

# CA Spectrum®

## Distributed SpectroSERVER Administrator Guide

Release 9.4



This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the "Documentation") is for your informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and may not be disclosed by you or used for any purpose other than as may be permitted in (i) a separate agreement between you and CA governing your use of the CA software to which the Documentation relates; or (ii) a separate confidentiality agreement between you and CA.

Notwithstanding the foregoing, if you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with "Restricted Rights." Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2014 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

# CA Technologies Product References

This guide references CA Spectrum®.

## Contact CA Technologies

### Contact CA Support

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

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

### Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to [techpubs@ca.com](mailto:techpubs@ca.com).

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



# Contents

---

## Chapter 1: Introducing Distributed SpectroSERVER 9

About Distributed SpectroSERVER .....	9
Landscapes .....	10
Landscape Maps .....	10
Modeling Catalogs .....	11
User Models .....	11
SpectroSERVER (.vnmrc) Resources .....	12
General SpectroSERVER (.vnmrc) Resources .....	13
Events Archive (.vnmrc) Resources .....	18
Work Thread (.vnmrc) Resources .....	19
Fault Tolerant Alarm Service (.vnmrc) Resources .....	21
How to Pack Up CA Spectrum Utilities and Move Them to Another Computer .....	22
Pack Up CA Spectrum Utilities .....	23
Move CA Spectrum Utilities to Another Computer .....	24

## Chapter 2: Setting Up a Distributed SpectroSERVER Environment 29

Name Resolution Requirements .....	29
CA Spectrum and Multiple Interfaces .....	29
Communication Among SpectroSERVERs .....	30
Port Conflict Resolution .....	30
DSS Environment Requirements .....	32
Location Servers .....	32
How Location Servers Interact .....	34
Main Location Server Connection .....	34
Designate a New Main Location Server .....	35
Landscape Map Integrity Preservation .....	36
Landscape Handles .....	36
Assign Landscape Handles .....	36
Change a Landscape Handle .....	37
Process Daemon (processd) .....	37
Processd Differences in Windows and Solaris Environments .....	38
Change the Windows Password in Processd .....	39
Install Tickets Files .....	40
Stop and Restart Processd .....	45
How the Userconf Process Works During Installation .....	46
How Userconf Works When a User Logs In .....	47

---

Network Partitioning .....	47
Duplicate Models in a Distributed Environment .....	48
Failure to Contact Home Landscape Errors.....	48
Host Resource Configuration File (.hostrc) .....	49
Time Zones in a DSS Environment.....	49
SpectroSERVER Shutdown.....	49
SpectroSERVER Shutdown in a Solaris Environment.....	50
SpectroSERVER Shutdown in a Windows Environment .....	50
Set the Landscape Map Entry Timeout .....	50
Problems with Landscape Mapping from Existing DSS Setup .....	51
Problems with Purging Old Landscapes .....	52

## Chapter 3: Communication Across Firewalls in the Distributed SpectroSERVER Environment 55

Communication Across Firewalls .....	55
SpectroSERVER and OneClick Web Server Communication Across Firewalls .....	56
OneClick Default Ports and Firewalls .....	57
HTTP Listen Port .....	57
CORBA Listen Port .....	57
Remote SpectroSERVERs and Firewalls.....	57
Primary and Secondary SpectroSERVER Communication Across Firewalls.....	59
CA Spectrum Configuration Files for NAT Firewall Environments.....	60
Default Port Configurations .....	60
Change the SpectroSERVER Port Number.....	60
Change the Archive Manager Port Number and Socket Number .....	61
Change the Location Server Port Number and Socket Number.....	61
Change the Visibroker Naming Service Port Number .....	62
Remote Copy Process Daemon (rcpd) Port Number Configuration.....	62
CLI Daemon (vnmshd) Port Number Configuration .....	62

## Chapter 4: Fault Tolerance 63

About SpectroSERVER Fault Tolerance .....	63
SpectroSERVER Precedence in a Fault Tolerant Environment .....	64
SpectroSERVER Data Synchronization.....	64
Support for Fault-Tolerant Archive Manager.....	65
Archive Manager Data Synchronization.....	66
Generate an Alarm if the Secondary SpectroSERVER Is Not Restarted .....	66
Secondary SpectroSERVER Readiness Levels .....	67
SpectroSERVER Alarm Synchronization.....	68
Synchronization from the Primary to the Secondary SpectroSERVER .....	69
Synchronization from the Secondary to the Primary SpectroSERVER .....	70

---

Establish Fault Tolerance.....	72
Validate Fault Tolerance Configuration .....	74
Test Fault Tolerance .....	75
Fault-Tolerant Recovery .....	76
Change the Host Names of the Primary and Secondary SpectroSERVERs .....	76
Monitor the Changeover Between the Primary and Secondary SpectroSERVERs .....	78
How to Monitor the Secondary SpectroSERVER Status .....	79

## **Chapter 5: Working with Trap Director 81**

Trap Director .....	81
Traps and Memory Usage .....	82
Trap Data Traffic Consolidation.....	82
How Trap Director Updates the Address Cache.....	83
Trap Director in a Fault-Tolerant Setup .....	84
Trap Storm Settings.....	84
Enable and Disable Trap Director.....	84
Define the Cache Record Retention Period .....	85

## **Index 87**





# Chapter 1: Introducing Distributed SpectroSERVER

---

This section contains the following topics:

[About Distributed SpectroSERVER](#) (see page 9)

[Landscapes](#) (see page 10)

[Landscape Maps](#) (see page 10)

[Modeling Catalogs](#) (see page 11)

[User Models](#) (see page 11)

[SpectroSERVER \(.vnmrc\) Resources](#) (see page 12)

[How to Pack Up CA Spectrum Utilities and Move Them to Another Computer](#) (see page 22)

## About Distributed SpectroSERVER

Distributed SpectroSERVER (DSS) is a powerful modeling feature that enables the distribution of management over a large-scale infrastructure. The infrastructure can be organized geographically or across multiple servers in a single physical location. DSS can improve CA Spectrum performance when managing a computing infrastructure. Performance improvements can come from distributing the network load from management traffic and delegating tasks to remote workstations.

Using DSS, you can create a unified representation of the infrastructure that is composed of multiple landscapes, each with a local SpectroSERVER. In a DSS environment, a SpectroSERVER client, such as the OneClick Console, can simultaneously access information from multiple SpectroSERVERs.

DSS includes the following features:

### **Distributed Visibility**

The OneClick console displays information for all deployed SpectroSERVERs. Summary alarm counts are displayed for each SpectroSERVER. The network manager can quickly locate trouble areas.

### **Localized Polling Traffic**

The management workstation that is polling the network is geographically closer to the devices that it manages with DSS. This setup reduces traffic on wide area links and avoids congestion on the local network. Multiple, smaller SpectroSERVERs generate less traffic than a single SpectroSERVER that polls devices from a great distance.

### Scalability

As a network expands, it is often less expensive to use low-end workstations as additional, dedicated SpectroSERVER servers. Such a solution is preferable to upgrading one workstation to accommodate one SpectroSERVER in a large, non-distributed environment.

### Fault Tolerance

DSS supports fault tolerance between SpectroSERVERs. A secondary SpectroSERVER can be provided as a fault tolerant backup or standby for a primary SpectroSERVER. Network management continues if the workstation running the primary SpectroSERVER fails.

You can protect against failures by simply reloading the database for a given landscape onto a secondary SpectroSERVER. When a failure occurs that disables the primary SpectroSERVER, the secondary SpectroSERVER automatically takes its place. All CA Spectrum applications automatically use the secondary SpectroSERVER.

### More information:

[Establish Fault Tolerance](#) (see page 72)

## Landscapes

A network domain that a single SpectroSERVER manages is known as a *landscape*. A landscape consists of the models, associations, attribute values, alarms, events, and statistics that belong to a specific SpectroSERVER. Each landscape in a network is unique. A unique landscape handle (ID) identifies each landscape.

In OneClick, a landscape icon represents each landscape. The landscape icon represents a SpectroSERVER database. Through double-click zones and menu selections, the landscape icon lets you access remote network models. The icon also presents a rollup of alarm information for the devices that are modeled in those remote databases.

**Note:** The terms *local* and *remote* are used to designate landscapes from the perspective of a particular SpectroSERVER.

## Landscape Maps

CA Spectrum automatically maintains a "map" of all of the landscapes that comprise a particular DSS environment. Changes that occur in the SpectroSERVER topology initiate an internal discovery mechanism that updates the landscape maps of all of the other SpectroSERVERs in that environment. For example, discovery detects a SpectroSERVER that is added or removed from the network.

A landscape model is a container model. Landscape models let you connect to other SpectroSERVERs in the landscape map through the CA Spectrum user interface. You can create a landscape model at any level of the three view hierarchies: Topology, Location, or Organization.

## Modeling Catalogs

A modeling catalog is a set of templates (such as model types or relations) that are installed on a particular SpectroSERVER. These templates are used to create models. The models that are created through these templates compose the landscape of that SpectroSERVER.

**Note:** OneClick queries the default landscape for a list of available model types at startup.

When you model multiple landscapes using DSS, each landscape must contain identical modeling catalogs. All model types in the modeling catalog of one landscape must also exist in the modeling catalog of every other landscape to which it connects. If you install a new management module on one SpectroSERVER, install the same management module on every other SpectroSERVER in the landscape map.

## User Models

A user model contains information about individual user permissions and other user data. This information is stored in a landscape. When you first install CA Spectrum, a default user model represents the user that you specified in the Installation Owner field during CA Spectrum installation.

**Note:** For more information, see the *Administrator Guide*.

In a DSS environment, the same user model must exist in all landscapes to enable users to connect to remote SpectroSERVERs in the landscape map and to enable the landscapes to communicate with each other. Adding or modifying a user model causes the SpectroSERVER associated with the landscape to query all other SpectroSERVERs in the landscape map. The query checks for the user model in other landscapes. If the user model is found in other landscapes, their user models are automatically updated to reflect any changes to the initial user model.

**Note:** Deleting a user model only deletes the user model in that landscape. You must manually delete duplicate user models on other SpectroSERVERs in the landscape map.

## SpectroSERVER (.vnmrc) Resources

SpectroSERVER resources are defined in the <\$SPECROOT>/SPECTRUM/SS/.vnmrc file. Many default values of these resource files are built into the code and remain blank. However, the coded default values are not used when values are specified in these resources.

Resources in the .vnmrc (Virtual Network Machine runtime configuration) file define path names and default settings for the SpectroSERVER. The CA Spectrum system software includes a runtime configuration file with default settings for SpectroSERVER resources.

You can modify resources to make the following changes:

- Define general SpectroSERVER resources and directory paths
- Adjust events archiving
- Specify name service variables
- Adjust thread allocation
- Control fault tolerant alarm synchronization

All of these resource entries are listed in the format “resource = resource\_value”. For many of the resources in the .vnmrc file, the resource value is blank.

If no value appears with a specific resource, CA Spectrum uses the default value. The resources in the .vnmrc file take effect when SpectroSERVER is started. If you change this file while SpectroSERVER is running, the changes take effect when SpectroSERVER is restarted.

### Example: Define a .vnmrc resource

This example sets the maximum number of records logged in the Event Log database:

```
max_event_records=5000
```

**max\_event\_records**

Is the resource name.

**5000**

Is the resource value.

## General SpectroSERVER (.vnmrc) Resources

The general SpectroSERVER (.vnmrc) resources control many SpectroSERVER functions. The following list describes the general SpectroSERVER (.vnmrc) resources:

### **comm\_port**

Specifies the TCP port that client user interfaces use to communicate with SpectroSERVER. The port number is defined during installation.

This command has the following format:

```
comm_port=0xBEEF
```

#### **0xBEEF**

Is the default TCP connection port socket. This socket can be any valid TCP port that is greater than the port number that is assigned to the IPPORT\_USERRESERVED parameter in the file /netinet/in.h. However, the port must be less than 65535 (0xFFFF).

### **snmp\_trap\_port\_enabled**

Binds the SpectroSERVER to the SNMP trap port and listen for traps. When set to False, the SpectroSERVER does not bind to the SNMP trap port.

This command has the following format:

```
snmp_trap_port_enabled=TRUE
```

**Default:** True

### **vnm\_file\_path**

Specifies the root subdirectory that contains SpectroSERVER external files, such as database files.

This command has the following format:

```
vnm_file_path=<directory>/spectrum/SS/CsVendor
```

#### **/spectrum/SS/CsVendor**

Is the file path for the root subdirectory

### **tcp\_buffer\_size**

Lets you alter the buffer size that is appropriate for traffic on your LAN. A large buffer size improves throughput. The resource accepts a numeric value that represents the number of bytes for the TCP buffer. This command has the following format:

```
tcp_buffer_size=<blank>
```

**Default:** blank

**Limits:** 8192 to 65536 bytes

**Note:** Values below 8 KB are rounded up to 8 KB. Values above 64 KB are set to 64 KB.

#### **resource\_file\_path**

Is the file path for VNM resource files, such as Ether Map.

This command has the following format:

```
resource_file_path=./CsResource
```

#### **./CsResource**

(Optional) Is the file path for VNM resource files. This parameter can be left blank.

#### **wait\_active**

Determines whether the server accepts connections as soon as all models are loaded or waits until all models are active. If set to *Yes*, a Control Panel message displays a running percentage of models that were activated during SpectroSERVER startup.

This command has the following format:

```
wait_active=no
```

**Default:** no.

#### **max\_bind\_retry\_count**

Specifies the maximum number of listen socket bind retries.

This command has the following format:

```
max_bind_retry_count=50
```

**Default:** 50.

#### **bind\_retry\_interval**

Specifies the delay in seconds between listen socket bind retries.

This command has the following format:

```
bind_retry_interval=30
```

**Default:** 30 seconds.

#### **min\_client\_version**

Is the minimum parm block version that the SpectroSERVER accepts for client connections.

This command has the following format:

```
min_client_version=0
```

**Default:** 0 minimum client connections.

**max\_connections**

Is the maximum number of connections the SpectroSERVER accepts. Connections can be from clients or from other servers, such as another SpectroSERVER, Archive Manager, or Location Server (main and local).

This command has the following format:

```
max_connections=50 gensv000316 docs002958
```

**Default:** 50 connections

A connection can fail when the maximum number of client connections is exceeded. When you increase the maximum SpectroSERVER connections, an increase the number of open file descriptors on your workstation is sometimes required for improved performance.

**handshake\_timeout**

Sets the time in seconds for exchanging initial ID information during a connection handshake.

This command has the following format:

```
handshake_timeout=40
```

**Default:** 40 seconds

**vnm\_message\_timeout**

The vnm\_message\_timeout resource sets the timeout in seconds for messages that are sent between VNMs.

