machine that the Receiver requests access to when it registers with the Sender daemon on the remote machine. For example, if the database file is named `db.localnet`, then the name that should be entered in this field is **`localnet`**.
- **Filter File** — This field requests the filter file name on the sender station used to select event, trap, and topology information for forwarding. By default, `filter_none.ccf` is listed.
- **Delete Permission** — This toggle allows you to control deletion of elements in response to deletions of element instances on remote sending stations. If Delete Permission is set to Source, the Receiver deletes an element in response to a delete trap from a remote station only if that connection was the source of the local element instance (as indicated by the `Created by cc` field on the local element's properties sheet). If Delete Permission is set to All, then a delete trap received from a remote station results in the deletion of that element on the local Console even if that connection was not the source of the local instance of that element.
- **Startup** — This toggle determines whether the Receiver automatically executes a synchronization with the selected sending station whenever the Receiver is started.
- **Sync Frequency** — This option allows you to enable automatic synchronization of the selected connection at scheduled intervals. Automatic synchronization can be scheduled to occur either Daily or Weekly. If None is selected, scheduled synchronization is disabled.
- **Day of Week** — A pulldown menu enables you to select the day of week for scheduled automatic synchronization. This selection takes effect only if you have selected Weekly for Sync Frequency.
- **Sync Time** — A pulldown menu enables you to select the time of day for scheduled automatic synchronization. This selection takes effect only if you have selected Daily or Weekly for Sync Frequency.

After you have added this information, click SELECT on the Add button to add the new entry to the Registration List. A sample Registration List is shown in Figure 4-4.

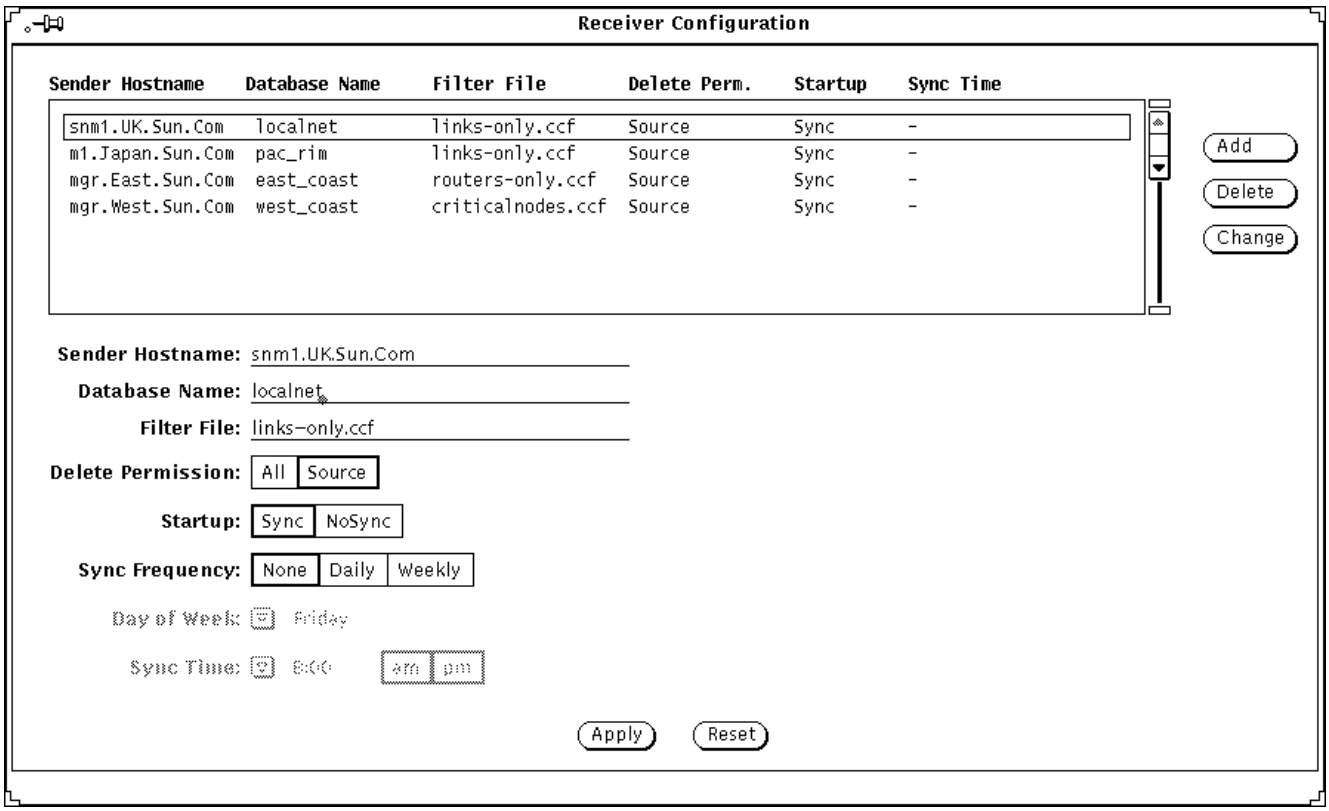

Figure 4-4 Sample Registration List

4.1.2 Modifying an Entry in the Registration List

If you want to change one of the existing entries in the Registration list, do the following:

1. Click SELECT on the entry that you want to change. The entry should now be highlighted.

2. The values for the three fields in the selected entry should now be displayed in the Sender Hostname, Database Name, and Filter File fields at the bottom of the screen. Edit these fields to make your desired changes.

3. Click SELECT on the Change button for your changes to take effect.

4.1.3 Deleting an Entry from the Registration List

To delete one of the entries from the Registration List, click SELECT on the entry to highlight it, then click SELECT on the Delete button.

4.1.4 Saving the Registration List

After you have finished adding or modifying entries in the Registration List, be sure to click SELECT on the Apply button to save your changes.

4.1.5 Abandoning Changes to the Registration List

If you want to abandon the changes you have made to the Registration List and revert to the previously saved version of the Registration List, click SELECT on the Reset button.

4.2 Configuring the Sender Daemon

If you select the Sender button from the CC Configuration main menu, the Sender Configuration window appears, as shown in Figure 4-5.

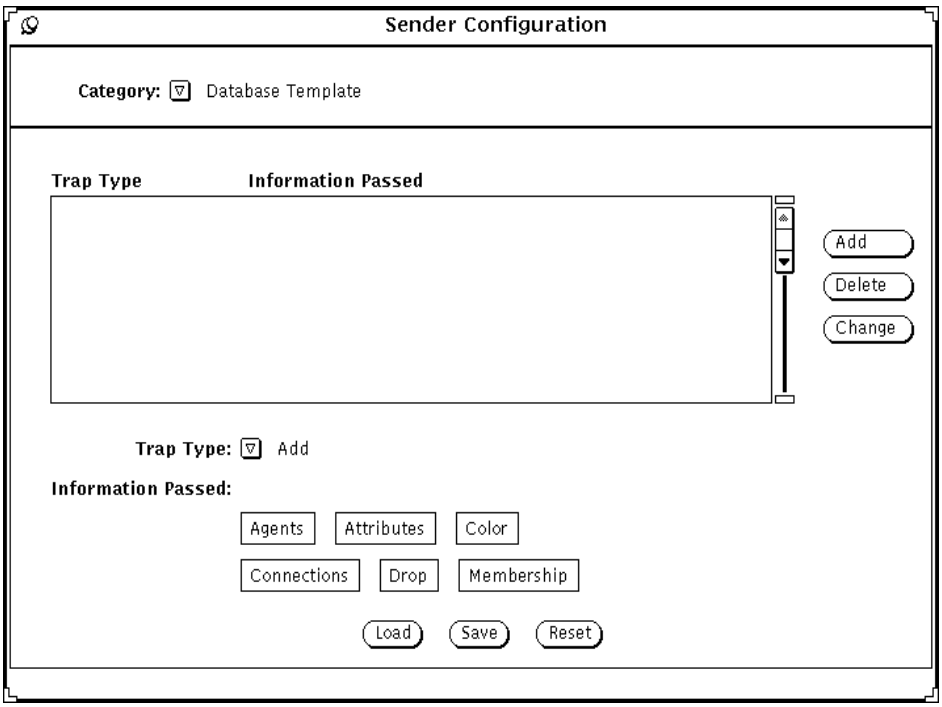

Figure 4-5 Sender Configuration Window

If you press the MENU button over the Category button, as shown in Figure 4-6, a menu appears with the following options:

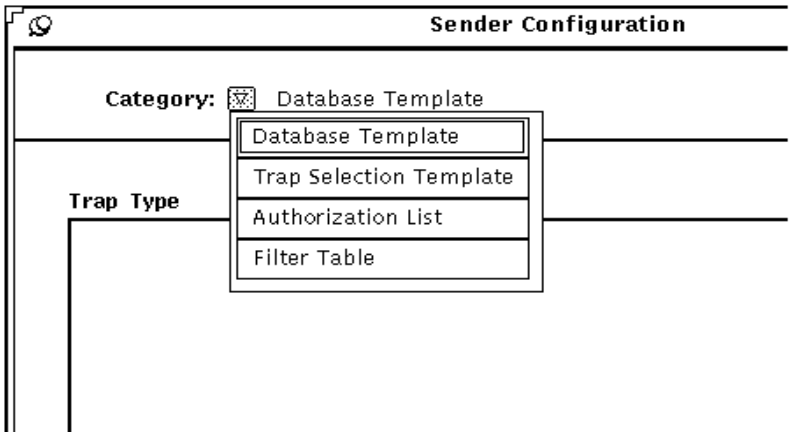

Figure 4-6 Sender Configuration Category Options

- **Database Template** — Select this Category if you want to define the topology information that the Sender daemon is to forward to specified Receiver processes. Multiple Database Templates can be defined**.**
- **Trap Selection Template** — Select this Category if you want to define the types of SNMP traps that the Sender daemon is to forward to specified Receiver processes. Multiple Trap Templates can be defined.
- **Authorization List** — Select this Category if you want to define the list of remote receiving stations authorized to register with the local Sender and the database instances they are authorized to access.
- **Filter Table** — Select this category if you want to define the selection criteria to use when forwarding SNMP traps, Glyph traps, SNM events, and database (topology) traps to receiving stations. Multiple filter tables can be defined.