This command has the following format:

```
vnm_message_timeout=180
```

**Default:** 180 seconds

**vnm\_close\_timeout**

The vnm\_close\_timeout resource sets the timeout in seconds for closing the connection between VNMs.

This command has the following format:

```
vnm_close_timeout=180
```

**Default:** 180 seconds

**connect\_time\_limit**

Sets the timeout in milliseconds for connection to the VNM.

This command has the following format:

```
connect_time_limit=5000
```

**Default:** 5000 milliseconds

#### **rcpd\_comm\_port**

Specifies the listening port of the Remote Copy Process Daemon (rcpd). This resource is used for fault-tolerant database backups.

This command has the following format:

```
rcpd_comm_port=0xCAFE
```

**Default:** 0xCAFE

#### **procd\_comm\_port**

Specifies the port that the VNM uses to connect to the process daemon.

This command has the following format:

```
procd_comm_port=0xFEED
```

**Default:** 0xFEED

#### **snmp\_trap\_port**

Specifies the port through which the VNM receives traps.

This command has the following format:

```
snmp_trap_port=162
```

**Default:** 162

#### **enable\_traps\_for\_pingables**

Specifies whether CA Spectrum can receive SNMP traps on pingable models. By default, the .vnmrc file does not contain this resource. Add it to the file manually, using the following syntax:

```
enable_traps_for_pingables = TRUE
```

**Default:** TRUE

#### **max\_device**

Applies to CA Spectrum products whose device\_limit resource is in effect. By default, CA Spectrum generates a yellow alarm when the number of models reaches 80% of the specified device limit. CA Spectrum generates a red alarm when the number of models reaches 100% of the specified device limit. This resource lets you set a value that falls below the actual device limit, which gives you an earlier warning.

This command has the following format:

```
max_devices=<# of device models>
```

##### ***# of device models***

Refers to models with IP addresses that are derived from the device model type AND whose specified number falls below the value of the device\_limit resource. The default value is blank.



**disable\_redundancy\_when\_using\_loopback**

Disables redundancy (0x11d2c) when modeling a device by loopback. This parameter is only applicable when the use\_loopback (0x12bb) attribute of the VNM model is set to True.

This command has the following format:

```
disable_redundancy_when_using_loopback=False
```

By default, this resource does not appear in the .vnmrc file and is therefore automatically evaluated as False.

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

**persistent\_alarms\_active**

Prevents CA Spectrum from retaining alarm-related information when the SpectroSERVER shuts down.

This command has the following format:

```
persistent_alarms_active=<False>
```

By default, this resource does not appear in the .vnmrc file. CA Spectrum automatically retains alarm-related information (such as troubleshooter assignments or status) when the SpectroSERVER is shut down and is brought back up. The alarms that were present before shutdown are preserved. These alarms are considered "persistent" alarms. Adding persistent\_alarms\_active=False to the .vnmrc file prevents CA Spectrum from retaining alarm-related information when the SpectroSERVER shuts down.

**Note:** We strongly recommend against adding the persistent\_alarms\_active resource unless you integrate a third-party application that retains alarm-related information.

Adding this resource has the following effects:

- Alarm status additions are lost (only recorded as past events)
- Existing alarms on SpectroSERVER shutdown are regenerated at startup with new timestamps.
- Regenerated alarms are forwarded to alarm notification tools.

**unsupported\_attr\_poll\_interval**

Specifies the amount of time that CA Spectrum waits to poll an external attribute that has returned a "noSuchName" error. If this parameter is not specified in the .vnmrc file, the default value (12 hours) is used.

This command has the following format:

```
unsupported_attr_poll_interval=43200
```

**Default:** 43200 seconds (12 hours)

**More information:**

[Remote Copy Process Daemon \(rcpd\) Port Number Configuration](#) (see page 62)

## Events Archive (.vnmrc) Resources

The events archive (.vnmrc) resources control the process of the SpectroSERVER sending events to the Archive Manager for logging. The following list describes the events archive (.vnmrc) resources:

**max\_event\_records**

Specifies the maximum number of records that can be stored in the Event Log database.

This command has the following format:

`max_event_records=# of Records`

**# of Record**

Is the number of Event Log records. The minimum value is equal to the event\_record\_increment value. System storage capacity limits the maximum value.

**Default:** 20,000

**event\_record\_increment**

Specifies the number of records to delete from the Event Log database when the number of records exceeds the max\_event\_records value.

This command has the following format:

`event_record_increment=# of Records`

**# of Records**

Is the number of Event Log records that are deleted from the database. The minimum value is 100.

**event\_batch\_max\_size**

Sets the maximum number of events per batch. If the batch becomes full, the events batch is immediately sent to the Archive Manager.

This command has the following format:

`event_batch_max_size=1000`

**Default:** 1000 (the maximum number of events)

**event\_batch\_timeout**

Determines the amount of time in seconds when the events batch is sent over to the Archive Manager.

This command has the following format:

```
event_batch_timeout=1
```

**Default:** 1 second

**log\_user\_events**

Controls whether an event is generated for each user-initiated write to a model attribute value. A value of True causes the VNM to generate events.

This command has the following format:

```
log_user_events=False
```

**Default:** False

**use\_log\_queue**

Places events in a separate queue when they are generated. These events are sent to the Archive Manager on a separate thread. If set to "TRUE", the event is sent to the Archive Manager on the same thread on which it was generated. This situation can delay alarm creation.

This command has the following format:

```
use_log_queue=<blank>
```

**Default:** <blank>

## Work Thread (.vnmrc) Resources

SpectroSERVER is a multithreaded process. During normal operation, each subsystem allocates numerous work threads.

SpectroSERVER maintains a pool of work threads that are reused over time. Subsystems use threads from the pool, up to their individual limits, during periods of increased activity. When the common pool of work threads is exhausted, new work threads are created. The pool grows to accommodate the increased activity.

Work threads that subsystems no longer require return to the common pool for later use. Threads are taken from the pool to serve subsequent needs or new threads are allocated if the pool is empty. Threads that remain unused for a specified time are removed from the pool, and their resources are returned to the system. This process is termed aging.

The following list describes the work thread (.vnmrc) resources:

**max\_total\_work\_threads**

Specifies the maximum number of work threads that can be allocated for all SpectroSERVER subsystems. Each work thread consumes processing resources and requires a significant block of memory. Set its value based on system capacity (memory size and speed).

When the value of this resource is too high, SpectroSERVER can periodically run out of memory. When the value of this resource is too low, SpectroSERVER operations can become sluggish.

If the value of this resource is newly set or changed, CA Spectrum reads the new value when the SpectroSERVER is restarted. Once the value is read, it is used to update the value of the WorkThreadsMaxAvail attribute on the VNM model. When the value of WorkThreadsMaxAvail has been updated, the value for max\_total\_work\_threads is removed from the .vnmrc file.

This command has the following format:

`max_total_work_threads=# of threads`

The default value of the VNM WorkThreadsMaxAvail attribute is 500.

**work\_thread\_age**

Work threads that a subsystem no longer requires are returned to the work thread pool. This resource specifies how long (in seconds) a work thread can remain in the pool without being used.

Set this resource to a value that is compatible with subsystem activity. Significant processing overhead is required to create a work thread. Setting the value of this resource too low taxes system resources by creating new threads too often in response to work thread demands. Setting the value too high means that resources remain committed unnecessarily.

This command has the following format:

`work_thread_age =seconds`

**Default:** 60 seconds

## Fault Tolerant Alarm Service (.vnmrc) Resources

In a fault-tolerant environment, alarms must be synchronized between the primary and secondary SpectroSERVERs. The servers connect to exchange alarm information. If the initial connection attempt fails, subsequent attempts can be made. The Fault Tolerant Alarm Service controls alarm synchronization using the following settings:

### **ftasv\_enabled**

Enables alarm synchronization. Include this command in the .vnmrc file for both the primary and secondary servers.

This command has the following format:

```
ftasv_enabled=true
```

**Default:** True

**Important!** You cannot have `ftasv_enabled` set to True for one server and set to False for the other server. Use the same value for both servers.

### **ftasv\_max\_conn\_retry\_count**

Specifies the number of times that the primary server attempts to connect to the secondary server for synchronization after a connection attempt has failed. This parameter is required in the .vnmrc file of the primary SpectroSERVER.

**Note:** The primary server uses this parameter when attempting to connect to the secondary server. It is not used when the secondary server attempts to connect to the primary.

This command has the following format:

```
ftasv_max_conn_retry_count=# of retries
```

**Default:** Four retries (for a total of five synchronization attempts with the secondary server).

### **ftasv\_conn\_retry\_interval**

Specifies the number of seconds between primary server attempts to synchronize with the secondary server. Use this parameter in the .vnmrc file of the primary SpectroSERVER.

**Note:** The primary server uses this parameter when attempting to connect to the secondary server. This parameter is not used when the secondary server attempts to connect to the primary.

This command has the following format:

```
ftasv_conn_retry_interval=# of seconds
```

**Default:** 30 seconds

### **ftasv\_debug**

Enables debug output for alarm synchronization activity. Include this command in the .vnmrc file for both the primary and secondary servers. Debug output is written to the VNM.OUT file of each server. Each message begins with "Fault Tolerant Alarm Service."

This command has the following format:

```
ftasv_debug=true
```

**Default:** False

**Important!** If the secondary SpectroSERVER is not running when the primary server attempts to connect to it, the primary exhausts its synchronization attempts. As a result, startup of the primary SpectroSERVER is delayed. When using default settings for ftasv\_max\_conn\_retry\_count and ftasv\_conn\_retry\_interval, the delay is two minutes (four retries with 30 seconds intervening). This delay is unavoidable because it occurs before model activation. The workaround is to verify that the secondary SpectroSERVER is running when the primary starts. You can also decrease the retry count or the interval size to reduce the potential delay.

### **More information:**

[Fault Tolerance](#) (see page 63)

[SpectroSERVER Alarm Synchronization](#) (see page 68)

## How to Pack Up CA Spectrum Utilities and Move Them to Another Computer

The packtool.pl script packs up the CA Spectrum Command Line Interface (CLI), CA Spectrum SS Logger, AlarmNotifier, the modelinggateway tool, and the sbgwimport tool so that you can transfer them to another computer. The script retains the relative directory structure of the utilities and their support files when the files are moved.

**Note:** Both computers must be running the same operating system.

Consider the following requirements before running the packtool.pl script:

- You must be a root user on Solaris and Linux platforms, or a user with administrative privileges on Windows platforms to run the packtool.pl script.
- The following requirements apply when you transfer the CA Spectrum utilities to a computer where only the OneClick server is installed:
  - The version and patch level of the OneClick server on each computer must match. After each patch installation, run the packtool.pl script again, and transfer the utilities again.
  - The installation user on the computer where the utilities are transferred must match the installation user on the computer where the files are packaged.
  - If a debug patch or PTF is installed on the target computer, reinstall the debug patch or PTF after the file transfer.
- Stop all CA Spectrum processes on the computer where the utilities are packaged.
- Stop all CA Spectrum processes on the computer where you are transferring the utilities before you extract the tools\_bundle file.
- To extract the tools\_bundle file, you must be a root user on Solaris and Linux platforms, or a user with administrative privileges on Windows.

**Important!** To run the CA Spectrum utilities, you must be logged on to the target computer as a valid CA Spectrum user or an error occurs. For example, if you are logged on to a Linux workstation as “root,” and there is no matching user model in CA Spectrum, you receive an error stating, “NO\_USER.”

To package CA Spectrum utilities and transfer them to another computer, take the following steps:

1. [Pack Up the SPECTRUM Utilities using the packtool.pl script.](#) (see page 23)
2. [Extract the tools\\_bundle file on another computer](#) (see page 24).

## Pack Up CA Spectrum Utilities

Before moving the CLI, SSLogger, AlarmNotifier, the modelinggateway tool, and the sbgwimport tool to another computer, run a script to bundle them.

**Follow these steps:**

1. Verify that the environmental variable <SPECROOT> is set to the CA Spectrum installation directory path on the computer where you are packing up the utilities.
2. Set the CYGWIN32 environment variable to the directory where the cygwin is installed.
3. Run the packtool.pl script, which packs up the utilities and their support files. The packtool.pl script can be found in the <SPECROOT>/SS-Tools directory.

To run the script from a Bash shell or other UNIX shell, enter the following command:

```
<${SPECROOT}>/SS-Tools/packtool.pl [-no_notifier | -no_event_alarms]  
[-f file_name]
```

**<\${SPECROOT}>**

Is the directory structure where CA Spectrum is installed on your SpectroSERVER.

**-no\_notifier**

Specifies that you do not want to pack up AlarmNotifier.

**-no\_event\_alarms**

Specifies that you do not want to pack up AlarmNotifier EvFormat or PCause files.

**-f file\_name**

Specifies a name for the executable file.

An executable file named linux\_tools\_bundle (Linux), solaris\_tools\_bundle (Solaris), or nt\_tools\_bundle.exe (Windows) that contains the CA Spectrum utilities and their support files is created.

## Move CA Spectrum Utilities to Another Computer

You can move CA Spectrum utilities to a computer that is only running the OneClick server.

**Follow these steps:**

1. [Pack up the CA Spectrum utilities](#) (see page 23).
2. On the computer where you want to extract the CA Spectrum utilities, change the directory to <\${SPECROOT}>.
3. Using binary mode, FTP the executable file named linux\_tools\_bundle (Linux), solaris\_tools\_bundle (Solaris), or nt\_tools\_bundle.exe (Windows) to the current directory.
4. Extract the tools\_bundle file from the DOS, Bash, or other UNIX shell.

The CA Spectrum utilities and their support files unpack into the appropriate directory structure.



5. If you transferred AlarmNotifier and event and pcause files, verify that the paths in `<$SPECROOT>/SG-Support/CsResource/preferences/*.prf` have the correct locations for the event and pcause files. Also, verify the `ui=` and the `lhandle=` options.
6. Set the `<$SPECPATH>` environmental variable to the path of the directory where you extracted the `tools_bundle` file:
  - On Windows, create the system environment variable `<$SPECPATH>` with the value in variable format:  
`driveletter:\PATH_TO_SPECTRUM`
  - On Solaris, add the following line to the `/opt/SPECTRUM/spectrum60.env` file:  
`SPECPATH=PATH_TO_SPECTRUM`
  - On Linux, add the following line to the `/opt/SPECTRUM/spectrum80.env` file:  
`SPECPATH=PATH_TO_SPECTRUM`
7. On Solaris and Linux platforms, define `BES_LIC_DIR` as `PATH_TO_SPECTRUM/bin/VBNS/license` in the `spectrum*.env` file in the `/opt/SPECTRUM` directory.
8. On Linux platforms, copy `/usr/bin/perl` to `<$SPECROOT>/bin`.
9. Verify that the `.hostrc` file contains the local host name and the name of the computer from which you transferred the utilities.
10. Verify that the `.hostrc` file on the main location server contains the host name of the computer where the `tools_bundle` file is extracted.
11. Verify that the `.LocalRegFile` contains the correct Main Location Server.
12. Set `CLISESSID` in a shell to use the CLI utility:

**Windows:**

```
set CLISESSID=<NUMBER>
```

**Solaris and Linux:**

```
export CLISESSID=<NUMBER>
```

**NUMBER**

Is any unique number for the shell.

13. Verify that `Notifier/.alarmrc` has the correct path to the `SetScript`, `ClearScript`, and `UpdateScript`. These scripts are located in the `Notifier` directory where you extracted the `tools_bundle` file.
14. Restart `processd`.

You can now run the utilities on this computer.

You can also move CA Spectrum utilities to a computer that is not running CA Spectrum.

**Follow these steps:**

1. Pack up the CA Spectrum utilities.
2. On the computer where you want to extract the CA Spectrum utilities, create a directory to unpack the CA Spectrum utilities and their support files. For example, create the following directories:

**Windows:**

c:\win32app\spectrum

**Solaris and Linux:**

/usr/spectrum

**Note:** Change the working directory (chdir) to these directories.

3. Send the executable file by FTP to the current directory using binary mode. The file has one of the following names: linux\_tools\_bundle (Linux), solaris\_tools\_bundle (Solaris), or nt\_tools\_bundle.exe (Windows).
4. Execute the tools\_bundle file from the DOS, Bash, or other UNIX shell.

The CA Spectrum utilities and their support files unpack into the appropriate directory structure.

5. Verify that the PATH variable is as follows:

**Windows:**

Verify that <\$SPECROOT>/lib is in the PATH variable.

**Solaris and Linux:**

- Create an /opt/SPECTRUM directory.
  - Create a link from <\$SPECROOT>/lib to /opt/SPECTRUM/lib.
  - Create a link from <\$SPECROOT>/bin to /opt/SPECTRUM/bin.
6. Set the <\$SPECROOT> and <\$SPECPATH> environmental variables to the path of the directory where you extracted the tools\_bundle file:

- On Windows, create the system environment variables <\$SPECROOT> and <\$SPECPATH> with the value in variable format:

driveletter:/PATH\_TO\_SPECTRUM  
driveletter:\PATH\_TO\_SPECTRUM

- On Solaris, create /opt/SPECTRUM/spectrum60.env and add the following lines:

SPECROOT=PATH\_TO\_SPECTRUM  
SPECPATH=PATH\_TO\_SPECTRUM

- On Linux, create /opt/SPECTRUM/spectrum80.env and add the following lines:

```
SPECROOT=PATH_TO_SPECTRUM
SPECPATH=PATH_TO_SPECTRUM
```

7. If you transferred AlarmNotifier and event and pcause files, verify that the paths in <SPECROOT>/SG-Support/CsResource/preferences/\*.prf have the correct locations for the event and pcause files. Also, verify the ui= and the lhandle= options.
8. On Solaris and Linux platforms, define BES\_LIC\_DIR as PATH\_TO\_spectrum/bin/VBNS/license in the spectrum\*.env file in the /opt/SPECTRUM directory.
9. Verify that the .hostrc file contains the local computer host name and the name of the computer from which you transferred the utilities.
10. Verify that the .LocalRegFile contains the correct Main Location Server.
11. Verify that vnmsh/.vnmshrc contains the correct main SpectroSERVER name.
12. Install the processd service:

**Windows:**

- If SRAdmin is not installed, install it as follows:

```
shell> cd %SPECROOT%\Install-Tools\sdic\nt
shell> .\sradmin --install
shell> .\sradmin --start
```

- Install the processd service:

```
shell> cd %SPECROOT%\lib\SDPM
shell> .\processd.exe --install --username USERNAME --password PASSWORD
shell> .\processd.exe --start
```

**Note:** If processd does not start, reboot the computer.

**Solaris and Linux:**

- Copy <SPECROOT>/lib/SDPM/processd\_init.sh to /etc/init.d/processd.
- Copy <SPECROOT>/lib/SDPM/processd.pl to /etc/init.d.
- Create a link from /etc/init.d/processd to /etc/rc2.d/S99processd.
- Start processd using /etc/init.d/processd start.

13. Verify that the .hostrc file on the main location server contains the host name of the computer where the tools\_bundle file is extracted.
14. Set CLISESSID in a shell to use the CLI utility:

**Windows:**

```
set CLISESSID=<NUMBER>
```

**Solaris and Linux:**

```
export CLISESSID=<NUMBER>
```

**NUMBER**

Is any unique number for the shell.

15. Verify that Notifier/.alarmrc has the correct path to the SetScript, ClearScript, and UpdateScript. These scripts are located in the Notifier directory where you extracted the tools\_bundle file.
16. On Windows platforms, if you want to use AlarmNotifier scripts, install Cygwin from <http://www.cygwin.com>. Be sure that the PATH variable contains the Cygwin bin directory in it.

You can now run the utilities on this computer.

**More information:**

[Pack Up CA Spectrum Utilities](#) (see page 23)

# Chapter 2: Setting Up a Distributed SpectroSERVER Environment

---

This section contains the following topics:

- [Name Resolution Requirements](#) (see page 29)
- [CA Spectrum and Multiple Interfaces](#) (see page 29)
- [Communication Among SpectroSERVERs](#) (see page 30)
- [Port Conflict Resolution](#) (see page 30)
- [DSS Environment Requirements](#) (see page 32)
- [Location Servers](#) (see page 32)
- [Landscape Handles](#) (see page 36)
- [Process Daemon \(processd\)](#) (see page 37)
- [Network Partitioning](#) (see page 47)
- [Duplicate Models in a Distributed Environment](#) (see page 48)
- [Host Resource Configuration File \(.hostrc\)](#) (see page 49)
- [Time Zones in a DSS Environment](#) (see page 49)
- [SpectroSERVER Shutdown](#) (see page 49)
- [Set the Landscape Map Entry Timeout](#) (see page 50)
- [Problems with Landscape Mapping from Existing DSS Setup](#) (see page 51)
- [Problems with Purging Old Landscapes](#) (see page 52)

## Name Resolution Requirements

The SpectroSERVER system or OneClick web server system must be able to resolve the host name of the SpectroSERVER to an IP address.

We recommend using hosts files for name resolution. This practice ensures that a network outage does not affect SpectroSERVER name resolution.

## CA Spectrum and Multiple Interfaces

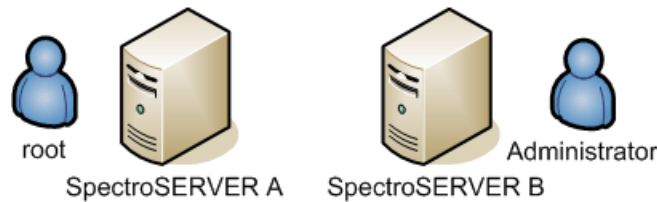
All CA Spectrum servers bind to and listen on all available interfaces and advertise themselves with host names that are not fully qualified. The result is flexibility when configuring a management topology. To establish connections, the CA Spectrum component servers try all interfaces in the order that is determined by the operating system until the connection succeeds.

## Communication Among SpectroSERVERs

Install and run SpectroSERVERs in a distributed environment with the same user account. No further user model configuration is therefore required. SpectroSERVERs that are installed and run as different users cannot communicate.

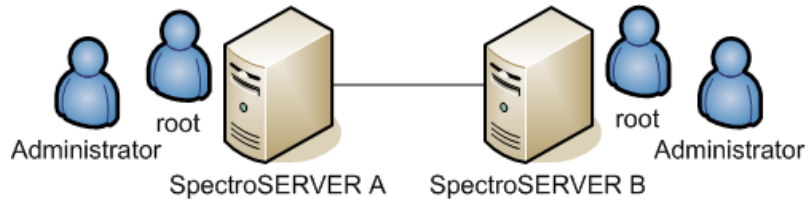
### Example: SpectroSERVERs A and B Cannot Communicate

In this example, SpectroSERVER A is running as "root". SpectroSERVER B is running as Administrator. They cannot establish a connection because the request cannot pass through server security:



### Example: Configuration for SpectroSERVER to SpectroSERVER Connection

This example shows a user logged in as "root" on SpectroSERVER B, and a user logged in as "Administrator" on SpectroSERVER A. This configuration enables the two SpectroSERVERs to communicate:



## Port Conflict Resolution

To avoid a port conflict with another application, you can change the default port numbers that CA Spectrum processes and services use.

The following table describes the default ports for CA Spectrum processes and services.

**Note:** "Not used for remote connections" is noted for ports in this table, where applicable. Ports that CA Spectrum does not use for remote connections typically do not present firewall configuration concerns.

Port (dec)	Port (hex)	CA Spectrum Components	Protocols	Notes
80	0x0050	OneClick/SRM	HTTP	None
162	0x00A2	SpectroSERVER	UDP	Port used to listen for SNMP traps.
3306	0x0CEA	mysql for SRM data ArchMgr (Archive Manager)	TCP/ODBC/JDBC	Used for remote connections only when migrating an SRM database from Windows to Linux or Solaris.
3307	0x0CEB	mysql for BOXI on Solaris/Linux	TCP	Not used for remote connections.
6844	0x1ABC	SDC	TCP	None
6400	0x1900	BOXI CMS	TCP	Not used for remote connections.
7777	0x1E61	CLI vnmsd	TCP	See <a href="#">CLI Daemon(vnmsd)</a> (see page 62).
14001	0x36B1	OneClick Server	TCP/CORBA	See <a href="#">OneClick WebServer Host</a> (see page 57).
14002	0x36B2	SpectroSERVER	TCP/CORBA	See <a href="#">SpectroSERVER</a> (see page 60).
14003	0x36B3	ArchMgr	TCP/CORBA	See <a href="#">Archive Manager</a> (see page 61).
14004	0x36B4	LocServ	TCP/CORBA	See <a href="#">Location Server</a> (see page 61).
14006	0x36B6	Naming service	TCP/CORBA	None
31415	0x7AB7	Telnet through SpectroSERVER		None
46517	0xB5B5	sradmin	TCP	Remote Administration Daemon. <b>Note:</b> For more information about the sradmin process, see the <i>Installation Guide</i> .
47870	0xBAFE	ArchMgr	TCP/SSAPI	See <a href="#">Archive Manager</a> (see page 61).
48879	0xBEEF	SpectroSERVER	TCP/SSAPI	See <a href="#">SpectroSERVER</a> (see page 60).

Port (dec)	Port (hex)	CA Spectrum Components	Protocols	Notes
51966	0xCAFE	rcpd	TCP	See <a href="#">Remote Copy Process Daemon (rcpd)</a> (see page 62).
56063	0xDAFF	LocServ	TCP	See <a href="#">Location Server</a> (see page 61).
61904	0xF1D0	processd	TCP	This port is not configurable.
64222	0xFADE	TL1d	TCP/EPI	TL1 gateway agent
65259	0xFEED	processd	TCP	This port is not configurable.

## DSS Environment Requirements

Your distributed environment must meet the following conditions to enable multiple SpectroSERVERs and the modeling of multiple landscapes to represent those servers:

- Each landscape must contain an identical modeling catalog of the model types in a database and their relations. This replication provides a consistent base for CA Spectrum intelligence.
- Each landscape must contain the same user models, which you can view in the OneClick Users tab.
- Minimize the number of connections between subnets that are modeled in different landscapes.
- Limit the number of identical devices that are modeled in more than one landscape.
- Assign a unique landscape handle to each modeled landscape. CA Spectrum can then distinguish it from other landscapes in the DSS environment.

### More information:

[Assign Landscape Handles](#) (see page 36)

## Location Servers

When you install a SpectroSERVER, you also automatically install a *location server*. This server identifies and locates other CA Spectrum services on the network. CA Spectrum processes use the location server to determine where these services are running. Processd starts and stops the location server.



In a distributed environment, CA Spectrum uses location servers to maintain the SpectroSERVER landscape map and provide connection services to client applications. During CA Spectrum installation, you designate one of the location servers in a landscape map (or CA Spectrum domain) as the *main location server (MLS)*.

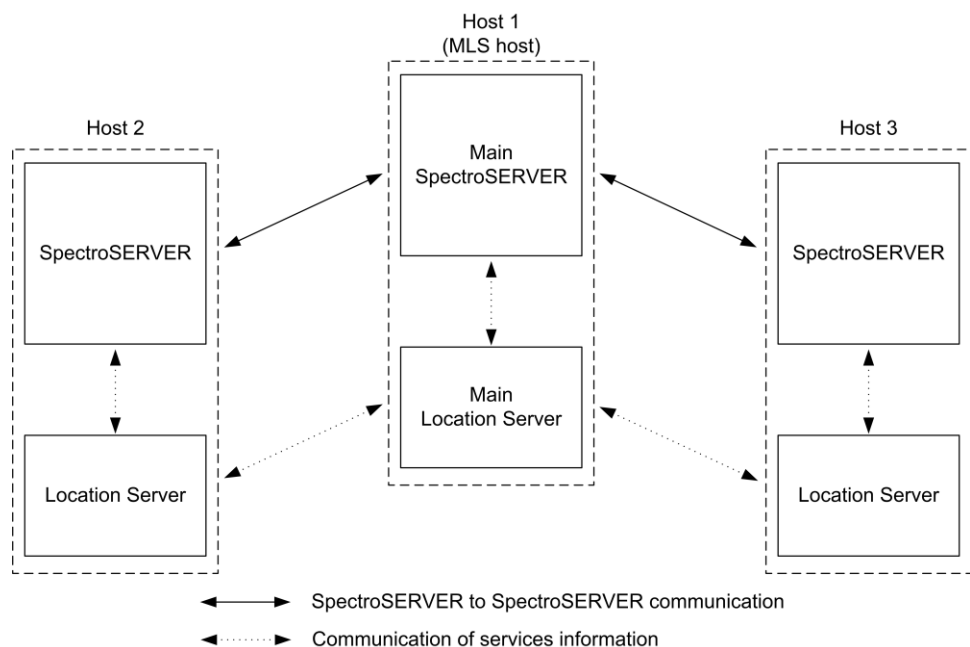
Install the main location server on a highly reliable computer. This server propagates service advertisements among location servers and communicates service locations among the location servers.

In the following diagram, the location server on Host 1 has been configured as the main location server for this environment. Because the main location server resides on the same host as a SpectroSERVER, that SpectroSERVER is considered to be the main SpectroSERVER.