4.2.1 Setting Up the Authorization List

Note – Refer to Section 4.2.3, “Specifying Topology Information for Forwarding” for information on how to set up the Database Template and Section 4.2.4, “Setting Up the Trap Selection Template” for Trap Selection Template discussions.

As shown in Figure 4-7, the local Sender’s authorization table contains two columns: Remote Receiver Host and Local Database Name.

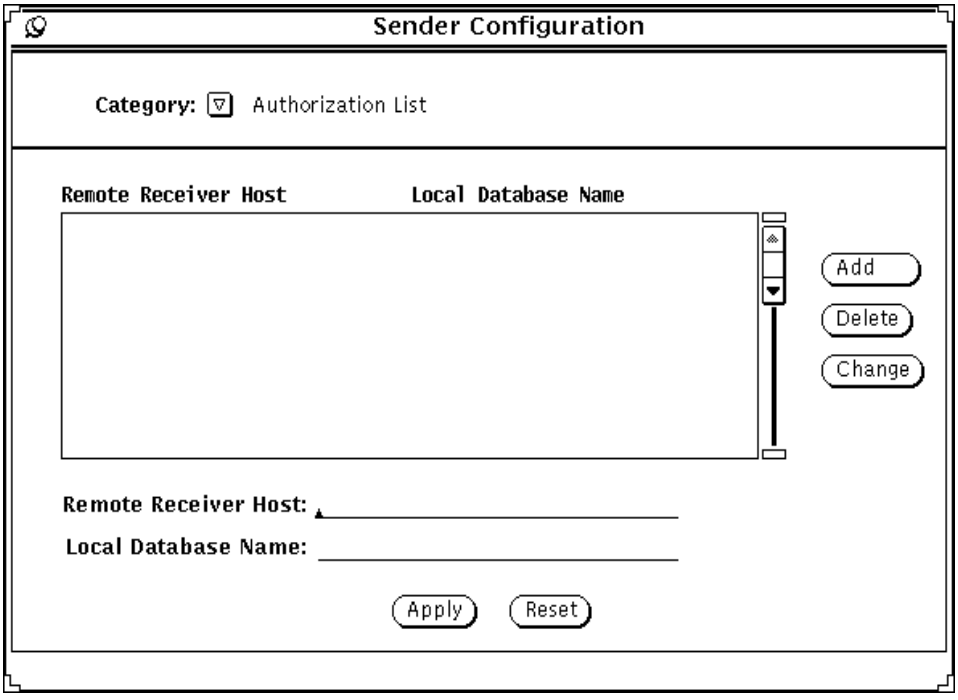

Figure 4-7 Authorization List Properties Sheet

- **Remote Receiver Host** — Names of Domain Manager host machines that are authorized to register with this Sender
- **Local Database Name** — A database name that the remote Domain Manager host is authorized to use in accessing the runtime management database on the local machine

Two fields at the bottom of the window enable you to create and modify rows in the Authorization List.

- Remote Receiver Host
- Local Database Name

4.2.1.1 *Adding a Remote Host to the Authorization List*

If you want to enter a new row in this list, type in the host name of the receiving station in the Remote Receiver Host field. Also type a database name in the Local Database Name field. For example, if you want to authorize a remote host to access a database named `db.localnet`, you enter **`localnet`** in the Local Database Name field.

When you have entered the information that you want to specify for the receiving station, click SELECT on the Add button to add the specified host to the authorization list.

You can make multiple entries in the authorization list for the same remote host if that host is authorized to access more than one database, as shown in Figure 4-8.

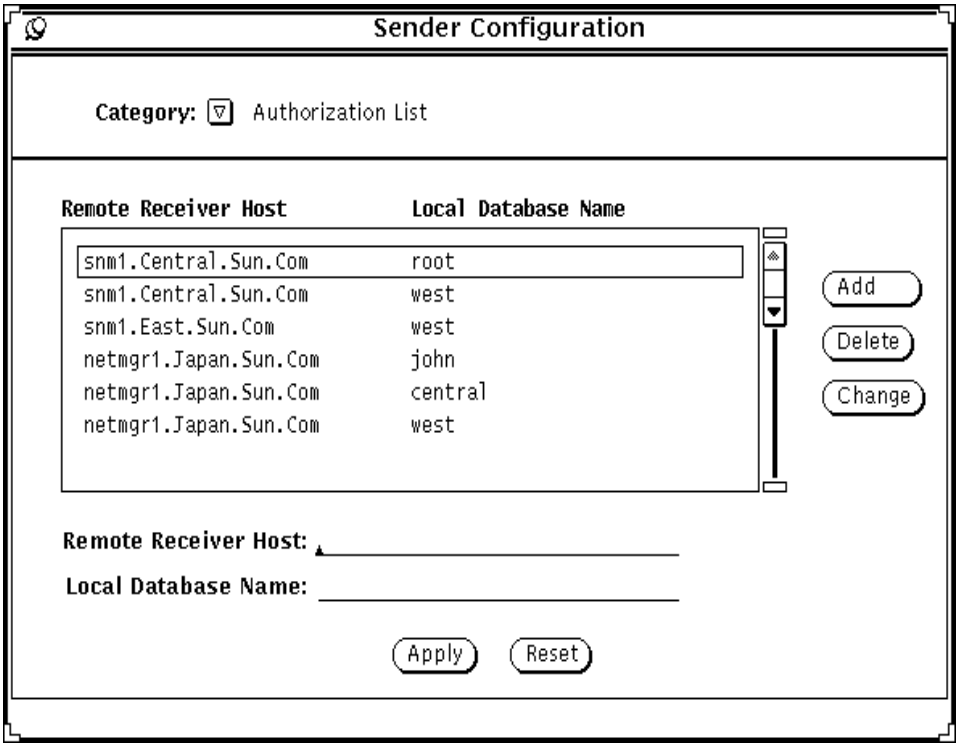

Figure 4-8 Sample Authorization List

For the example above, the machine `netmgr1.Japan.Sun.Com` has access to all three regions, `db.john`, `db.central`, and `db.west`.

4.2.1.2 *Changing an Entry in the Authorization List*

If you want to modify one of the entries (rows) in the authorization table, click SELECT on that row to highlight it. The information for that host is now displayed in the Remote Receiver Host and Local Database Name fields at the bottom of the window. After you have changed these values, click SELECT on the Change button to enter the changes into the table.

4.2.1.3 Removing a Remote Host from the Authorization List

If you want to delete one of the entries (rows) in the authorization list, click SELECT on that row to highlight it, then click SELECT on the Delete button.

4.2.1.4 Saving the Authorization List

After you have added new entries or made other changes in the Authorization List, click SELECT on the Apply button to save your changes.

4.2.1.5 Abandoning Changes to the Authorization List

If you want to abandon the changes you have made to the Authorization List and revert to the previously saved version, click SELECT on the Reset button.

4.2.2 Setting Up the Filter Table

The selection criteria used by the local Sender in forwarding SNMP traps, Glyph traps, SNM events, and topology information to registered receiving stations is contained in one or more filter files. The entries in each filter file constitute a single Filter Table. When you first select Filter Table from the Sender Configuration Category menu, the Filter Table properties sheet are blank, as shown in Figure 4-9.

Note – Filtering is the most powerful and complex part of Cooperative Consoles. Please read and understand Section 3.4.2.2, “Filtering Events and Traps for Forwarding” before you attempt to configure the Filter Table.