All server-to-server communication is routed through the main location server host. Two non-MLS SpectroSERVERs can communicate only through the main SpectroSERVER.

The SpectroSERVER on Host 2 in the diagram has a distributed service with a resource modeled on the SpectroSERVER on Host 3. All communications about the remote resource are routed through the main location server host. The servers on Host 2 and Host 3 do not directly communicate.

**Important!** This diagram reflects a two-tiered setup consisting of one MLS and multiple child SpectroSERVERs. Such a configuration is the standard, recommended configuration. Although more complex configurations consisting of multiple MLSs with hierarchical relationships (multi-tiered deployments) are possible, they are not recommended. Multi-tiered deployments increase complexity without providing more functionality.



## How Location Servers Interact

Every CA Spectrum installation includes the following two files:

- .LocalRegFile lets CA Spectrum processes locate their location server.
- LS/.locrc is the configuration file for the locally installed location server.

These files manage the location server hierarchy and the interactions between each location server and the main location server.

In a distributed SpectroSERVER deployment, the location servers interact as follows:

1. SSAPI applications and servers read .LocalRegFile to determine the location server to use.
2. Each local location server reads the /LS/.locrc file to determine which server is the main location server.
3. The local location server connects to the main location server to find network services. If the main location server becomes unavailable, other location servers retain service information in memory until the main location server is available.

## Main Location Server Connection

Each SpectroSERVER must have a connection to the SpectroSERVER running on the server that is designated as the main location server. This designation is specified in the `<$SPECROOT>/LS/.locrc` file.

The connection to the main location server requires the following configuration:

- Each SpectroSERVER must let the other SpectroSERVER host connect to it. Connections require appropriate entries in the .hostrc files on each server.
- Each SpectroSERVER requires a CA Spectrum user model for the user account under which the other SpectroSERVER is running.

**Note:** The main landscape (or the value of MAIN\_LOCATION\_HOST\_NAME in the .locrc file) is modeled in CA Spectrum. The VNM topology for each SpectroSERVER contains this landscape unless it is already modeled elsewhere. An alarm is generated on this model if the connection to the main landscape is lost.

## Designate a New Main Location Server

You can change the computer that is designated as the main location server.

On the SpectroSERVER, you designate a new main location server through the Location Server Configuration dialog or through the .locrc file.

On the OneClick web server, you set the main location server information in the CA Spectrum Configuration page in the OneClick Administration pages. This action updates the name of the main location server to which the OneClick web server connects.

**Note:** If you change the main location server on the SpectroSERVER without updating the OneClick web server, CA Spectrum does not function properly. For more information, see the *Administrator Guide*.

### Example: Designating a New Main Location Server on the SpectroSERVER

A distributed network is composed of Computers 1 through 4, with Computer 1 designated as the main location server. To make Computer 4 the main location server, first reconfigure Computer 4 as the new main location server. Then reconfigure Computers 1, 2, and 3 to point to Computer 4 as the main location server.

#### Follow these steps:

1. Bring up the SpectroSERVER Control Panel.
2. Select Location Server from the Configure menu.  
The Location Server Configuration dialog opens.
3. Change the Main LS Host field in the Location Server Characteristics area to point to Computer 4.

**Note:** You can also edit the .locrc file in the <\$SPECROOT>/LS directory to change the main location server. The following entry in the .locrc file identifies the main location server:

```
MAIN_LOCATION_HOST_NAME=Computer 4
```

#### More information:

[Establish Fault Tolerance](#) (see page 72)

## Landscape Map Integrity Preservation

CA Spectrum users commonly maintain separate production and test environments. Protect the integrity of the landscape map by preventing SpectroSERVERs in the test environment from connecting to a SpectroSERVER in the production environment SpectroSERVER as their main location server, or the reverse.

## Landscape Handles

A SpectroSERVER in a distributed deployment identifies each landscape by its *landscape handle*. Each landscape must have a unique landscape handle, which is a number that is divisible by 4 and is in the range of 4 to 16,376. (In hexadecimal format, handles are in the range 0x100000 to 0xffe00000 with the lower 20 bits set to zero). The encoded landscape handle appears at the top of all views that are associated with that landscape.

**Note:** We recommend using a sequential numbering scheme for landscape handles. You can associate a landscape handle with a significant landscape feature, such as a building number or some portion of a subnet IP address. However, your entry is encoded into the 12 most significant bits of the landscape handle. The result can therefore appear unrelated to the appropriate landscape feature.

## Assign Landscape Handles

Landscape handles can be assigned through the SpectroSERVER Validation dialog when installing CA Spectrum or by using a utility named `lh_set`.

**Important!** Run the `lh_set` utility before you run the SpectroSERVER for the first time. Otherwise, CA Spectrum assigns a default landscape handle that is the same every time that CA Spectrum assigns it. As a result, duplicate landscape handles can be created when multiple landscapes are configured. Such landscapes can never be accessed simultaneously from the same application.

### Follow these steps:

1. Navigate to the `SS` directory.
2. Enter the following command:

```
../SS-Tools/lh_set <landscape handle>
```

You can specify the new landscape handle in either decimal or hexadecimal notation. If you use decimal notation, the `lh_set` utility converts your entry into a hexadecimal landscape handle.

**Note:** For more information, see the *Installation Guide*.

**More information:**

[DSS Environment Requirements](#) (see page 32)

## Change a Landscape Handle

If a SpectroSERVER has started, the process for changing the landscape handle of the SpectroSERVER database entails more steps. The landscape handle is included in the model handle of each model in the SpectroSERVER database. Changing the landscape handle therefore requires converting all model handles from the old landscape handle to the new landscape handle.

Change the landscape handle of all models that were created automatically at startup, and update the landscape handle in several resource files.

**Follow these steps:**

1. Use the Modeling Gateway tool to export the models from the SpectroSERVER whose landscape handle you want to change.
2. Shut down the SpectroSERVER.
3. Initialize the database with the SSdbload utility.  
**Note:** For more information, see the *Database Management Guide*.
4. Assign the new landscape handle using the lh\_set utility.
5. Start the SpectroSERVER.
6. Import the models into the SpectroSERVER using the Modeling Gateway tool.

**Note:** For more information, see the Modeling Gateway Toolkit Guide.

## Process Daemon (processd)

CA Spectrum uses a process launching and tracking daemon called Process Daemon (processd) to let you control processes on multiple servers in a DSS environment. This daemon starts processes when an application such as the CA Spectrum Control Panel makes a request. You can also configure processd with install tickets to start processes automatically at system startup. Install tickets can also automatically restart critical processes that stop unexpectedly.

The CA Spectrum installation program configures processd to start automatically on Solaris systems (where it must run as root) and on Windows systems (where it runs as the LocalSystem account).

Process launch and monitoring with processd occur only when an application requests such actions. Typically, processd operates in the background.

If a process does not start or is not working properly, processd writes an error message to the <\$SPECROOT>\lib\SDPM\processd\_log file. The error message includes information to identify the problem.

When restarted, processd creates a processd\_log.bak file to preserve old error messages and appends any new error messages to the processd\_log file.

**Important!** You must be thoroughly familiar with CA Spectrum, distributed networking, and network configuration before attempting any of the procedures described in this section.

**More information:**

[Install Tickets Files](#) (see page 40)

## Processd Differences in Windows and Solaris Environments

The processd daemon works slightly differently on Windows and Solaris platforms because of differences in their native security architectures.

- When you request a process start from a remote connection in a Windows environment, the process starts as the user who is specified in the Windows Service Configuration dialog. On Solaris, the process starts as the user making the request.
- For server processes that must remain running after user logout (or before anyone logs in), a SERVERPROCESS field is used in the Windows environment. To prevent Windows from shutting these processes down during logout, these processes are started as the user who is specified in the Windows Service Configuration dialog. On Solaris, the processes are started as the TICKETUSER that is specified in the install ticket.

**More information:**

[Install Tickets Files](#) (see page 40)

## Change the Windows Password in Processd

During installation on Windows, you are prompted to enter a username and password in the Windows Service Configuration dialog. The processd daemon uses this username and password to start CA Spectrum processes as the specified user. Providing the security information lets these processes run when no user is logged in.

**Note:** For more information, see the *Installation Guide*.

If the password you specified in the Windows Service Configuration dialog changes or expires, you can update processd with the new password.

**Important!** If your password contains special characters, such as an exclamation point (!), escape these characters with a backslash (\). Or you can change the password from a command prompt.

**Follow these steps:**

1. Log in as a member of the local Administrators group.

You must be a member of the CA Spectrum Users group and an administrator to change the processd user name and password.

2. Open a command prompt.
3. Navigate to the /lib/SDPM directory.
4. Enter the following command:

```
processd --install --username user --password newpassword
```

***user***

Specifies the user name.

***newpassword***

Specifies the new password for this user name.

If the user name or the password contains special characters that the shell can misinterpret, enclose the user name or password in quotation marks. For example, the user name contains a domain name and user, with a backslash (\) separator. The password contains an asterisk (\*). Both of these characters are special characters. Therefore, the user name and password are enclosed in quotation marks, as in the following example:

```
processd --install --username "DOMAIN\JSmith" --password "283EJ*"
```

**Note:** If the password for the Windows user name changes but the processd service password is not updated, processd does not start. The following events are generated in the Windows event log:

- Logon attempt with current password failed with the following error:  
Logon failure: unknown user name or bad password.
- The CA Spectrum Process Daemon service failed to start due to the following error:  
The service did not start due to a logon failure.

## Install Tickets Files

You can configure processd to start processes at system startup. You can also select processes to restart if they stop running. Files that are known as *install tickets* support this functionality.

Install tickets use files in the following two directories:

- *<Spectrum Installation Directory>/lib/SDPM/partslist*
- *<Spectrum Installation Directory>/lib/SDPM/runtime*

The partslist directory contains the individual install ticket files.

The runtime directory contains encoded files in the form of *<PID>.rt* where *<PID>* is the process ID of the running process.

**Note:** SDPM stands for CA Spectrum Distributed Process Manager.

You can add install tickets to the partslist directory. New install tickets must conform to certain formatting rules. Restart processd to identify any new or modified install tickets in the partslist directory. These files are read at processd startup and are stored in cache memory for future use.

Install ticket files follow a naming convention that refers to the process whose configuration information it contains. Install ticket filenames are in the form of *<PARTNAME>.idb*. The *<PARTNAME>* variable is an internal key to identify processes that processd controls.

The following defining fields are used for install tickets. The format is *<fieldname>;<value>*; where *<fieldname>* is one of the names that are displayed below. The *<value>* is a string that provides a definition for the corresponding field name. Quotation marks are not supported.

### PARTNAME

Identifies a particular process/application with a multicharacter string with no spaces. The install ticket for this application has a filename in the form *<PARTNAME>.idb*. *<PARTNAME>* is the process name.



**APPNAME**

Defines the name of the application as a multicharacter string.

**WORKPATH**

Specifies where you want to run the application. Supply a value for this field *before* it can be used as part of the ARGV field that is described later.

**LOGNAMEPATH**

Specifies the log file for output from the application. This field must begin with `<$SPECROOT>` or `<$WORKPATH>`. No spaces are allowed.

**ADMINPRIVS**

Is reserved for use in Purism-supplied install tickets only. Comment it out in install tickets that you create.

**AUTORESTART**

Indicates whether a process is automatically restarted if it stops. If this field is not included in the install ticket, automatic restart is disabled by default. Accepted values are Y, y, N, n.

**AUTOBOOTSTART**

Indicates whether the process is started whenever processd starts. If this field is not included in the install ticket, the function is disabled by default. Accepted values are Y, y, N, n.

**STATEBASED**

Indicates whether the process has more than one state when starting up. Usually, the process sends an “is ready” ticket when ready to communicate. This field is reserved for use in Purism-supplied install tickets only. If this field is not included in the install ticket, it is disabled by default. Accepted values are Y, y, N, n.

**NUMPROCS**

Specifies the number of instances of a process that can run on one platform. Numeric values are accepted.

**RETRYTIMEOUT**

Specifies the number of seconds that processd tries to restart the application after failure.

**TICKETUSER**

Defines the username of the user who is authorized to run the process when the AUTOBOOTSTART and/or AUTORESTART fields are set. This field is only required when those fields are included in the install ticket. This field is not applicable on Windows because all processes run as the processd install user.

#### **RETRYMAX**

Specifies the number of times that processd tries to restart an application within the specified RETRYTIMEOUT period.

#### **STARTPRIORITY**

Indicates the relative startup priority of the process relative to other processes. The following values are used:

- 10 for standalone processes
- 20 for processes that depend on standalone processes
- 30 for processes that depend on the SpectroSERVER

#### **SERVERPROCESS**

Indicates to processd whether a process continues to run after the user logs out. When this field is enabled, the process is always started as the user who is authorized to run the processd service. This field is only applicable in the Windows environment. The default value is 'yes' for the following processes:

- Archive Manager (ARCHMGR.idb)
- Location Server (LOCSERV.idb)
- SpectroSERVER (SS.idb)

#### **SERVICE**

Indicates whether the ticket represents a Windows service. Lets processd manage Windows services from the Service Control Manager.

#### **ENV**

Adds one or more variables to the application environment. Multiple values are listed on a separate line with the macro <CSPATHSEP> between the value and the semicolon that ends the line.

#### **ARGV**

Defines the argument list of the process, which includes the executable path and any number of arguments (spaces are allowed).

#### **More information:**

[Process Daemon \(processd\)](#) (see page 37)

[Processd Differences in Windows and Solaris Environments](#) (see page 38)

## **Examples: Install Tickets**

Install tickets apply the following syntax conventions:

- Any line beginning with a pound sign (#) is treated as a comment.

- A semicolon (;) must follow all field names and all values following each field name.
- Anything following the second semicolon is disregarded.
- Four macros can be used: <CSEXE>, <CSBAT>, <CSCMD>, and <CSPATHSEP>. Based on the platform, the appropriate definition is substituted. Use <CSPATHSEP> instead of the actual path separator to avoid parsing conflicts.

The following example shows a valid install ticket:

```
# Processd Install Ticket for SpectroSERVER Daemon.  
PARTNAME;SS;  
APPNAME;SpectroSERVER Daemon;  
WORKPATH;$SPECROOT/SS;  
LOGNAMEPATH;$WORKPATH/VNM.OUT;  
ADMINPRIVS;y;  
#AUTORESTART;N;  
#AUTOBOOTSTART;N;  
STATEBASED;y;  
NUMPROCS;3; // unlimited  
RETRYTIMEOUT;0; // seconds  
#TICKETUSER;$USER;  
RETRYMAX;0; // retries  
STARTPRIORITY;20;  
SERVERPROCESS;Y;  
#ENV;<var>=<value>;  
ARGV;$SPECROOT/SS/SpectroSERVER<CSEXE>;
```

The following example shows another valid install ticket:

```
# Processd Install Ticket for Visibroker Naming Service
PARTNAME;NAMINGSERVICE;
APPNAME;Visibroker Naming Service;
WORKPATH;$SPECROOT/bin/VBNS;
LOGNAMEPATH;$WORKPATH/NAMINGSERVICE.OUT;
ADMINPRIVS;y;
AUTORESTART;y;
AUTOBOOTSTART;y;
#STATEBASED;N;
NUMPROCS;1; // one per host
RETRYTIMEOUT;600; // 10 minutes
#TICKETUSER;$USER;
RETRYMAX;5; // 5 retries
STARTPRIORITY;10;
SERVERPROCESS;Y;
#ENV;<var>=<value>;
ENV;CLASSPATH=$SPECROOT/lib/vbjorb.jar<CSPATHSEP>;
ENV;CLASSPATH=$SPECROOT/lib/vbsec.jar<CSPATHSEP>;
ENV;CLASSPATH=$SPECROOT/lib/lm.jar<CSPATHSEP>;
ENV;CLASSPATH=$SPECROOT/lib/sanct6.jar<CSPATHSEP>;
ARGV;
$SPECROOT/bin/JavaApps/bin/nameserv<CSEX>
-DORBpropStorage=
$SPECROOT/.corbarc
-Dvbroker.orb.admDir=
$SPECROOT/bin/VBNS
-Dborland.enterprise.licenseDir=
$SPECROOT/bin/VBNS/license
-Dborland.enterprise.licenseDefaultDir=
$SPECROOT/bin/VBNS/license
-Djava.endorsed.dirs=
$SPECROOT/lib/endorsed
-Dorg.omg.CORBA.ORBClass=
com.inprise.vbroker.orb.ORB
-Dorg.omg.CORBA.ORBSingletonClass=
com.inprise.vbroker.orb.ORB
com.inprise.vbroker.naming.ExtFactory;
```

## Start a New Install Ticket Process

If you added an install ticket file, you can use the restart option to start the process specified in the ticket. With the restart option, you do not have to stop and restart processd and all of the processes that it is monitoring.

### Follow these steps on Solaris:

1. Log in as root.
2. Navigate to the /lib/SDPM directory.
3. Enter the following command:  

```
processd.pl restart
```

The process is started.

### Follow these steps on Windows:

1. Log in as a member of the CA Spectrum Users group.
2. Open a command prompt.
3. Navigate to the /lib/SDPM directory.
4. Enter the following command:  

```
perl processd.pl restart
```

The process is started.

## Stop and Restart Processd

If you suspect that the lib/SDPM/runtime directory has become corrupted, stop and restart processd.

### Follow these steps: on Solaris

1. Log in as root.
2. Navigate to the /lib/SDPM directory.
3. Enter the following command:  

```
processd.pl stop
```
4. Verify that all CA Spectrum processes are shut down (otherwise problems with the SpectroSERVER can occur).
5. Remove all entries in the /lib/SDPM/runtime directory.
6. Restart processd by entering the following command:  

```
processd.pl start
```

**Follow these steps: on Windows**

1. Log in as a member of the CA Spectrum Users group.
2. Open a command prompt.
3. Navigate to the /lib/SDPM directory.
4. Enter the following command:  

```
perl processd.pl stop (or start)
```

**Note:** On Windows, the processd.pl <start/stop> commands also stop and start the CA Spectrum processes that run as NT services (such as MySQL and VisiBroker).

**More information:**

[SpectroSERVER Shutdown in a Solaris Environment](#) (see page 50)

## How the Userconf Process Works During Installation

The userconf process runs processd in the background. Userconf lets you perform user-specific configuration of CA Spectrum during installation and afterwards, when a user logs in to CA Spectrum.

The userconf process performs the following tasks during CA Spectrum installation:

1. Runs userconf -install %SPECROOT%, which makes the following changes:
  - Adds SPECROOT and SPECPATH to the system environment.
  - Adds \$SPECPATH\lib to the \$PATH system variable.
  - Removes the old \$SPECPATH\lib entries from the path, if the installation directory has changed.
  - Removes %SPECPATH%\lib from the path (if it exists).
  - Modifies the registry so that userconf runs each time that a user logs in (with no arguments).
2. Runs userconf -start to start processd. The -start flag starts processd without verifying that the user is a member of the CA Spectrum Users Group.

**Note:** The installation program adds the current user to the CA Spectrum Users Group. However, the change does not take effect until the user logs out and logs back in. Add the -start flag so that processd starts without verifying that the user is a member of the CA Spectrum Users Group. The installation then continues without requiring the user to log out and log back in.
3. Runs userconf -restart to stop and restart processd if the user is installing Exceed for the first time (which occurs after the CA Spectrum installation). Processd can then update PATH environment changes that are part of the Exceed installation.

## How Userconf Works When a User Logs In

The installation adds the userconf process to the registry Run key. When a user logs in, the process starts automatically and checks whether the user is a member of the CA Spectrum Users group. The userconf process applies the following rules:

- If the user is a member of the CA Spectrum Users group, userconf verifies cygwin and Exceed and reconfigures as needed before starting processd. If Microsoft VC++ is installed, userconf configures it appropriately for use with the CA Spectrum SDK.
- If the user is not a member of the Users group, a message states that CA Spectrum is installed but the user is not in the CA Spectrum Users group.

To enable the user to use CA Spectrum, add the user account to the CA Spectrum Users group. The user must then log out and log back in.

A user who does not want to use CA Spectrum can select the “user doesn't want to run Spectrum” check box. With this option, userconf continues to run when the user logs in, but the process exits silently.

**Note:** To see the message again, run the following command:

```
userconf -install %SPECROOT%
```

## Network Partitioning

Network partitioning is not significant when you are modeling a network that is already configured. However, if you are creating multiple landscapes and you want to use Discovery to create your Topology hierarchy, use methods that support Discovery.

Your objectives are to minimize the duplication of models in your landscapes and the number of connections between landscapes. To facilitate fault isolation, duplicate some models, such as a router that connects two IP subnets.

You have the following options for network partitioning:

- Create partitions that are subnets, which are defined by IP address boundaries. IP address ranges provide the most convenient means of restricting Discovery exploration.
- If IP address boundaries do not provide a way to separate landscapes, use unique community names to differentiate devices in each landscape. This method requires more effort. Add these names to the community names table of each device.

- Create partitions according to another parameter (such as the time zone or state) that contains multiple IP subnet addresses. You must model your network manually in most cases. However, maintain as few connections between landscapes as possible and observe modeling rules. Excessive connections can compromise the advantages of DSS.

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

## Duplicate Models in a Distributed Environment

In a DSS environment, CA Spectrum tracks duplicate models (for example, a user model that exists on multiple landscapes) by designating one as a "home" model. The home model resides on the SpectroSERVER that is running on the host with the "root" main location server (MLS).

**Important!** If you change the root MLS in a DSS environment, the state of all duplicate models is updated to match the "home" model on the MLS. The MLS is the final authority in any DSS environment.

CA Spectrum uses the home model to synchronize information for all duplicate models. Modification requests that are made to the duplicate models that are not the home models are relayed to the landscape of the home model. The home model then distributes the request to all of the other duplicate models.

## Failure to Contact Home Landscape Errors

Editing operations can fail when the home landscape of the model that you are editing cannot be contacted. OneClick reports relation errors such as the following error from the SpectroSERVER:

The operation failed because the model being edited exists in multiple landscapes and its home landscape could not be contacted.

This type of error can indicate that the home model on the SpectroSERVER that is running on the "root" MLS is unreachable. When duplicate models are added or deleted in a distributed environment, they are routed through the home landscape. If that SpectroSERVER is down or cannot be contacted, the relation fails and generates an error.



Failure to contact the home landscape can occur under any of these conditions:

- The home SpectroSERVER or an intermediate SpectroSERVER between the duplicate model and the home SpectroSERVER is not running.
- A network problem is preventing SpectroSERVER access.
- A security issue is preventing the SpectroSERVERs from communicating.

## Host Resource Configuration File (.hostrc)

The .hostrc file restricts client application access to each local SpectroSERVER. Client applications cannot connect to the local landscape unless the .hostrc file is configured to permit this access. You can use a text editor to edit the .hostrc file.

**Note:** For more information, see the *Administrator Guide*.

By default, the .hostrc file is initially installed with the local hostname, which restricts all remote access. Adding an individual hostname enables connections to and from that server. Enable unrestricted access to and from all remote servers by adding a "+" symbol. Add the machine name to the .hostrc file to enable client access from individual remote servers.

CA Spectrum implements host security in multiple layers, by hostname or by IP address. We recommend listing hostnames in the .hostrc file, which includes all IP addresses that are associated with the hostnames. Connection attempts that are initiated by the hostname do not always succeed if an IP address is used to connect.

## Time Zones in a DSS Environment

In DSS installations that span multiple time zones, all activities that occur on a SpectroSERVER reflect the local time of the server, including scheduled events. All schedules created in OneClick and applied to modeled devices start and end according to the local time of the SpectroSERVER or device landscape.

**Note:** For more information, see the *Operator Guide*.

## SpectroSERVER Shutdown

We recommend shutting down the SpectroSERVER using the CA Spectrum Control Panel before you shut down the server where the SpectroSERVER is running. However, if you use the operating system shutdown procedure as a way to shut down the SpectroSERVER, increase the amount of time for processd to stop.

## SpectroSERVER Shutdown in a Solaris Environment

In a Solaris environment, by default, `processd` waits 20 seconds for all subprocesses to shut down before it stops. If the SpectroSERVER takes a long time to shut down, `processd` waits for a longer time before stopping. Use the `PROCESSD_SHUTDOWN_TIMEOUT` environment variable to change the default value. Add this variable and its appropriate value (in milliseconds) to the `<$SPECROOT>/spectrum60.env` file.

For example, to make `processd` wait 60 seconds for all subprocesses to shut down, add the following line to the `spectrum60.env` file:

```
PROCESSD_SHUTDOWN_TIMEOUT=60000
```

Then [stop and restart processd](#) (see page 45) for the changes to take effect.

## SpectroSERVER Shutdown in a Windows Environment

In a Windows environment, `processd` waits until all subprocesses have shut down before it stops. However, the Windows registry setting `WaitToKillServiceTimeout` sets the length of time that Windows waits for all services to stop after a Windows shutdown is initiated. If a service such as `processd` is still running after the `WaitToKillServiceTimeout` elapses, Windows terminates this service. The default value is 20 seconds, which is not always enough time for the SpectroSERVER to shut down completely at system shutdown. You can increase the timeout value.

### Follow these steps:

1. Open the Registry Editor.
2. Go to `HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Control`.
3. Click the Control key.
4. Double-click the `WaitToKillServiceTimeout` value in the right pane.
5. In the window, change the value to anything up to 600,000 milliseconds (10 minutes).
6. Click OK.
7. Restart the Windows server for the change to take effect.

## Set the Landscape Map Entry Timeout

By default, each landscape entry remains in the landscape map indefinitely. Using the location server configuration file (`.locrc`), you can specify an amount of time after which the landscape entry times out and is removed automatically from the landscape map.

**Note:** In previous CA Spectrum releases, landscape entries timed out after one hour, and they were removed automatically from the landscape map. Starting in CA Spectrum 9.2.2, landscape entries do not time out by default.

**Important!** We do *not* recommend using a timeout value to automatically remove an entry from a landscape map. Instead, leave the default timeout value. Use the `MapUpdate` command to remove an entry from the landscape map when required. For more information, see the *Database Management User Guide*.

**Follow these steps:**

1. Navigate to the `<$SPECROOT>/LS` directory.
2. Update or add the following entry in the `.locrc` file:

`MET_INTERVAL=time_out_value`

***time\_out\_value***

Indicates the time in milliseconds after which a landscape entry times out and is removed automatically from the landscape map.

**Default:** 0 milliseconds (no time-out)

## Problems with Landscape Mapping from Existing DSS Setup

**Symptom:**

I experienced problems with the landscape map when I set up a separate distributed environment by copying from another installation. I noticed a failure to switch over to the secondary SpectroSERVER after the primary goes down and problems with user security.

**Solution:**

To copy a landscape map, remove all references to the previous DSS setup. The landscape maps must contain no references to the previous DSS setup. Use the following procedure to map a landscape from an existing DSS setup.

**Follow these steps:**

1. Decouple the SpectroSERVERs as follows:
  - a. Make each SpectroSERVER a standalone server.

The location server is now pointing to the local server where it is installed.
  - b. Verify that the landscape map is distinct for each SpectroSERVER.

**Note:** In the existing DSS setup, the landscape map was a single map that included all of the servers.
2. Save the SpectroSERVER databases.

The landscape maps are clean.

3. Change the MLS of these SpectroSERVER to restore the original MLS.  
**Note:** Do not change the original DSS settings.
4. Reload the databases of each SpectroSERVER on different hosts.
5. Select a SpectroSERVER as your MLS in the new DSS setup, and point other SpectroSERVERs to it.

The dupModelList in the user models updates properly.

6. Remove secondary entries in the landscape map, if they exist.
7. Set up secondary SpectroSERVER entries.
8. Load the database from the primary SpectroSERVER in the new DSS.

The landscape is mapped from the existing DSS setup.

**More information:**

[Setting Up a Distributed SpectroSERVER Environment](#) (see page 29)

## Problems with Purging Old Landscapes

**Symptom:**

Our production CA Spectrum environment consisted of a single distributed installation. The environment had multiple SpectroSERVERs and OneClick servers, which all used the same Main Location Server.

We decided to remove two SpectroSERVERs and a OneClick server from the production environment to create a separate development environment. We configured the development SpectroSERVERs and OneClick server to reference a new Location Server. We updated the .hostrc files so that the servers in each environment only had access to their respective Location Servers. In addition, we updated the landscape map in the production environment to remove the development landscapes.

However, if I look at the list of Monitored Landscapes on development OneClick server, I still see all of the old production landscapes. They are listed as having "No permission" because of the changes to the .hostrc file.

**Solution:**

With the setup that you describe, you now have two separate environments. However, your development landscape map is caching stale production landscapes. You removed stale landscapes from your production environment, but you did not remove them from your development environment. Therefore, you must manually remove the stale landscapes from the development environment.

You have two options for removing the stale landscapes.

**Solution:**

You can use the following command to remove stale landscapes from the development landscape map:

```
MapUpdate -remove landscape handle
```

**Solution:**

You can restart all SpectroSERVER processes. For more information, see [Stop and Restart Processd](#) (see page 45). Then restart the OneClick server. For more information about OneClick server administration, see the Administrator Guide.