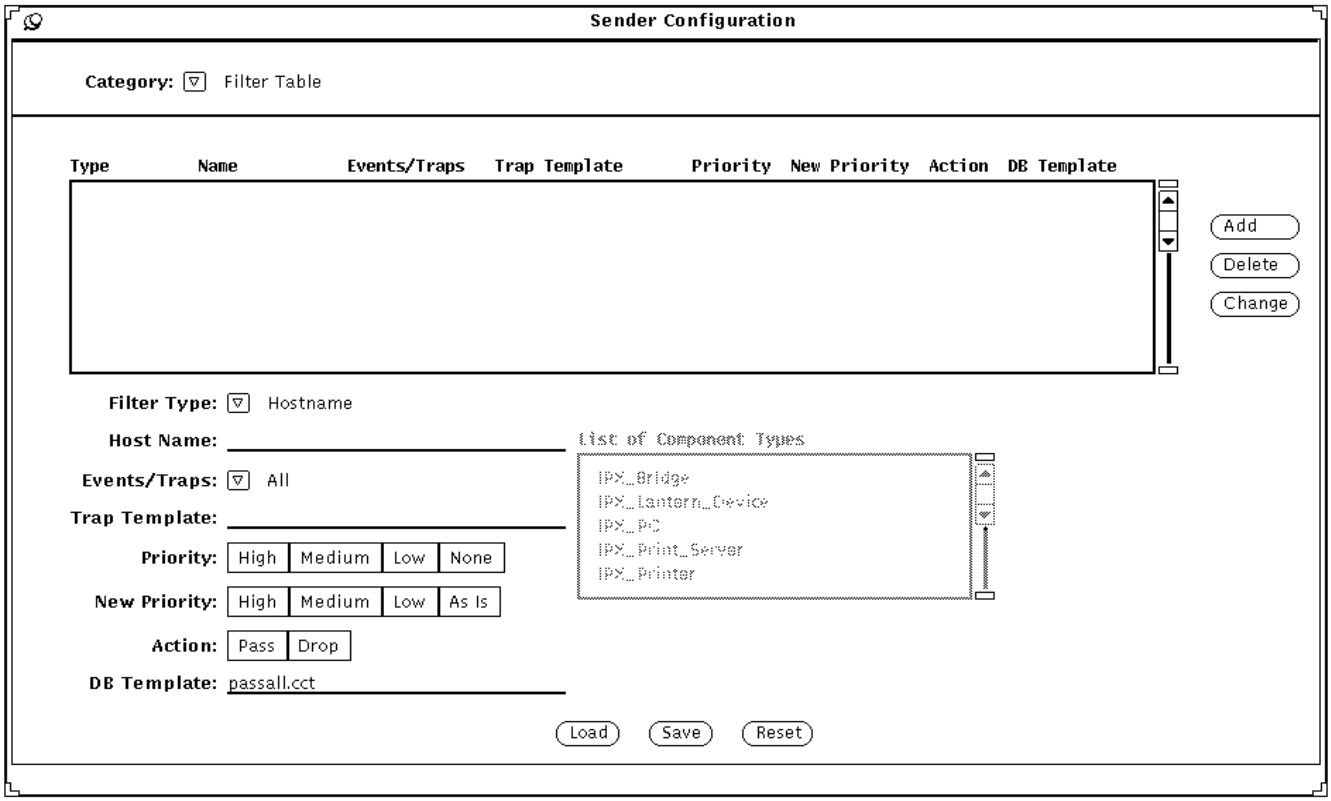

Figure 4-9 Sender Filter Table Window

4.2.2.1 *Loading a Filter*

You can either load an existing filter file or define a new filter table. To load an existing filter file, click SELECT on the Load button. You see a load window, as shown in Figure 4-10.

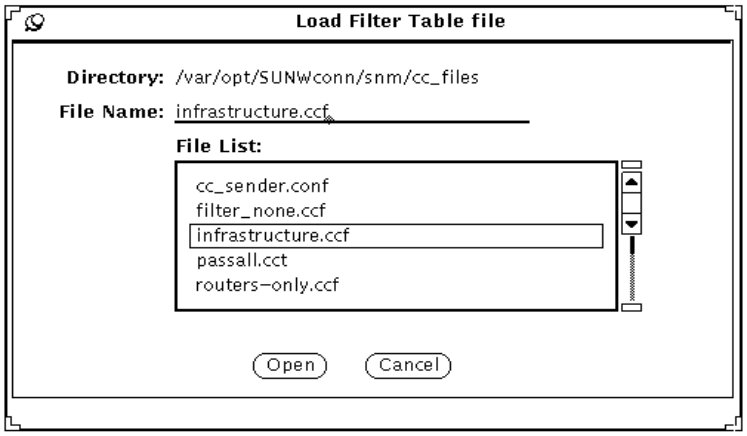

Figure 4-10 File Load Window

The CC product provides several sample filter files as listed in the **File List** scroll menu. For example, to load the `Infrastructure.ccf` filter file, select `Infrastructure.ccf` from the file list, then click SELECT on the Open button.

These sample filter files are located in the following directory:

- `/var/adm/snm/cc_files` on Solaris 1.x environments
- `/var/opt/SUNWconn/snm/cc_files` on Solaris 2.x environments

A sample Filter Table is shown in Figure 4-11.

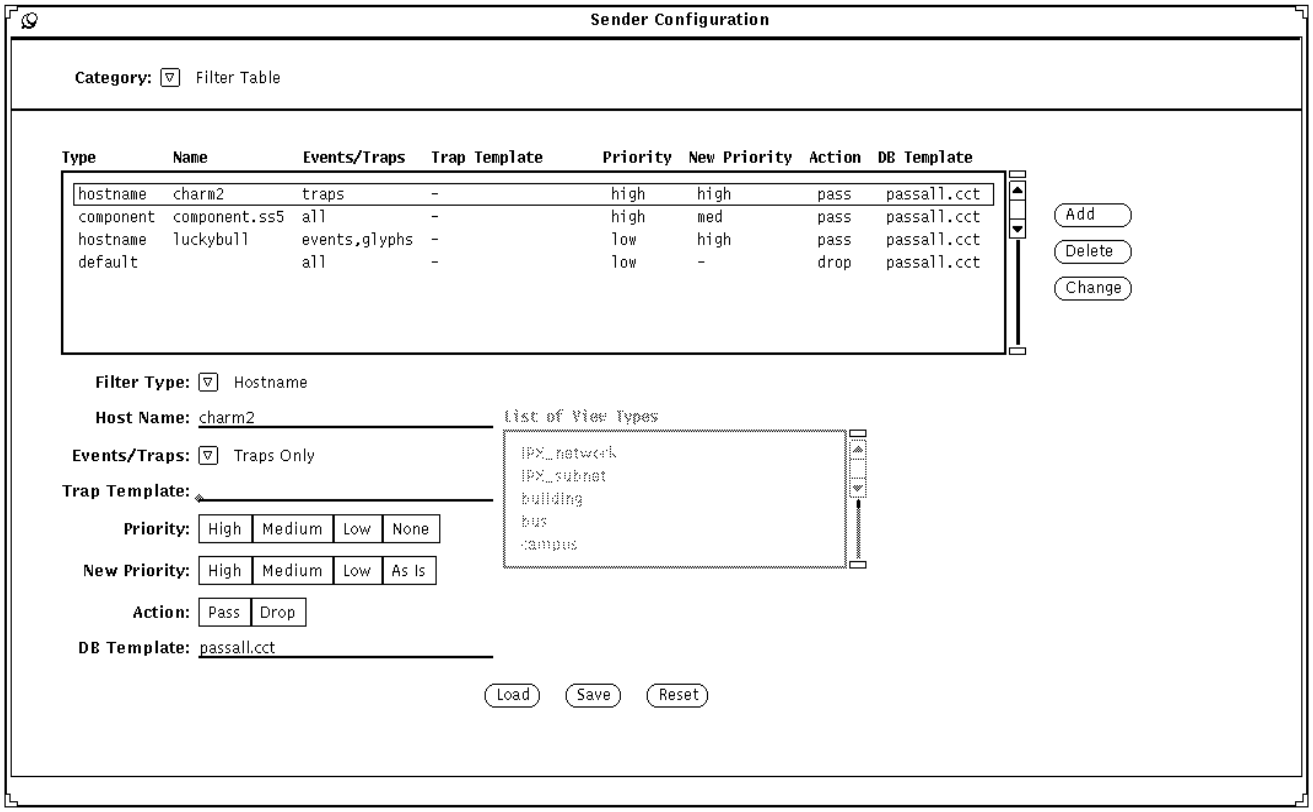

Figure 4-11 Sample Filter Table

Each row in the table is a filter that defines selection criteria and action to be taken if the events or traps match the selection criteria. Events and traps are checked against the filters in the table. See Section 4.2.2.2, “Adding a New Filter” for explanations of the different filter types.

4.2.2.2 *Adding a New Filter*

To add a new filter to a Filter Table, select the filter type. The filter types are listed if you press MENU on the Filter Type button, as shown in Figure 4-12.