When you restart the servers, the issue is resolved.



# Chapter 3: Communication Across Firewalls in the Distributed SpectroSERVER Environment

---

This section contains the following topics:

[Communication Across Firewalls](#) (see page 55)

[SpectroSERVER and OneClick Web Server Communication Across Firewalls](#) (see page 56)

[OneClick Default Ports and Firewalls](#) (see page 57)

[Remote SpectroSERVERs and Firewalls](#) (see page 57)

[Primary and Secondary SpectroSERVER Communication Across Firewalls](#) (see page 59)

[CA Spectrum Configuration Files for NAT Firewall Environments](#) (see page 60)

[Default Port Configurations](#) (see page 60)

## Communication Across Firewalls

Communicating across a firewall can apply in many network environments. In a distributed environment, firewalls are likely to affect your deployment.

**Note:** Before you begin a distributed CA Spectrum installation, configure the firewall to enable traffic on port 46517 (the port sradmin).

Your options for enabling communications among CA Spectrum components depend on the types of firewalls that you deploy. Other factors, such as cost and the level of security that is required, also play a role. These factors include Virtual Private Networks (VPNs), node-to-node tunnels or conduits, and proxies that encapsulate packets before they pass through the firewall.

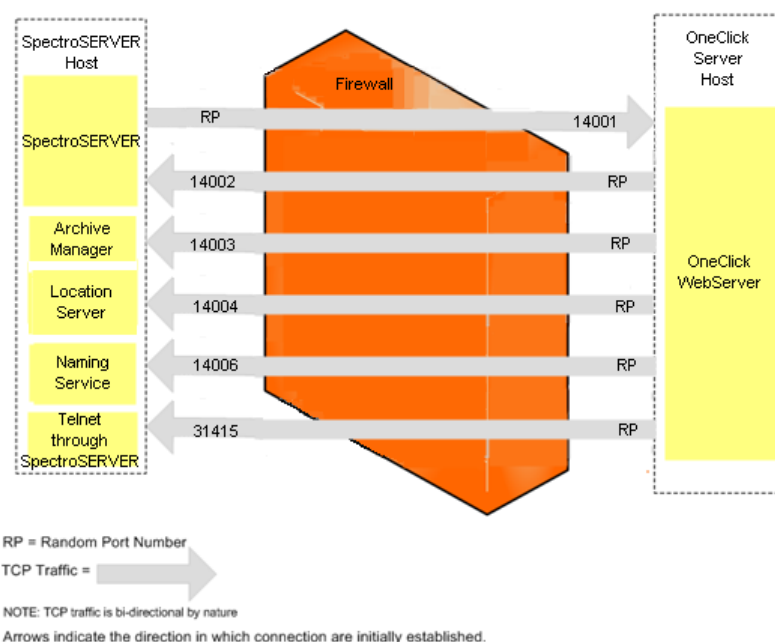
We recommend working with the firewall administrator for your network to work out a strategy for letting CA Spectrum and OneClick traffic traverse firewalls.

## SpectroSERVER and OneClick Web Server Communication Across Firewalls

The OneClick web server communicates with processes on the SpectroSERVER host system to gather data to display in OneClick clients. The OneClick web server typically initiates this communication.

The OneClick web server establishes connections to specific SpectroSERVER host-side TCP ports. The web server uses these ports for sending requests and receiving responses. However, OneClick uses a single listening port (default 14001). The SpectroSERVER initiates the connection to that port. As a result, modifying firewall configuration is often necessary. The SpectroSERVER uses bidirectional IOP (Internet Inter-ORB protocol) to communicate with its CORBA clients.

The following diagram illustrates the IP connectivity that is required for a OneClick web server to communicate with a SpectroSERVER. In all cases where TCP is used, the connections are established from a random port to a specific, fixed port.



**Note:** In a Fault-Tolerant configuration, the same ports must be opened between the OneClick Server Host and the secondary SpectroSERVER Host, including port 14003 if running a secondary Archive Manager.



## OneClick Default Ports and Firewalls

### HTTP Listen Port

The default port used by OneClick for HTTP communication is port 80. If you configure the OneClick web server to use something other than port 80, your firewall must also be set up to allow this traffic.

**Note:** For more information, see the *Administrator Guide* and the *Installation Guide*.

OneClick users on the Windows XP SP2 platform who choose to leave the Windows Firewall enabled may have problems running the OneClick Console.

**Note:** For more information on configuring the Windows firewall, refer to the Microsoft Knowledge Base article 842242 at <http://support.microsoft.com>.

### CORBA Listen Port

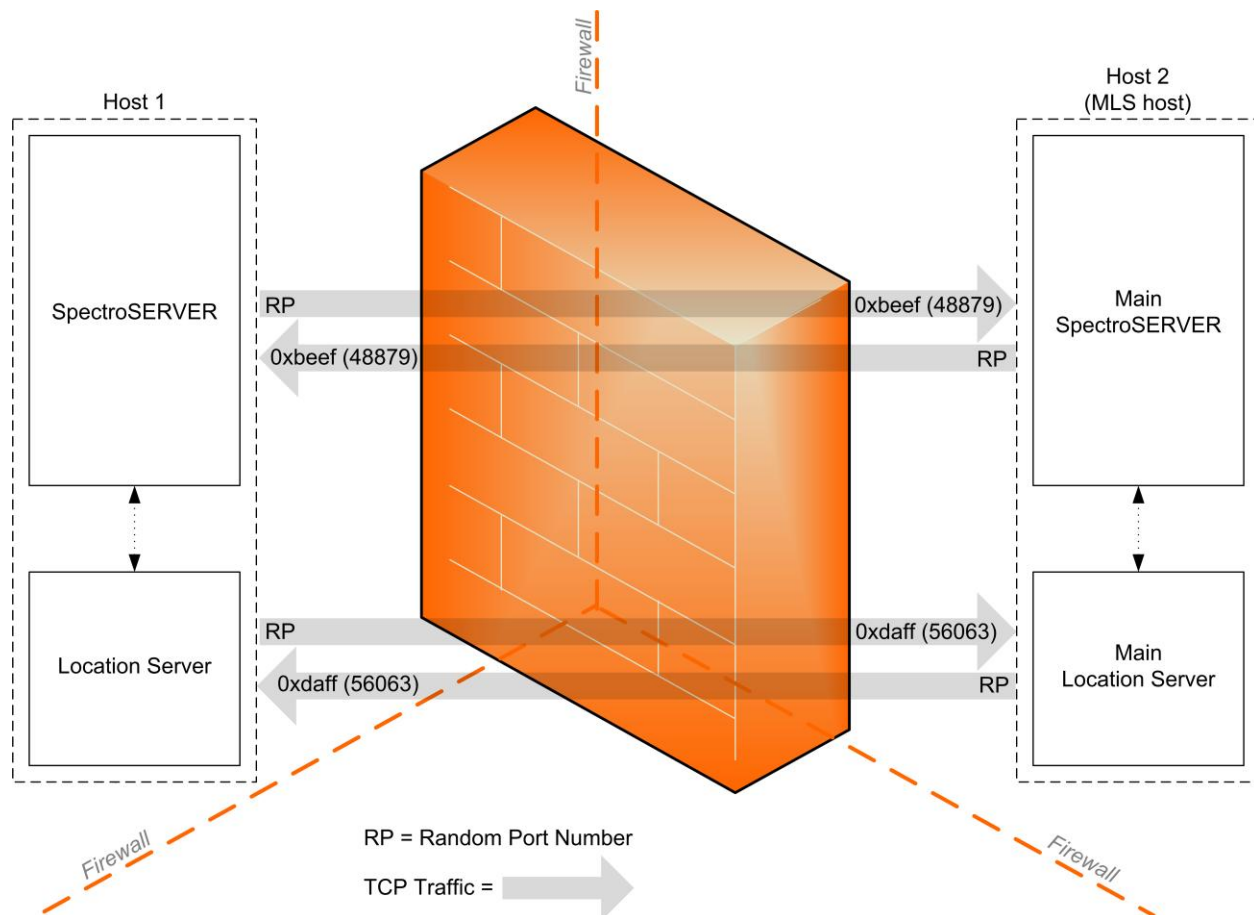
The Default CORBA OneClick server listen port value is located in `<${SPECROOT}>/tomcat/webapps/spectrum/META-INF/context.xml`.

```
vbroker.se.iiop_tp.scm.iioptp.listener.port=<new port number>
```

## Remote SpectroSERVERs and Firewalls

The following figure illustrates the IP connectivity that is required when two remote SpectroSERVERs communicate through a firewall. In all cases where TCP is used, the connections are established from a random port to a specific, fixed port.

**Important!** All communication among SpectroSERVERs that are not Main Location Servers (MLSs) is routed through the MLS-SpectroSERVER. As a result, each SpectroSERVER only communicates directly with the MLS-SpectroSERVER.

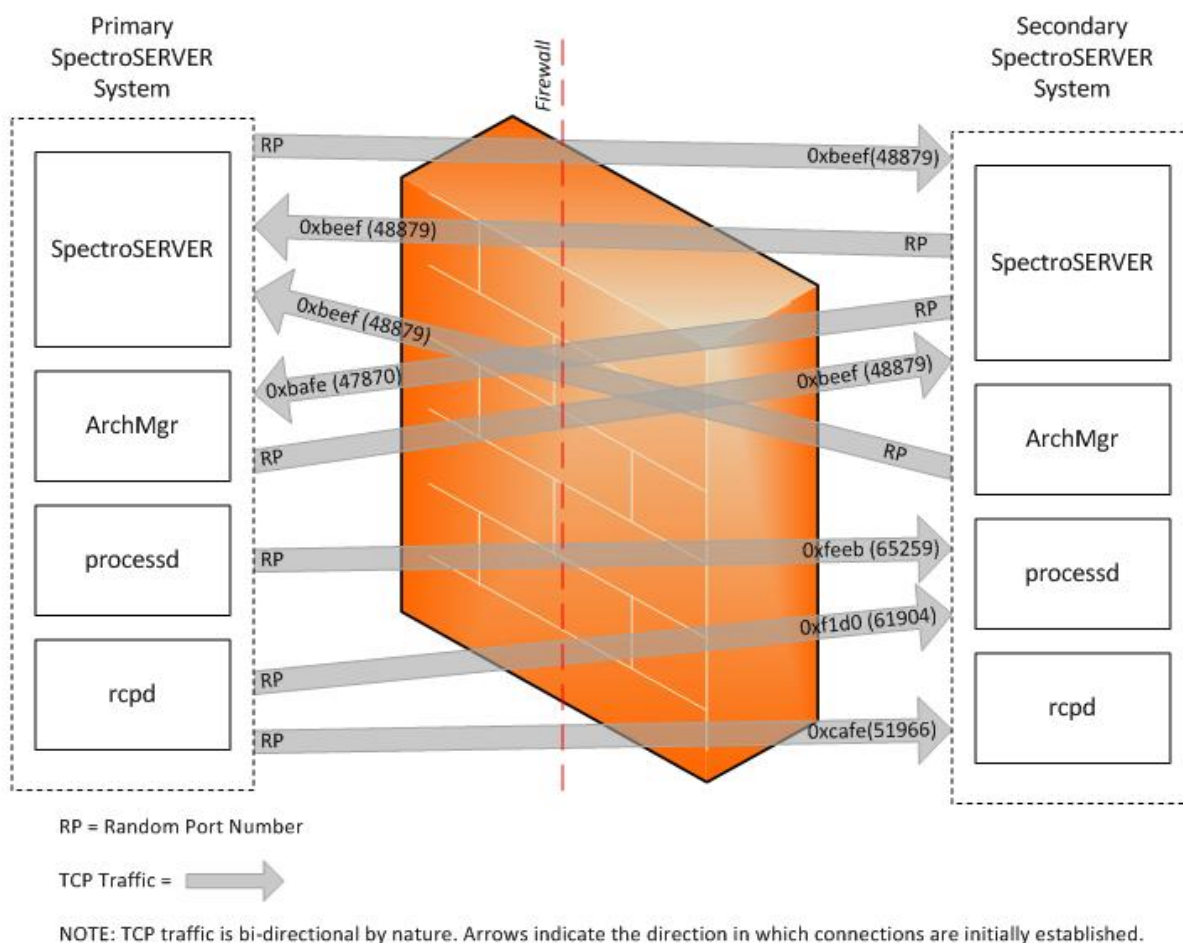


**More information:**

[Location Servers](#) (see page 32)

## Primary and Secondary SpectroSERVER Communication Across Firewalls

The following figure illustrates the IP connectivity that is required when primary and secondary SpectroSERVERs in a fault-tolerant environment communicate through a firewall. In all cases where TCP is used, the connections are established from a random port to a specific, fixed port.



## CA Spectrum Configuration Files for NAT Firewall Environments

Network Address Translation (NAT) is used to construct networks with consistent internal IP addressing and a single node that handles IP address translation to the Internet. CA Spectrum supports NAT. You can deploy CA Spectrum in a private IP address domain and can maintain connectivity to clients outside of NAT firewalls. Therefore, DSS environments can now encompass multiple domains that are separated by these firewalls.

The only requirement for a NAT environment is the ability of clients to resolve the server by name. On the private side of the NAT, the host name must resolve to the private-side IP address. On the public side of the NAT, the host name must resolve to the public-side IP address.

## Default Port Configurations

You can change the default port or socket number that a CA Spectrum process uses so that the process works correctly in a firewall environment.

You can change the port number or socket number for the following processes:

- [OneClick WebServer](#) (see page 57)
- [SpectroSERVER](#) (see page 60)
- [Archive Manager](#) (see page 61)
- [Location Server](#) (see page 61)
- [Naming Service](#) (see page 62)
- [Remote Copy Process Daemon \(rcpd\)](#) (see page 62)
- [CLI Daemon \(vnmsd\)](#) (see page 62)

**Note:** The only ports that you cannot change are the ports for the Remote Administration Daemon, sradmin, and the port for Telnet through SpectroSERVER.

## Change the SpectroSERVER Port Number

You can change the port number that the SpectroSERVER uses for CORBA requests.

Use a text editor to edit the .vnmrc file to reflect the new port number. This file is located in <\$SPECROOT>/SS/. Enter the following command:

```
orb_args=-Dvbroker.se.iiop_tp.scm.iiop_tp.listener.port=<new port number>
```

**More information:**

[Port Conflict Resolution](#) (see page 30)

## Change the Archive Manager Port Number and Socket Number

You can change the port number and the socket number that the Archive Manager uses for specific requests.

Use a text editor to edit the .configrc file to reflect the new port number. This file is located in <\$SPECROOT>/SS/DDM. Enter the following command:

```
orb_args=-Dvbroker.se.iiop_tp.scm.iiop_tp.listener.port=<new port number>
```

To change the socket number that the Archive Manager uses to listen for requests from VNM and SSAPI clients, use a text editor to edit the .configrc file. This file is located in <\$SPECROOT>/SS/DDM. Change the following variable:

```
ARCH_MGR_SOCKET_NUMBER=<new port number>
```

**More information:**

[Port Conflict Resolution](#) (see page 30)

## Change the Location Server Port Number and Socket Number

You can change the port number and the socket number that the Location Server uses for specific requests.

To change the port number the Location Server uses for CORBA requests, use a text editor to edit the .locrc file to reflect the new port number. This file is located in <\$SPECROOT>/LS. Enter the following command:

```
orb_args=-Dvbroker.se.iiop_tp.scm.iiop_tp.listener.port=<new port number>
```

To change the socket number that the Location Server uses for VNM and SSAPI requests, use a text editor to edit the .locrc file. Change the following variable:

```
LOC_SERVER_SOCKET_NUMBER=<new port number>
```

**More information:**

[Port Conflict Resolution](#) (see page 30)

## Change the Visibroker Naming Service Port Number

You can change the port number that the Visibroker Naming Service uses.

**Note:** The default port number is 14006.

**Follow these steps:**

1. Right-click My Computer and select Properties.  
The System Properties dialog opens.
2. Click the Advanced tab and click the Environment Variables button.
3. Select the NAMING\_SERVICE\_PORT and edit it to reflect the new port number:

NAMING\_SERVICE\_PORT=<new port number>

To change the Visibroker Naming Service port number on Solaris, set the environment variable in the spectrum60.env file. This file is located in the /opt/SPECTRUM directory. Change the following variable:

NAMING\_SERVICE\_PORT=<new port number>

## Remote Copy Process Daemon (rcpd) Port Number Configuration

The remote copy process daemon (rcpd) port number is configurable through the rcpd\_comm\_port .vnmrc resource.

**More information:**

[General SpectroSERVER \(.vnmrc\) Resources](#) (see page 13)  
[Port Conflict Resolution](#) (see page 30)

## CLI Daemon (vnmshd) Port Number Configuration

The CLI Daemon (vnmshd) port number is configurable through the vsh\_tcp\_port parameter.

**Note:** For information about the vsh\_tcp\_port parameter, see the *Command Line Interface User Guide*.

**More information:**

[Port Conflict Resolution](#) (see page 30)

# Chapter 4: Fault Tolerance

---

This section contains the following topics:

[About SpectroSERVER Fault Tolerance](#) (see page 63)

[SpectroSERVER Alarm Synchronization](#) (see page 68)

[Establish Fault Tolerance](#) (see page 72)

[Monitor the Changeover Between the Primary and Secondary SpectroSERVERs](#) (see page 78)

[How to Monitor the Secondary SpectroSERVER Status](#) (see page 79)

## About SpectroSERVER Fault Tolerance

Fault tolerance requires more than one SpectroSERVER to manage a given landscape. A copy of the database for that landscape is loaded on each SpectroSERVER. However, only a single copy is active at any time. The SpectroSERVER with the active database is known as the *primary SpectroSERVER*. The inactive database runs on a standby SpectroSERVER, which is the secondary SpectroSERVER. You can also install another inactive copy of the database on a tertiary SpectroSERVER.

If the primary SpectroSERVER fails, the database on the secondary SpectroSERVER becomes active, and the secondary SpectroSERVER starts managing the network. Applications that are connected to the primary SpectroSERVER are automatically switched to the secondary SpectroSERVER. When the primary SpectroSERVER returns to service, the applications automatically switch back to the primary SpectroSERVER, and the secondary SpectroSERVER becomes inactive again.

**Note:** Not all applications can exercise the full range of their capabilities when they are being run from a secondary SpectroSERVER. The main reason to set up a fault-tolerant environment is to ensure continuous monitoring of the network, not to create a full copy of CA Spectrum.

### More information:

[Trap Director in a Fault-Tolerant Setup](#) (see page 84)

## SpectroSERVER Precedence in a Fault Tolerant Environment

Primary, secondary, and tertiary SpectroSERVERs that manage the same landscape must all have the same landscape handle and the same modeling catalog. The servers are distinguished from one another with a numeric precedence value. The lowest number indicates the primary SpectroSERVER. SpectroSERVERs are installed with a default precedence value of 10. To designate a SpectroSERVER as a secondary server, assign it a higher precedence number, such as 20. Likewise, a tertiary SpectroSERVER would have a higher precedence than the secondary, for example, 30.

When you first set up a fault tolerant environment, you can assign precedence values at the time you are loading database copies on any standby SpectroSERVERs using the [SSdbload utility](#) (see page 72).

To change precedence values later, you can use the Loaded Landscapes subview. Access this subview by selecting a local landscape in the Navigation panel, and then selecting the Information tab in the Component Detail panel.

**Note:** The Loaded Landscapes subview is different from the SpectroSERVER Control subview. Access the SpectroSERVER Control subview by selecting the VNM in the Navigation panel and then selecting the Information tab in the Component Detail panel.

**More information:**

[Establish Fault Tolerance](#) (see page 72)

## SpectroSERVER Data Synchronization

A single database is active at any given time in a fault tolerant CA Spectrum environment. Therefore, the other databases must be updated periodically to reflect new models and changes to attribute values in the active database. This synchronization of data is accomplished through the CA Spectrum Online Backup feature. You can run Online Backup on demand or at regularly scheduled intervals. When you run Online Backup against the primary SpectroSERVER, it creates a backup copy of the current database. Online Backup automatically loads the copy onto each designated secondary SpectroSERVER.



As in any DSS environment, each of the SpectroSERVERs in a fault tolerant environment must have the same modeling catalog installed. Online Backup copies the current modeling catalog. However, it does not copy all of the .i files or other elements that are associated with individual management modules. Therefore, if you install any new management modules on your primary SpectroSERVER, also install the same new management modules on any secondary SpectroSERVERs.

**Note:** For more information, see the *Database Management Guide*.

EventDisp and the Alertmap files that are defined in the <\$SPECROOT>/custom/Events directory are propagated to fault-tolerant servers when the secondary SpectroSERVER polls the primary SpectroSERVER for status information.

## Support for Fault-Tolerant Archive Manager

Starting with Release 9.4 release, you can run the Archive Manager on the secondary SpectroSERVER host in a fault-tolerant SpectroSERVER environment. This secondary Archive Manager provides visibility to events in OneClick when the primary Archive Manager is down.

Primary or secondary SpectroSERVER locally stores events in the following two scenarios:

- When primary Archive Manager is down, and the primary SpectroSERVER is running. In this case, primary SpectroSERVER locally stores events as they are created until primary Archive Manager is up.
- When the primary SpectroSERVER host itself is down. In this case, the secondary SpectroSERVER locally stores events as they are created until the primary Archive Manager is up.

Before Release 9.4 release, OneClick server had no access to events in these two scenarios. As a result, no events were available in the OneClick Events view. Now, you can start the secondary Archive Manager on the secondary SpectroSERVER host to provide visibility to not only events as they are created when the primary Archive Manager is down, but also historical events.

When you start the secondary Archive Manager, it acts as a client to the primary SpectroSERVER to receive and log events as they are created. This behavior does not affect the normal connection between the primary SpectroSERVER and primary Archive Manager. As soon as the primary Archive Manager goes down, OneClick fails over to the secondary Archive Manager to provide event data.

When the primary SpectroSERVER host itself goes down, the secondary SpectroSERVER locally stores events, but also forwards events to secondary Archive Manager. When the primary Archive Manager comes up, the secondary SpectroSERVER transfers all the locally stored events to it.

## Archive Manager Data Synchronization

The secondary Archive Manager provides a best-effort synchronization of events, and there is no event synchronization that occurs between the primary Archive Manager and the secondary Archive Manager. When the secondary Archive Manager is running and connected to a SpectroSERVER, it receives a copy of all events as they are generated. Anytime the secondary Archive Manager is down, events are not stored on the secondary. This functionality is distinctly different from the functionality of primary Archive Manager, where events are stored by the SpectroSERVER for later transfer to the primary Archive Manager.

This means that when the secondary Archive Manager is started for the first time, its DDM database does not contain any events, and no attempt is made to synchronize with the primary. Once the secondary Archive Manager has been running for MAX\_EVENT\_DAYS configured in the .configrc, it is generally in sync with the primary Archive Manager database.

## Generate an Alarm if the Secondary SpectroSERVER Is Not Restarted

When a primary SpectroSERVER synchronizes its database with the secondary SpectroSERVER, a Contact Lost to Secondary Server (0x00010c0e) event and alarm are generated. The secondary SpectroSERVER has been brought down to load the new database from the primary SpectroSERVER.

You can set up a rule to process this alarm so that the alarm is generated only if the secondary SpectroSERVER is not restarted.

The EventPair rule lets you specify that a new event is generated if the Contact Lost to Secondary Server event occurs and a Contact Established to Secondary Server (0x00010c0f) event does not follow within a specified time period. You can then specify that this new event creates an event and an alarm indicating that the secondary SpectroSERVER is still down.

### Follow these steps:

1. Open the EventDisp file with a text editor.

**Note:** The EventDisp file is located in the <\$SPECROOT>/SS/CsVendor/Cabletron directory.

- Find the line that reads 0x00010c0e E 50 A 2, 0x00010c0e and change this line to the following:

```
0x00010c0e R Aprisma.EventPair, 0x00010c0f,  
    <numberofsecondstowait><generatedeventcode>  
<generatedeventcode>
```

Is the event code to generate if the secondary SpectroSERVER does not come up within the time specified in <numberofsecondstowait>.

- Add the following line to the EventDisp file:

```
<generatedeventcode>E 50 A 2, <generatedalarmcode>  
<generatedeventcode>
```

Is the event code generated in Step 2 if the secondary SpectroSERVER did not come up. 'E 50' indicates that the event is logged and has a severity value of 50. A 2 indicates that a major alarm is created. <generatedalarmcode> is the alarm code to generate based on this event.

- Create a Probable Cause file for this alarm that indicates that contact with the secondary SpectroSERVER has not been reestablished after data synchronization.

**Note:** For more information, see the *Event Configuration User Guide*.

## Secondary SpectroSERVER Readiness Levels

A secondary SpectroSERVER is considered to be at one of three different levels of readiness. Readiness depends on server configuration and status. The readiness levels are defined as follows:

### Hot

The secondary SpectroSERVER is running and is available to take over immediately upon failure of the primary SpectroSERVER because it is already polling. To configure a secondary SpectroSERVER for this level of readiness, add the following line to the .vnmrc file: secondary\_polling=yes. This statement causes the standby to commence polling and processing traps whenever it starts, regardless of its connection status with the primary SpectroSERVER.

### Warm

The secondary SpectroSERVER is running, but the server can take a short time to become fully available. The secondary SpectroSERVER has not been configured to start polling *until* it loses contact with the primary SpectroSERVER. For example, it has no secondary\_polling entry in the .vnmrc file, or the entry is set to no.

If the secondary\_polling entry is not in the .vnmrc file or the entry is set to no, the secondary SpectroSERVER does not process traps while in standby mode.

### Cold

The secondary SpectroSERVER is not running and must be started when there is a failure of the primary SpectroSERVER. In this case, it is irrelevant whether the secondary SpectroSERVER is configured for secondary polling.

### More information:

[Establish Fault Tolerance](#) (see page 72)

[Validate Fault Tolerance Configuration](#) (see page 74)

## SpectroSERVER Alarm Synchronization

The primary and secondary SpectroSERVERs use a Global Alarm Service (GAS) connection to share alarm information. The SpectroSERVERs use the alarm information to synchronize alarms. This synchronization helps to prevent duplicate alarms.

Options for fault-tolerant alarm synchronization include enablement of the service and debug logging. Settings in the .vnmrc file control these options. For more information, see [Fault Tolerant Alarm Service \(.vnmrc\) Resources](#) (see page 21).

When you are deploying VHM and a Chassis device with a secondary SpectroSERVER configured, alarm synchronization displays some unexpected behavior. When the primary SpectroSERVER goes down, alarms that were correlated on the primary SpectroSERVER are not correlated on the secondary SpectroSERVER. Instead, you see an alarm for each different condition or symptom. When the primary SpectroSERVER comes back online, the alarm correlation takes place appropriately. But the correlation is still not performed on the Chassis device.

The correlations are not preserved because the alarms on the secondary SpectroSERVER are generated rather than being created from events.

The process of alarm synchronization differs depending on whether the synchronization is from the primary SpectroSERVER to the secondary SpectroSERVER or from the secondary to the primary. The following sections describe each of these scenarios:

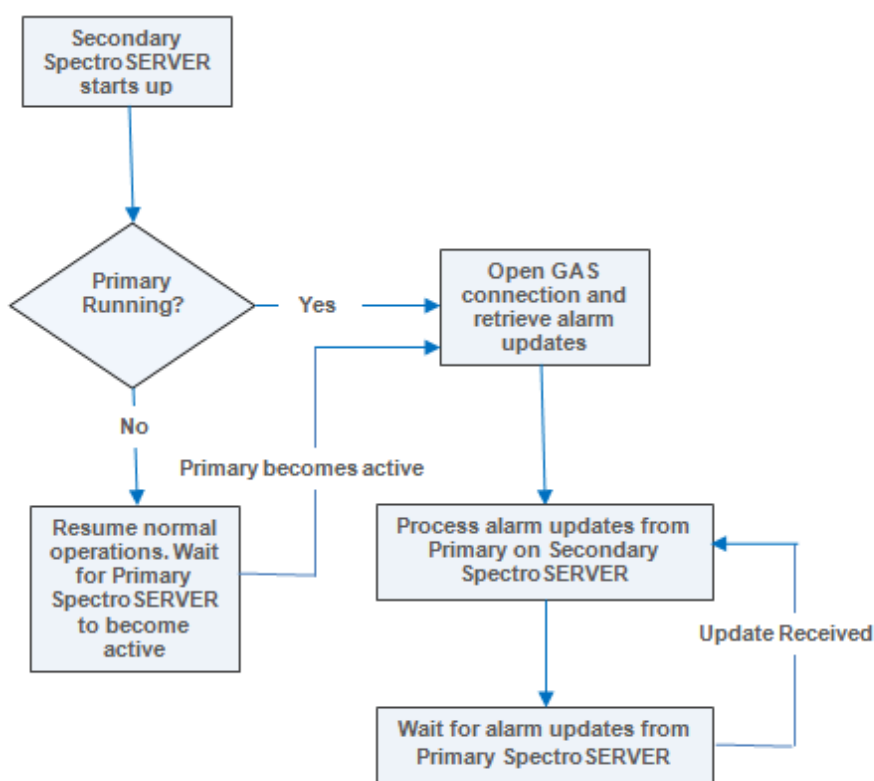
- [Synchronization from the Primary to the Secondary SpectroSERVER](#) (see page 68)
- [Synchronization from the Secondary to the Primary SpectroSERVER](#) (see page 70)

## Synchronization from the Primary to the Secondary SpectroSERVER

If the SpectroSERVER is running as a secondary SpectroSERVER, it tries to open a GAS connection to the primary SpectroSERVER. If the connection succeeds, the secondary SpectroSERVER registers to receive alarm updates from the primary SpectroSERVER. The secondary server remains connected to the primary and continues to receive alarm updates as they occur on the primary server. Otherwise, the secondary SpectroSERVER tries once each minute to connect until the primary SpectroSERVER becomes active.