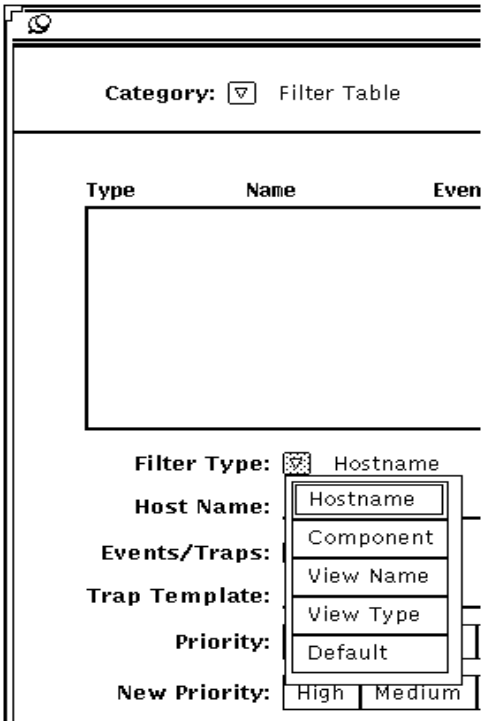

Figure 4-12 Filter Type Menu

- **Filter Type** — This selection determines the primary selection criterion for processing a trap or event. The following values are possible: *Hostname*, *Component*, *View Name*, *View Type*, or *Default*.
  - *Hostname* — When you select Hostname as the filter type, the filter selects events or traps on the basis of host name match. For example, you might want to use a Hostname type filter to select critical nodes by name in order to forward events or traps relating to those devices.
  - *Component* — When you select Component as the filter type, the component type name is used to select events and traps. For example, you might want to use a Component type filter to forward events or traps pertaining to all devices of a certain type such as `router`.
  - *View Name* — When you select View Name as the filter type, the filter selects events or traps on the basis of view name match.
  - *View Type* — When you select View Type as the filter type, the filter selects events or traps on the basis of view type match.
  - *Default* — If the event or trap is not selected by any of the filters in the table, the action specified by the Default filter is taken. For example, if only one hostname entry is present, and Default (Drop) is absent, then all information is passed. Note that a Filter Table should have one and only one Default entry.
- **Host Name** — This field changes depending on the filter type selected. The following values are possible: Host Name, Component, View Name, and View Type. The default setting is Host Name.
  - If the Type is *Hostname*, this field value must be either an IP address or the host name of a glyph in the SNM database. The user must enter in this field value.
  - If the Type is *Component*, a list of component types is provided. From this list, select a component type (for example, `component.router`). The selected component type is then displayed for this field value as shown in Figure 4-13.
  - If the Type is *View Name*, this field value must be a view name. For example, if the view type is named `mpk16(view.subnet)`, then the name to enter in this field is **`mpk16`**.
  - If the Type is *View Type*, a list of view types is provided. From this list, select a view type (for example, `building`, `bus`, and `campus` are all view types). The selected view type is then displayed for this field value as shown in Figure 4-14.

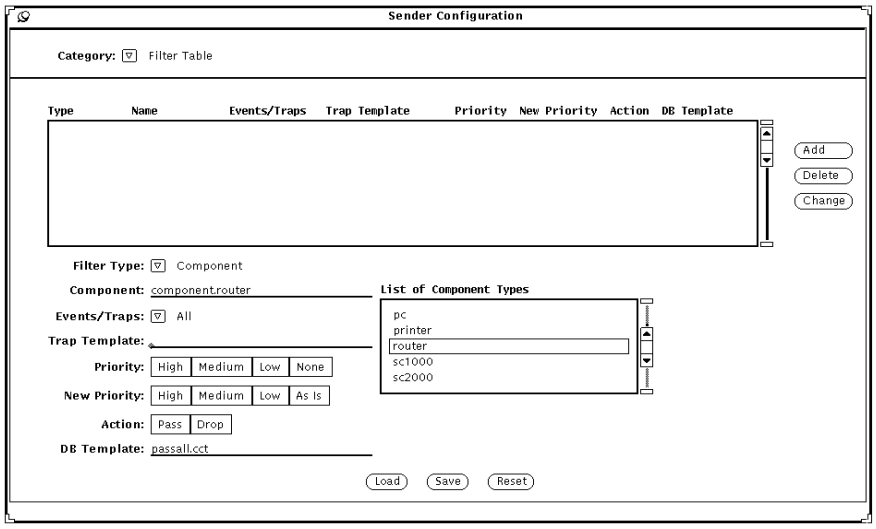

Figure 4-13 Component Type Menu

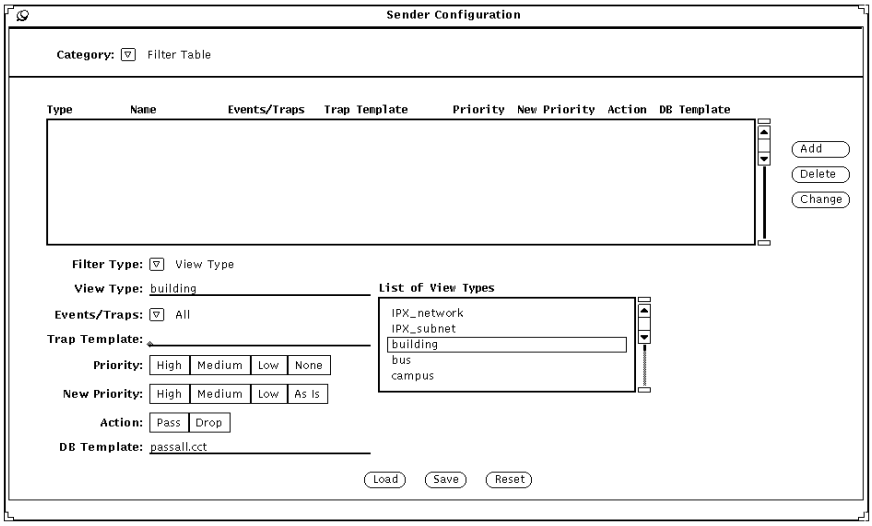

Figure 4-14 View Type Menu

- **Events/Traps** — The **Events/Traps** field allows you to further discriminate the filtering process. To select the options for this field, press MENU on the Events / Traps button, as shown in Figure 4-15.

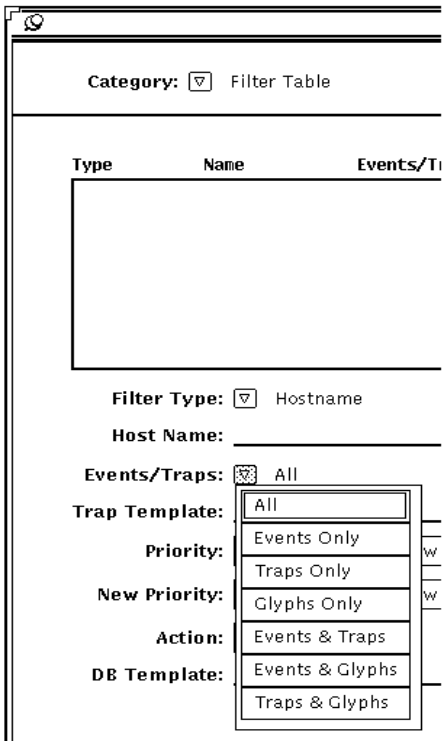

Figure 4-15 Events / Traps Menu

The following values are possible: *All*, *Events Only*, *Traps Only*, *Glyphs Only*, *Events & Traps*, *Events & Glyphs*, and *Traps & Glyphs*.

- *All*— Selects SNMP Traps, Glyph Traps, and SNM Events. Note that database (topology) traps are not included.
- *Events Only*— Selects the SNM Events only. SNM Events are generated as a result of the SNM event request from the Site / SunNet / Domain Manager console.
- *Traps Only*— Selects the SNMP Traps only. These traps can be Standard or Enterprise specific SNMP traps.

- *Glyphs Only*— Selects the Glyph Traps only. Glyph Traps are originated when the glyph state is changed manually on the Console. The glyph state can also be changed by the forwarding of SNMP Traps and SNM Events.
- *Events & Traps*— Selects SNM Events and SNMP Traps.
- *Events & Glyphs*— Selects SNM Events and Glyph Traps.
- *Traps & Glyphs*— Selects SNMP Traps and Glyph Traps.

- **Trap Template** — The **Trap Template** field specifies the file name to be used in selecting the types of SNMP traps and trap numbers for filtering. If this field is left blank, the trap filtering is based on the selection of the **Events/Traps** field only. The role of the Trap Template is discussed in Section 4.2.4, “Setting Up the Trap Selection Template.”

- **Priority** — The priority fields apply to SNMP Traps and SNM Events only. This field specifies the priority of the SNMP Traps and SNM Events for filtering. For example, if you select High for this field, then only high priority events and traps are selected for filtering. If you select Medium, then both medium and high priority events and traps are selected. If you select Low, then Low, Medium, and High priority events and traps are selected. For backward compatibility, the None button exists but it has the same functionality as the Low button.

  All SNMP Traps or SNM Events with a priority less than the priority selected in the **Priority** field are not selected. For example, if Medium is selected, then all traps and events with a lower priority than Medium are not selected.

- **New Priority** — If a value is specified here, this becomes the new priority of the forwarded trap or event. When traps or events are forwarded, their priority is changed to a value specified in this field. If you want to change all events selected by this filter to be forwarded as low priority traps, click SELECT on Low. If you select the As Is button, the priority of a selected trap or event when forwarded remain the same.

- **Action** — This field applies to SNMP Traps, Glyph Traps, and SNM Events only. The **Action** field specifies that the event or trap is to be forwarded or dropped. Click SELECT to highlight the appropriate action in the **Action** field.

- **DB Template** — This field applies to database traps only. The **DB Template** field specifies the file name used in selecting the topology information to be forwarded or dropped when SNM Database Traps match the filter's **Name** field. The role of the Database Template is discussed in Section 4.2.3, "Specifying Topology Information for Forwarding."

After you have made all your selections for a single filter, click SELECT on the Add button if you want your selections to define a new filter (row) in the table.

Refer to Chapter 5, "Cooperative Consoles Examples" for the different filter entries examples.

4.2.2.3 *Changing a Filter*

To change the values of an existing filter in the filter table, do the following:

1. Click SELECT on the filter you want to change in the table.
2. Enter or select the values in the Filter Type, Name, Events/Traps, Trap Template, Priority, New Priority, Action, and DB Template fields.
3. Click SELECT on the Change button to modify the values of the selected filter.

4.2.2.4 *Deleting a Filter*

To delete a filter in the Filter Table, click SELECT on the filter to be deleted and click SELECT on the Delete button.

4.2.2.5 *Saving the Filter File*

After you are finished adding new filters, deleting or modifying existing filters in the Filter Table, click SELECT on the Save button to save your changes to this filter file. For changes to existing filter files, select the filter file name from the file list scroll menu (for example, `infrastructure.ccf`) and click SELECT on the Save button, as shown in Figure 4-16.

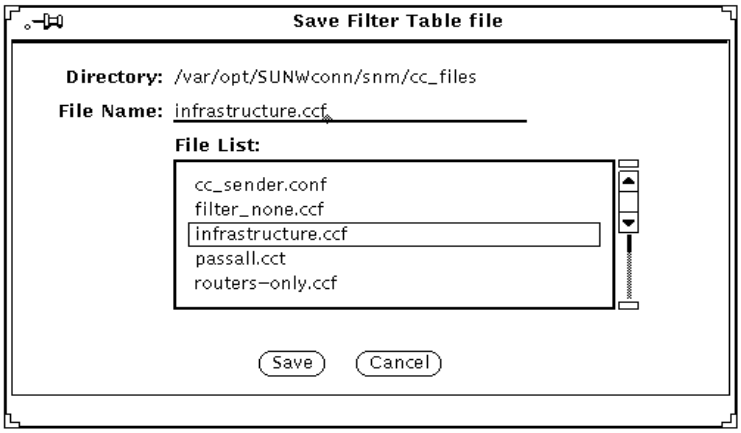

Figure 4-16 File Save Window

4.2.2.6 *Abandoning Changes to the Filter File*

If you want to abandon the changes you have made to the Filter Table and revert to the previously saved version of this filter file, click SELECT on the Reset button, as shown in Figure 4-11.

4.2.3 *Specifying Topology Information for Forwarding*

The Site / SunNet / Domain Manager Console generates database traps in response to changes in the runtime database. When an SNM database trap is generated, the Sender daemon uses a filter table specified by a remote Receiver process to determine whether the trap should be forwarded to that Receiver process. Within the filter table, a Database Template file is specified to determine the topology information to be forwarded or dropped.

The Database Template files used for selecting topology information to be forwarded / dropped depend on the filter file requested by the Receiver at the time it registered with the local Sender daemon.

Each filter entry in the filter file can specify a **DB Template** file to be used for elements that match the selection criteria of that filter. The element is selected by the criteria as mentioned under Section 3.4.2.2, “Filtering Events and Traps for Forwarding.”

Database Template files are only accessed in response to SNM database traps.

4.2.3.1 *Setting Up the Database Template*

To configure a Database Template file, select Database Template from the Sender Configuration Category menu. When initially selected, the Database Template window is blank, as shown in Figure 4-17.

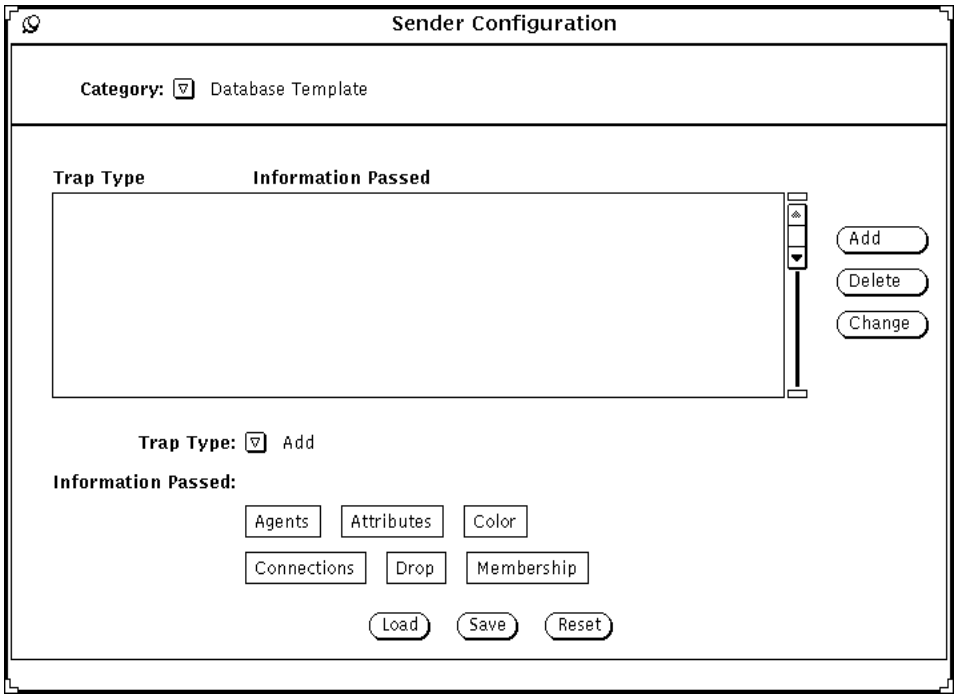

Figure 4-17 Database Template Window

4.2.3.2 *Loading a Database Template File*

You can either load an existing Database Template file or create a new Database Template. To load an existing Database Template file, click SELECT on the Load button. A Database Template Load window appears, as shown in Figure 4-18.

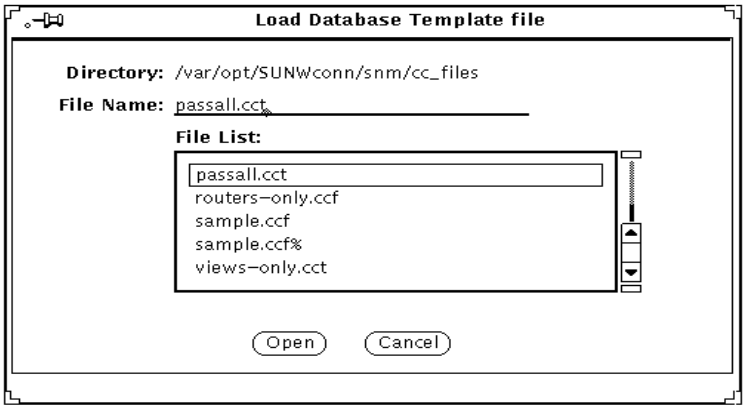

Figure 4-18 Database Template Load Window

The CC product provides several sample database template files as listed in the **File List** scroll menu. For example, to load the `passall.cct` database template file, select `passall.cct` from the file list, then click SELECT on the Open button.

These sample database template files are located in the following directories:

- `/var/adm/snm/cc_files` on Solaris 1.x environments
- `/var/opt/SUNWconn/snm/cc_files` on Solaris 2.x environments

The `passall.cct` sample Database Template is shown in Figure 4-19.

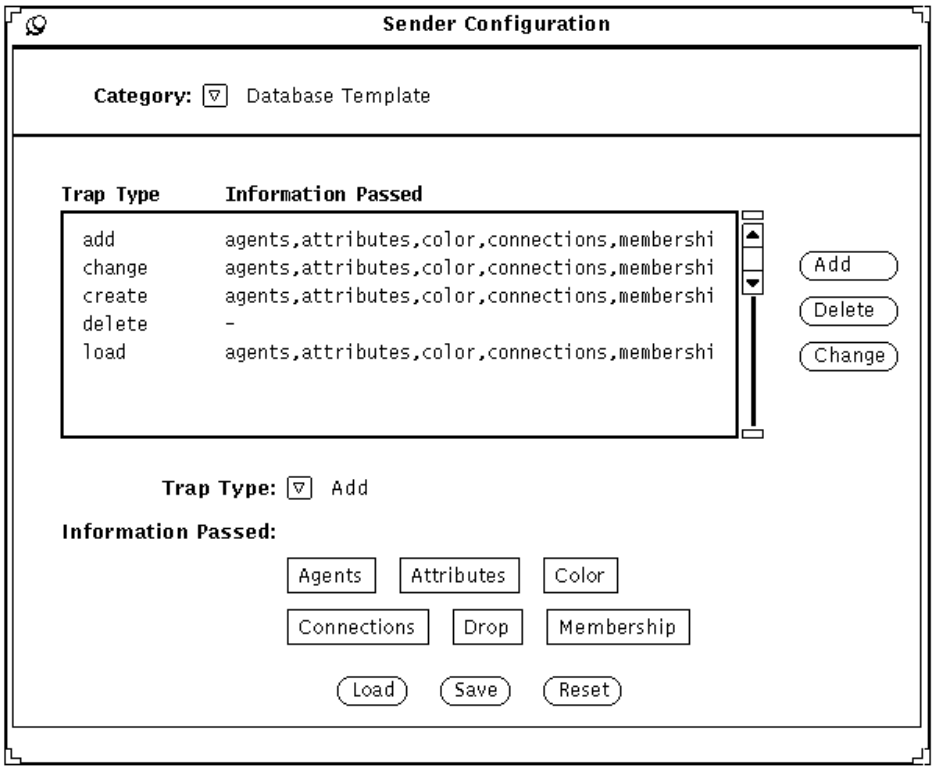

Figure 4-19 Sample Database Template

4.2.3.3 *Adding a Database Trap Type Entry*

To add an entry for one of the SNM database trap types, press MENU over the Trap Type button, as shown in Figure 4-20, and select one of the available trap types.

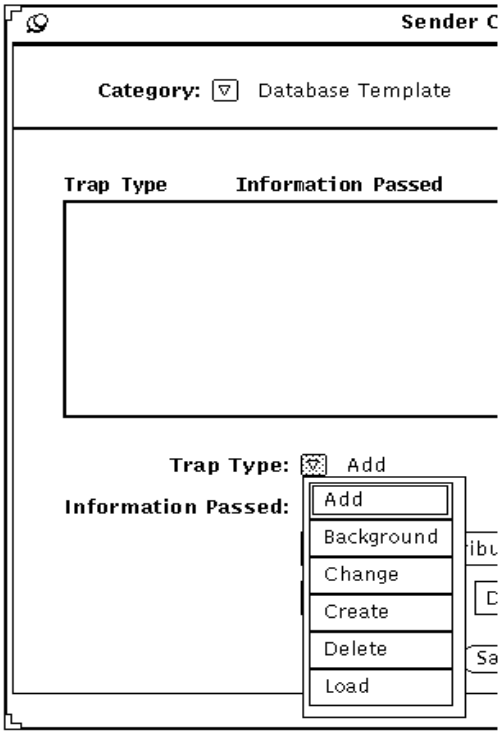

Figure 4-20 Trap Type Menu

A Database Template can contain six entries (rows). Each entry corresponds to one of the following types of SNM database trap:

- **Add** — Generated when a new element is added by the database Application Programmatic Interface.
- **Background** — Generated when a background image is added to a view.

Note – Due to the design limitation, **Background** is not supported at this release.

- **Change** — Generated when attributes of the element (such as the agents list or the screen coordinates) are changed.
- **Create** — Generated when a new element is created via the Site/SunNet/Domain Manager console.
- **Delete** — Generated when an element is deleted from the database.

Note – Due to the design limitation, the **Delete** trap cannot be dropped.

- **Load** — Generated when a new management database is loaded.

To specify the type of action to be taken for one of these SNM database trap types, an entry for that type must be in the template. If no entry is specified for a database trap type in the template, then that particular trap type is dropped.

The following table outlines the types of topology information that can be passed for each database trap type:

Table 4-1 Database Template Format

Trap Type	Keywords (one or more can be specified)
Add	membership, color, agents, attributes, connections, drop
Background	drop
Change	membership, color, agents, attributes, connections, drop
Create	membership, color, agents, attributes, connections, drop
Delete	drop
Load	membership, color, agents, attributes, connections, drop

The keywords used determine the forwarded content for each of the available trap types. The keywords have the following interpretation:

- **Agents** — If selected, a list of the agents checked off on the properties sheet is forwarded, along with the proxy system name for proxy agents.
- **Attributes** — If selected, the attributes of the element (such as, IP address or contact name) as defined in the element's schema are forwarded.

- **Color** — If selected, the element's RGB values are forwarded.
- **Connections** — If selected, information about simple connections between the selected element and other elements is forwarded.
- **Drop** — Select this button if you want the selected database trap type to be ignored (not forwarded). If this button is selected, the other buttons are ignored.
- **Membership** — If selected, the Viewname of each view the element belongs to is forwarded as is the screen position (coordinates) of the element within that view.

Each of these buttons is a toggle — click SELECT on the button to turn it off or on. The button is on when it is highlighted.

After you have selected the desired combination of values for these six toggles, click SELECT on the Add button to add an entry for this trap type to the template.

If no information (for example, Color) is selected for a trap type in the database template, then the information (in this case, color) is not passed.

Note – Agents, Attributes, Color, and Connections buttons have no effect for database traps of types Background or Delete.

4.2.3.4 Changing the Forwarding Specified for a Trap Type

If you want to modify the topology information specified for a trap type in the Database Template, do the following:

1. **Click SELECT on the trap type entry you want to change.**

   The six buttons along the bottom of the window should be highlighted to indicate the selected values for that trap type.

2. **Make the desired changes in the forwarding action for this trap type.**

   Click SELECT on the Agents, Attributes, Color, Connections, Drop, or Membership buttons.

3. **Click SELECT on the Change button to apply your selections to the Database Template entry.**

4.2.3.5 *Deleting a Database Trap Type Entry*

You can delete the existing entry in the Database Template for a particular trap type if you click SELECT on that entry, then click SELECT on the Delete button.

4.2.3.6 *Saving the Database Template*

After you have added new trap type entries, or deleted or modified existing trap type entries in the Database Template, click SELECT on the Save button to save your changes to this Database Template file. For changes to existing database template files, select the database template file name from the file list scroll menu (for example, `views-only.cct`) and click SELECT on the Save button, as shown in Figure 4-21.

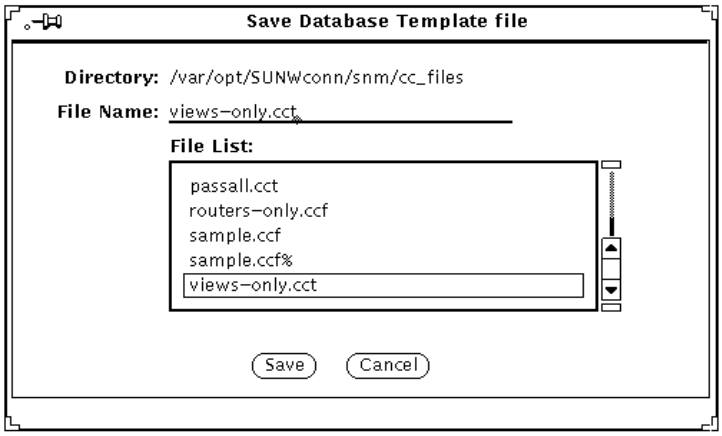

Figure 4-21 Database Template Save Window

4.2.3.7 *Abandoning Changes to the Database Template*

If you want to abandon the changes you have made to the Database Template and revert to the previously saved version of this Database Template file, click SELECT on the Reset button.

4.2.4 Setting Up the Trap Selection Template

To configure a Trap Selection Template file, select Trap Selection Template from the Sender Configuration Category menu. This template allows the user to filter only SNMP traps based on the trap type (Standard / Enterprise) and trap numbers. When initially selected, the Trap Selection Template window displays the available Enterprise traps names, as shown in Figure 4-22.

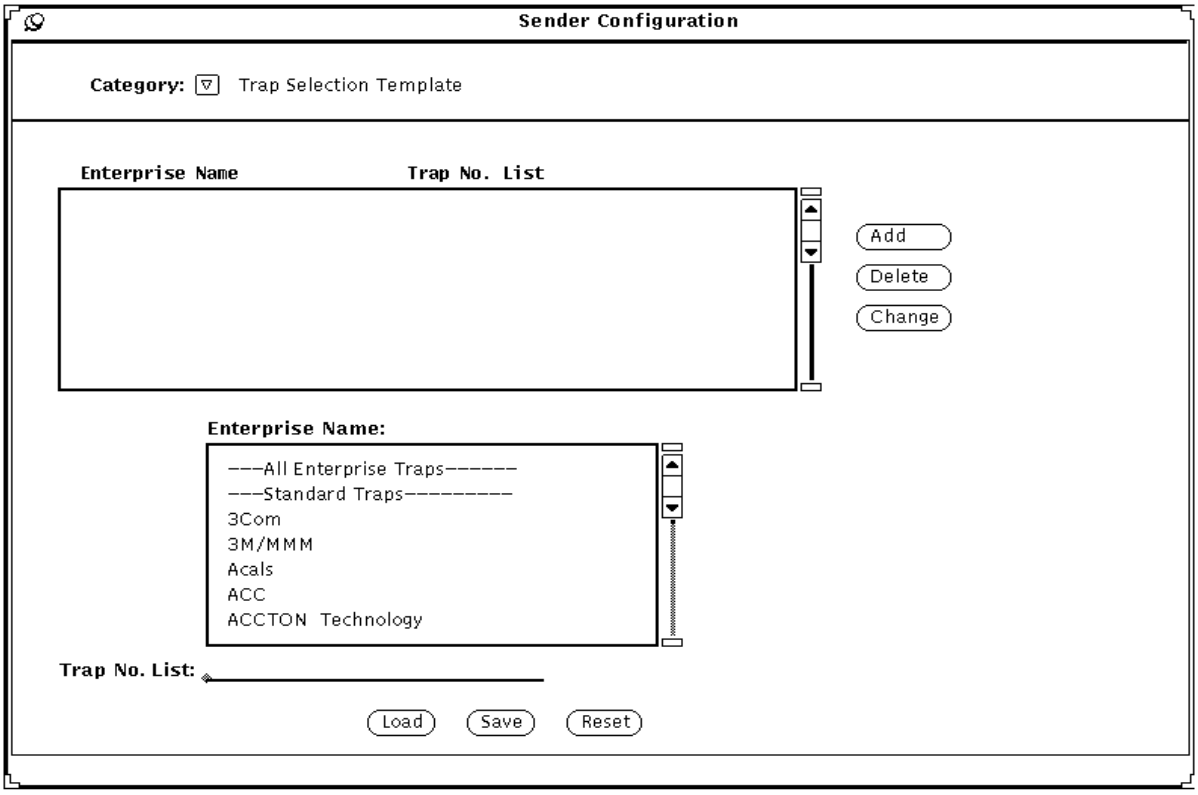

Figure 4-22 Trap Selection Template Window

4.2.4.1 *Loading a Trap Selection Template File*

You can either load an existing Trap Template file or create a new Trap Template. If you want to load an existing Trap Template file, click SELECT on the Load button. A Trap Template Load window appears, as shown in Figure 4-23.

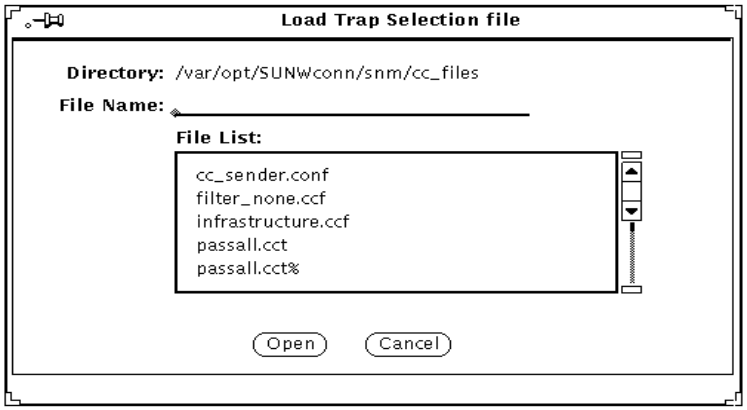

Figure 4-23 Trap Selection Template Load Window

You are prompted to enter the trap selection template file name. If you want to load an existing trap selection template file, select this file name from the **File List** scroll menu, then click SELECT on the Open button.

As shown in Figure 4-22, the Sender's Trap Selection Template contains these columns:

- **Enterprise Name** — This column lists the Enterprise names whose traps are to be forwarded or dropped. Any SNMP Traps that match the Enterprise names in this list are selected. Standard Traps and / or All Enterprise Traps can also be selected from this list.
- **Trap No. List** — This field allows users to select the SNMP trap numbers to be filtered. The numbers can be specified as single numbers or as ranges separated by commas, for example, 1, 2-5, 10.

To create and modify rows in the Trap Selection Template, use the Enterprise Name table and the Trap number list field at the bottom of the window.

4.2.4.2 Adding a Trap Type Entry

If you want to enter a new row in this list, follow these steps:

1. **Click SELECT on the enterprise names provided in the Enterprise Name field.**

   These names are sorted alphabetically (except for the first two items).

2. **Enter a list of trap numbers by typing the numbers or ranges separated by commas in the Trap No. List field.**

   If this field is left blank, then *all* traps for the selected enterprise are specified.

After you have entered the information, click SELECT on the Add button to add the specified Enterprise names and trap numbers to the trap template file.

4.2.4.3 Changing a Trap Type Entry

If you want to modify one of the entries (rows) in the trap selection template, click SELECT on that row to highlight it. The information for that trap is now highlighted in the Enterprise Name table. From this table, scroll up or down to select a new enterprise name. Also, enter the desired trap numbers in the Trap No. List field then click SELECT on the Change button to enter the changes into the template.

4.2.4.4 Deleting a Trap Type Entry

To delete a trap type in the Trap Selection Template, click SELECT on the trap to be deleted, then click SELECT on the Delete button.

4.2.4.5 Saving the Trap Type File

After you have finished adding new trap type entries, or deleting or modifying existing trap type entries in the Trap Selection Template, click SELECT on the Save button to save your changes to this Trap Selection Template file. You are prompted to enter the name of the Trap Selection Template file. If you are saving changes to an existing trap selection template file, select it from the file list scroll menu, then click SELECT on the Save button, as shown in Figure 4-24.

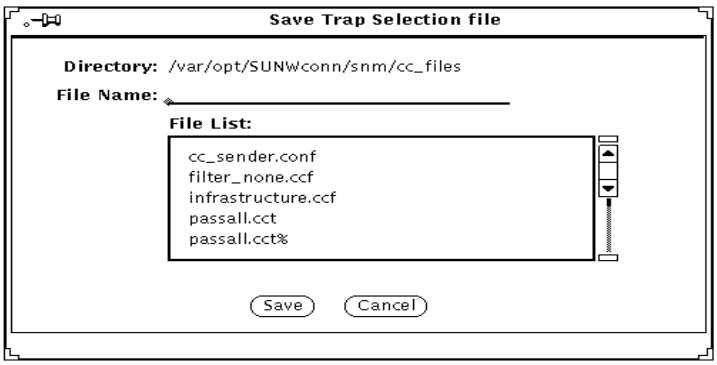

Figure 4-24 Trap Selection Template Save Window

4.2.4.6 *Abandoning Changes to the Trap Selection Template*

If you want to abandon the changes you have made to the Trap Selection Template and revert to the previously saved version of this template file, click SELECT on the Reset button.

4.3 *Configuring Cooperative Consoles for Internationalization*

The default locale for `cc_sender` is **C** (C locale). However, whatever locale a particular SNM database is created in, that same locale should be used for the `cc_sender`. The `cc_sender` can be instructed to use the locale for particular databases with the following file:

`/var/opt/SUNWconn/snm/cc_files/cc_sender.lang`

The format of this file is as follows:

<database_name1> <locale_name1>
<database_name2> <locale_name2>
<database_name3> <locale_name3>

For example, to set the database `mydb` in Japanese (ja) locale and the database `yourdb` in C locale, you would configure the `cc_sender.lang` file as follows:

```
mydb    ja
yourdb  C
```

Cooperative Consoles Examples 5

This chapter lists examples of filter file entries. The scenarios range from simple setups to more difficult setups involving Database Template and Trap Selection Template.

Table 5-1 indicates that by default, the action is to pass all events, SNMP traps and glyph traps. No priorities are set. The DB Template field indicates to forward or drop database traps as specified by the file `passall.cct`.

Table 5-1 All Events and Traps with no Priority Example

Type	Name	Events/Traps	Trap Template	Priority	New Priority	Action	DB Template
default		all	-	-	-	pass	passall.cct

Table 5-2 indicates that by default, the action is to pass all events, SNMP traps and glyph traps that are of priority medium or high. All low priority events, SNMP traps, and glyph traps are dropped. The DB Template field indicates to forward or drop database traps as specified by the file `passall.cct`.

Table 5-2 All Events and Traps with Medium Priority Example

Type	Name	Events/Traps	Trap Template	Priority	New Priority	Action	DB Template
default		all	-	med	-	pass	passall.cct

Table 5-3 indicates that by default, the action is to pass all events, SNMP traps and glyph traps that are of priority medium or high. After these events and traps are forwarded, the new priority will be set to high. All low priority events, SNMP traps, and glyph traps are dropped. The DB Template field indicates to forward or drop database traps as specified by the file `passall.cct`.

Table 5-3 All Events and Traps with Medium / High Priority Example

Type	Name	Events/Traps	Trap Template	Priority	New Priority	Action	DB Template
default		all	-	med	high	pass	passall.cct

Table 5-4 indicates that by default, the action is to pass all events that are of priority medium or high. After these events and traps are forwarded, the new priority will be set to high. All low priority events, SNMP traps, and glyph traps are dropped. The DB Template field indicates to forward or drop database traps as specified by the file `passall.cct`.

Table 5-4 Events Only with Medium / High Priority Example

Type	Name	Events/Traps	Trap Template	Priority	New Priority	Action	DB Template
default		events	-	med	high	pass	passall.cct

Table 5-5 indicates that by default, the action is to pass all events and SNMP traps listed in the trap selection template file `stdtraps.cce` with priority medium or high. Convert the priority of these SNMP traps to high. Drop all other SNMP traps. All low priority events and SNMP traps are dropped. The DB Template field indicates to forward or drop database traps as specified by the file `passall.cct`.

Table 5-5 Events and SNMP Traps with Medium / High Priority Example

Type	Name	Events/Traps	Trap Template	Priority	New Priority	Action	DB Template
default		events, traps	stdtraps.cce	med	high	pass	passall.cct

Table 5-6 indicates to pass all events, SNMP traps and glyph traps that are from host `venus`. Pass or drop the database traps that are from host `venus` as specified in DB Template file `passall.cct`. By default, drop all other events, SNMP traps and glyph traps that are not from `venus`. Pass or drop all other database traps (that are not from venus) as specified by the DB Template file `firewall.cct`.

Table 5-6 Hostname With All Events and Traps Example

Type	Name	Events/Traps	Trap Template	Priority	New Priority	Action	DB Template
hostname	venus	all	-	-	-	pass	passall.cct
default	-	-	-	-	-	drop	firewall.cct

Table 5-7 indicates to pass all events, SNMP traps and glyph traps that are from any component of type `component.router`. Pass or drop the database traps that are from component of type `component.router` as specified in DB Template file `passall.cct`. By default, drop all other events, SNMP traps and glyph traps that are not from element of type `component.router`. Pass or drop all other database traps (that are not from `component.router`) as specified by the DB Template file `firewall.cct`.

Table 5-7 Component Router Example

Type	Name	Events/Traps	Trap Template	Priority	New Priority	Action	DB Template
component	component.router	all	-	-	-	pass	passall.cct
default	-	-	-	-	-	drop	firewall.cct

Table 5-8 indicates to pass all events, SNMP traps and glyph traps that are from any element in the view or subview of `big-cloud`. Pass or drop the database traps that are from any element in the view `big-cloud` or any subviews of it as specified in DB Template file `passall.cct`. By default, drop all other events, SNMP traps and glyph traps. Pass or drop all other database traps (that are not from `big-cloud`) as specified by the DB Template file `firewall.cct`.

Table 5-8 Viewname With all Events and Traps Example

Type	Name	Events/Traps	Trap Template	Priority	New Priority	Action	DB Template
viewname	big-cloud	all	-	-	-	pass	passall.cct
default	-	-	-	-	-	drop	firewall.cct

Table 5-9 indicates to pass all events, SNMP traps and glyph traps that are from any element in view or subview of type `campus`. Pass or drop the database traps that are from any element in view of type `campus` or any subviews of it as specified in DB Template file `passall.cct`. By default, drop all other events, SNMP traps and glyph traps. Pass or drop all other database traps (that are not from `campus`) as specified by the DB Template file `firewall.cct`.

Table 5-9 Viewtype With all Events and Traps Example

Type	Name	Events/Traps	Trap Template	Priority	New Priority	Action	DB Template
viewtype	campus	all	-	-	-	pass	passall.cct
default	-	-	-	-	-	drop	firewall.cct

Table 5-10 indicates to drop all events, SNMP traps and glyph traps that are from host `venus`. Pass or drop the database traps that are from host `venus` as specified in DB Template file `passall.cct`. By default, pass all other events, SNMP traps and glyph traps. Pass or drop all other database traps (that are not from venus) as specified by the DB Template file `firewall.cct`.

Table 5-10 Hostname Drop all Events and Traps Example

Type	Name	Events/Traps	Trap Template	Priority	New Priority	Action	DB Template
hostname	venus	all	-	-	-	drop	passall.cct
default	-	-	-	-	-	pass	firewall.cct

Table 5-11 indicates to pass all events, SNMP traps and glyph traps that are from the host `venus`. Pass or drop the database traps that are from host `venus` as specified in DB Template file `passall.cct`. Also, pass all events, SNMP traps and glyph traps that are from the host `splendid`. Pass or drop the database traps that are from host `splendid` as specified in DB Template file `maypass.cct`. By default, drop all other events, SNMP traps and glyph traps. Pass or drop all other database traps (in this case, that are neither from `venus` nor `splendid`) depending on the DB Template file `firewall.cct`.

In this case, when an event, SNMP trap, glyph trap or database trap is received by the CC Sender program, it is checked to see whether or not it satisfies the criteria mentioned for hostname `venus`. If the criteria is satisfied, then the *pass* action will be performed. If not, the criteria for hostname `splendid` is checked. If this criteria is met, then the action *pass* will be performed. Otherwise, the action *drop* mentioned for the default case will be performed.

Table 5-11 Multiple Hostnames with All Events and Traps Example

Type	Name	Events/Traps	Trap Template	Priority	New Priority	Action	DB Template
hostname	venus	all	-	-	-	pass	passall.cct
hostname	splendid	all	-	-	-	pass	maypass.cct
default	-	-	-	-	-	drop	firewall.cct

Table 5-12 indicates to pass all events, SNMP traps and glyph traps that are from the host `splendid`. Pass or drop the database traps that are from host `splendid` as specified in DB Template file `maypass.cct`. Also, pass all SNMP traps that are from the view or subview of `eternal`. Pass or drop the database traps that are from view `eternal` or its subviews as specified in DB Template file `passnone.cct`. By default, drop all other events, SNMP traps and glyph traps. Pass or drop all other database traps depending on the DB Template file `firewall.cct`.

Since only SNMP traps are passed from the view `eternal`, the events and glyph traps that come from the view `eternal` will satisfy the third line for default and will be dropped.

Interchanging lines 1 and 2 in this filter file will have no effect on the outcome of the filter process. The reason is that the filtering mechanism follows a predefined set of precedences. The precedence will be given on the value of the type field. The lines with type *hostname* will take the highest precedence and the line with *default* will be given the least. The following list indicates the decreasing order of precedence.

- Hostname
- Component
- View Name
- View Type
- Default

So even if the lines in this example are interchanged, it will have no effect on the outcome of the filtering.

Table 5-12 Hostname and Viewname Example

Type	Name	Events/Traps	Trap Template	Priority	New Priority	Action	DB Template
hostname	splendid	all	-	-	-	pass	maypass.cct
viewname	eternal	traps	-	-	-	pass	passnone.cct
default	-	-	-	-	-	drop	firewall.cct

One of the special cases is when the same event/trap is capable of satisfying more than one filter lines. For example, consider the following filter file in Table 5-13.

Assume that host `splendid` appears in the view or subview `eternal`. If an SNMP trap comes from host `splendid`, then according to the first line, it should be passed. But according to the second line, it should be dropped. In that case, the 'first come first served' rule is adopted. Since the condition for host `splendid` is met first, (even if the lines are interchanged) the SNMP trap will be forwarded.

Table 5-13 Hostname and Viewname (Drop) Example

Type	Name	Events/Traps	Trap Template	Priority	New Priority	Action	DB Template
hostname	splendid	all	-	-	-	pass	maypass.cct
viewname	eternal	traps	-	-	-	drop	passnone.cct
default	-	-	-	-	-	drop	firewall.cct

One can specify multiple filter lines for the same host, view or component. For example, consider the following filter file in Table 5-14.

The filtering process will undergo the following steps. All the Database traps that are coming from host `nethound` will be forwarded or dropped as specified in the database template file `maypass.cct`. As a result of the first filter line, all the events and glyph traps from the host `nethound` will be dropped.

As a result of the second filter line, all the SNMP traps that are specified in the trap selection template `stdtraps.cce` and that are from the host `nethound` will be dropped. The SNMP Traps from host `nethound` that are not mentioned in `stdtraps.cce` will be considered for subsequent filtering.

As a result of the third filter line, all the SNMP traps that are specified in the trap selection template `suntraps.cce` and that are from the host `nethound` will be forwarded. The SNMP traps from host `nethound` that are not mentioned in `suntraps.cce` (as well as `stdtraps.cce`) will be considered for subsequent filtering.

As a result of the fourth filter line, all the SNMP traps that are specified in the trap selection template `enterprise.cce` and that are from the host `nethound` will be dropped. The SNMP Traps from host `nethound` that are not

mentioned in `enterprise.cce` (as well as `stdtraps.cce` and `suntraps.cce`) will be processed by the fifth filter line for the default case. In addition, all the events, glyph traps and SNMP traps from all other elements will be processed by the fifth filter line.

As a result of the last filter line, the SNMP traps that are mentioned in the file `enterprise.cce` will be forwarded. The events and glyph traps will be forwarded. All the database traps will be forwarded or dropped per DB Template file `firewall.cct`.

Table 5-14 Same Hostname Example

Type	Name	Events/Traps	Trap Template	Priority	New Priority	Action	DB Template
hostname	nethound	events,glyphs	-	-	-	pass	maypass.cct
hostname	nethound	traps	stdtraps.cce	-	-	drop	maypass.cct
hostname	nethound	traps	suntraps.cce	-	-	pass	maypass.cct
hostname	nethound	traps	enterprise.cce	-	-	drop	maypass.cct
default	-	glyphs,traps	enterprise.cce	-	-	pass	firewall.cct

Table 5-15 indicates to pass all events and glyph traps that are from any component of type `component.router`. Pass the SNMP traps that are specified in the the trap selection template `stdcisco.cce`. Pass or drop the database traps that are from component of type `component.router` as specified in DB Template file `passall.cct`. By default, drop all other events, SNMP traps and glyph traps that are not from element of type `component.router`. Pass or drop all other database traps depending on the DB Template file `firewall.cct`.

Table 5-15 Component Trap Template Example

Type	Name	Events/Traps	Trap Template	Priority	New Priority	Action	DB Template
component	component.router	all	stdcisco.cce	-	-	pass	passall.cct
default	-	-	-	-	-	drop	firewall.cct

Diagnosis A

This chapter provides you with information on what you need to do when you encounter problems using CC. We recommend you check the daemon messages and log files which may help you identify and resolve problems.

Also, we recommend you run CC_Receiver and CC_Sender in debug mode. Running in debug mode allows you to get traces of the operations that are being performed to track whether the programs are performing the operations you expect.

- Check to see if there are any CC_Sender daemon messages logged in the file `/var/adm/messages`.
- Check the SNM standard log file under the default installation directory to see if events and traps are received.
- To run CC_Receiver in Debug mode, you have to customize CC_Receiver with the `-d` option. To customize the CC_Receiver, refer to section “Using the Receiver -h Option” in Chapter 3.

  While you are customizing, you can give the `-dN` option to the CC_Receiver where `N` is the debug trace level that can range from 1 to 5. The higher the trace level, the more detailed the information.

  The trace output is displayed on the standard error and will give you an idea of what is happening.
- To run CC_Sender in Debug mode, perform the following:

1. Login as superuser.

2. Comment out the lines corresponding to CC_Sender in the file `/etc/inetd.conf`.

   There are two entries corresponding to CC_Sender in this file. Search for *sender* and insert the character '#' at the beginning of these two lines to comment out the lines.

3. Search and kill the `inetd` process with the `-HUP` signal.

```
# ps -ef | grep inetd
# kill -HUP <proc-id of inetd>
```

4. Start CC_Sender manually.

   If you have installed CC_Sender in the default directory, type the following command line at your prompt:

```
# /opt/SUNWconn/snm/agents/cc_sender -dN
```

   `N` is the debug trace level that can range from 1 to 5. The higher the trace level, the more detailed the information. The trace output is displayed on the standard error and will give you an idea of what is happening.

Once you are done with this, to run CC_Sender automatically, perform the following steps:

1. Remove the comments ('#' characters) corresponding to CC_Sender in the file `/etc/inetd.conf`.

2. Search and kill the `inetd` process with the `-HUP` option as described earlier.

   This allows CC_Sender to run automatically.

Index

G

H

I

L

N

P

R

S

T

V