**Note:** The Fault Tolerant Alarm Service must be enabled on both the primary and secondary servers for the secondary SpectroSERVER to attempt to connect to the primary SpectroSERVER for alarm synchronization. To control this feature, use the `ftasv_enabled` parameter in the `.vnmrc` file. For more information, see [Fault Tolerant Alarm Service \(.vnmrc\) Resources](#) (see page 21).

The following diagram describes how the secondary SpectroSERVER processes alarm updates from the primary SpectroSERVER:



- The new primary SpectroSERVER alarms are added on the secondary SpectroSERVER and marked as 'stale'. A new alarm replaces an existing alarm if they are equivalent. You can determine whether the alarms are equivalent by verifying the following parameters:

- Unique Alarm (IDs)
- Model Handle
- Probable Cause
- Alarm Discriminators

Two alarms are considered equivalent if they have the same ID. Alarms that are on the same model (that is, they have the same model handle) and that have the same probable cause are also considered equivalent unless they have different alarm discriminators.

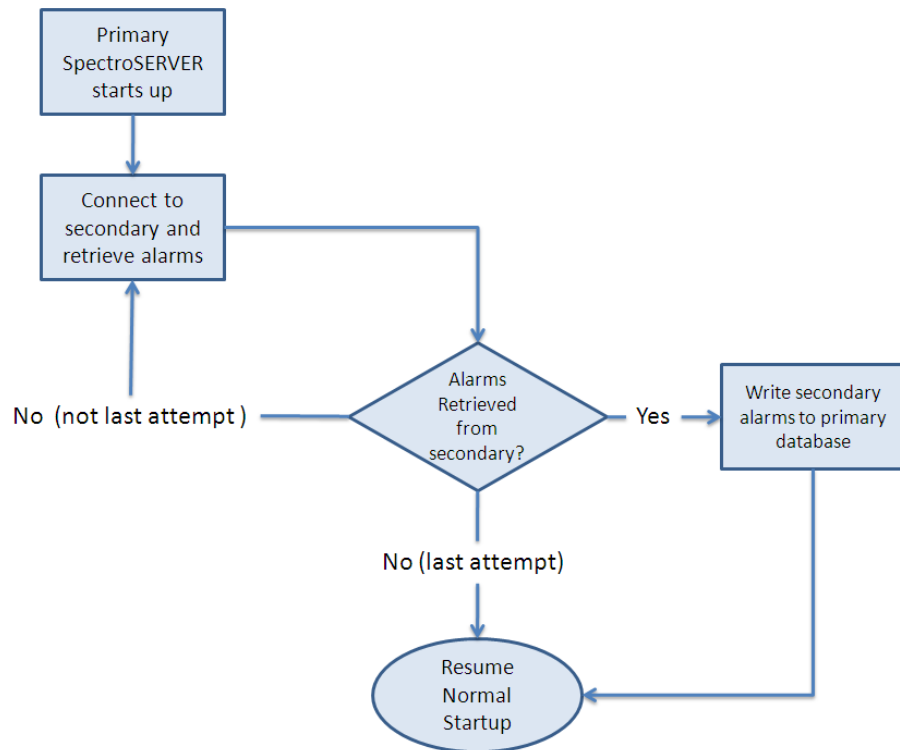
- The alarm attribute updates from the primary SpectroSERVER are applied to the same alarms on the secondary SpectroSERVER. If an alarm is cleared on the primary SpectroSERVER, it is also cleared on the secondary SpectroSERVER.
- The isManaged and isNotHibernating attributes are updated for maintenance mode alarm synchronization.

## Synchronization from the Secondary to the Primary SpectroSERVER

When the primary SpectroSERVER comes online after a failure or planned maintenance, it connects to the secondary SpectroSERVER, and it gets all available alarms. If the first connection attempt fails, the primary server can make multiple connection attempts to synchronize. The `ftasv_max_conn_retry_count` and `ftasv_conn_retry_interval` parameters in the `.vnmrc` file of the primary SpectroSERVER control the number and frequency of attempts. For more information, see [Fault Tolerant Alarm Service \(.vnmrc\) Resources](#) (see page 21).

**Important!** If the secondary SpectroSERVER is not running when the primary server attempts to connect to it, the primary exhausts its synchronization attempts. The result is a delay in starting the primary. This delay is unavoidable because it occurs before model activation. Avoid the situation by verifying that the secondary server is running when the primary starts. You can also decrease the retry count or interval size to reduce the potential delay. Or you can disable the Fault Tolerant Alarm Service using the `ftasv_enabled` parameter in the `.vnmrc` file.

The following diagram describes how the primary SpectroSERVER opens a GAS connection to the secondary SpectroSERVER and retrieves all of its alarms.



#### After a Successful Connection:

- After it retrieves the list of alarms, the primary SpectroSERVER writes the alarms to its database. Primary SpectroSERVER startup continues and these alarms are read from the database.
- The alarms from the secondary SpectroSERVER replace the alarms that are stored in the primary database at retrieval.
- All alarms that are read from the database are processed and asserted on the correct models.
- The secondary 'alarm set' event (0x10714) generates for each new alarm that was generated on the secondary SpectroSERVER while the primary SpectroSERVER was down.
- The secondary alarm clear event (0x10715) generates for each alarm that cleared on the secondary SpectroSERVER while the primary SpectroSERVER was down.

**Note:** No synchronization occurs for blue and gray alarms. Synchronization occurs for brown alarms, but not for WA\_Link models. The WA\_Link model exception only applies to brown alarms.

**After a Failed Synchronization:**

- If alarm synchronization fails when the primary SpectroSERVER is attempting to connect to the secondary, the primary starts up, but an event and critical alarm (0x10c35) are generated on the primary LocalScope model. The event and alarm include the reason for the failure of each attempt. To continue running the primary SpectroSERVER after alarm synchronization has failed, run an Online Backup to synchronize the primary and secondary SpectroSERVER databases.

## Establish Fault Tolerance

You can set up a fault-tolerant environment when you first install CA Spectrum, before any models have been created. Or you can set up a fault-tolerant environment after you install CA Spectrum.

The following procedure describes how to set up two SpectroSERVERs: a primary and a secondary. You can also set up a tertiary SpectroSERVER by taking the same steps. However, assign the tertiary SpectroSERVER a higher precedence number than the secondary SpectroSERVER.

**Note:** To establish fault tolerance in an environment with a Southbound Gateway integration, see the *Southbound Gateway Toolkit Guide*.

**Follow these steps:**

1. Install the same version of CA Spectrum with the same modeling catalog on both the primary SpectroSERVER and the secondary SpectroSERVER. Each server requires the same landscape handle.
2. Verify that both the primary and secondary SpectroSERVERs have entries in their .hostrc files that give the SpectroSERVERs mutual access permissions.

**Note:** If you are specifying secure users for the secondary SpectroSERVER in the .hostrc file on the primary SpectroSERVER, and the secondary SpectroSERVER is running in the Windows environment, include the user SYSTEM in the secure user list.

3. Verify that the MAIN\_LOCATION\_HOST\_NAME parameter in the .locrc file on the secondary SpectroSERVER server points to the same system name as the .locrc file on the primary SpectroSERVER. Otherwise, synchronization fails.
4. Configure the primary and secondary SpectroSERVERs so that the user running each SpectroSERVER is the same. If the users are not the same, the secondary SpectroSERVER fails or does not run properly after an Online Backup.



5. Make a copy of the primary SpectroSERVER database by running Online Backup. Or, if the SpectroSERVER is shut down, use the SSdbsave utility with the -cm argument (to save the modeling catalog and any new models).

**Note:** For more information, see the *Database Management Guide*.

6. Verify that the save file that you created is available to the server that hosts the secondary SpectroSERVER. Copy the file to the server if necessary.
7. On the secondary server, with SpectroSERVER shut down, navigate to the CA Spectrum SS directory and load the save file using the following command:

```
../SS-Tools/SSdbload -il -add precedence savefile
```

**precedence**

Specifies a numeric value greater than the primary server default value of 10 (20 is recommended).

**savefile**


Specifies the name of the save file previously created.

8. (Optional) Add the line 'secondary\_polling=yes' to the .vnmrc file to let the secondary SpectroSERVER function as a [“hot” backup](#) (see page 67).
9. Start the primary SpectroSERVER, if it is not already running.
10. Start the secondary SpectroSERVER.
11. To verify the setup, use the MapUpdate command with the view argument to display the current landscape map.

**Note:** For more information, see the *Database Management Guide*.

The secondary SpectroSERVER is now available to take over automatically if the primary SpectroSERVER fails. If you previously activated secondary polling, the secondary SpectroSERVER is available immediately. Otherwise, polling begins as soon as the server detects that it has lost contact with the primary SpectroSERVER.

When service switches from the primary SpectroSERVER to the secondary

SpectroSERVER, the Connection Status icon  displays yellow. To view the connection status of all servers in a landscape, click the Connection Status icon. In the Connection Status dialog, the Connection Status icon for each server in the landscape displays yellow to indicate the “switched” condition.

When the primary SpectroSERVER comes back online, the secondary SpectroSERVER stops polling (unless you have set secondary\_polling to 'yes'). All of the applications switch back to the primary SpectroSERVER. However, any edits that you make to the secondary SpectroSERVER while it is active are *not* automatically replicated to the primary SpectroSERVER. Manually recreate these modifications on the primary SpectroSERVER.

When you restart the primary SpectroSERVER, connections are accepted as soon as all models are loaded, but *before* all models are activated. The models can take some time to activate. Because the secondary SpectroSERVER stops polling as soon as the primary SpectroSERVER is restarted, a gap in your network management coverage can result.

To avoid this situation, edit the .vnmrc file on the primary SpectroSERVER so that the wait\_active resource is set to 'yes'. This parameter causes the server to wait until all of the models are activated before accepting any connections. The message area in the CA Spectrum Control Panel also dynamically displays the percentage of models that are activated. The SpectroSERVER can appear to take longer to come up. However when all of the models are activated, the SpectroSERVER is ready to manage the network.

You can also set the wait\_active resource to 'yes' on the secondary SpectroSERVER. During a planned shutdown of the primary SpectroSERVER, you can then verify in the CA Spectrum Control Panel that the secondary SpectroSERVER is ready to take over.

**Note:** For more information, see the *Database Management Guide*.

**More information:**

[About Distributed SpectroSERVER](#) (see page 9)

[SpectroSERVER Precedence in a Fault Tolerant Environment](#) (see page 64)

[Designate a New Main Location Server](#) (see page 35)

## Validate Fault Tolerance Configuration

After you have set up fault tolerance in a distributed SpectroSERVER deployment, verify that the OneClick server has access to both primary and secondary SpectroSERVERs. Without connectivity to both servers, the OneClick server cannot fail over to the secondary SpectroSERVER.

**Follow these steps:**

1. Access the OneClick Administration, Landscapes web page.
2. Check the 'Secondary Status' column. Verify that OneClick has established contact with the secondary SpectroSERVER.

The status also indicates whether Fault Tolerance is ready for failover.

The Fault Tolerance configuration is validated.

## Test Fault Tolerance

During an initial installation, the secondary SpectroSERVER might not have access to all of the devices to which the primary SpectroSERVER has access. This situation causes the secondary SpectroSERVER to generate false alarms. To avoid false alarms, verify that the secondary SpectroSERVER can manage your network devices by testing fault tolerance.

**Note:** Test fault tolerance whenever new devices are added to the primary SpectroSERVER.

**Follow these steps:**

1. With both the primary and secondary SpectroSERVERs up and running, bring down the primary SpectroSERVER.

The Connection Status icon  is yellow to indicate the "switched" condition.

A red connector indicates that the OneClick server was not able to contact the secondary SpectroSERVER.

2. Wait 15 - 20 minutes for the secondary SpectroSERVER to run.
3. Verify the following conditions:
  - The Connection Status icon does not display red.
  - All device models and pingable models maintain SNMP or ICMP contact.  
If this contact is lost, verify that the secondary SpectroSERVER has access to your devices. Contact a network administrator to resolve this problem, if applicable.
  - CA Spectrum is managing all devices that have an established contact state. Verify the status by checking for device contact or management contact loss alarms from any of the device models.
4. Restart the primary SpectroSERVER.

The Connection Status icon displays green to indicate a normal contact state.

## Fault-Tolerant Recovery

Following are the two possible failure scenarios:

- The primary SpectroSERVER stops. The secondary SpectroSERVER then forwards event and statistical information to the primary Archive Manager that is running on the server that hosts the primary SpectroSERVER. When the primary SpectroSERVER restarts, no event and statistical data have been lost.
- The computer where the primary SpectroSERVER and the primary Archive Manager are running stops operating completely. The secondary SpectroSERVER then caches event and statistical data in its database until the primary SpectroSERVER computer comes back online. If a secondary Archive Manager is running, historical, and real-time information is available in OneClick, but the information is still cached for transfer to primary Archive Manager.

Restart both the primary Archive Manager and the primary SpectroSERVER if their server goes down, or if the primary SpectroSERVER stops operating.

**Note:** It is no longer necessary to start the Archive Manager before the SpectroSERVER, the cached events from the secondary SpectroSERVER can be transferred at any time, even after the primary SpectroSERVER has started logging new events.

**Follow these steps:**

1. Start the SPECTRUM Control Panel on the primary SpectroSERVER host.
2. To start the SpectroSERVER, click Start SpectroSERVER on the SPECTRUM Control Panel.

When the primary Archive Manager is again operational, the secondary SpectroSERVER connects and transfers its cached event data to the primary Archive Manager.

## Change the Host Names of the Primary and Secondary SpectroSERVERs

SpectroSERVERs in a fault-tolerant environment use a precedence value that is associated with their host names to recognize their relationship to one another. Therefore, to preserve the fault-tolerant relationship, use SSdbsave and SSdbload to change the host name of your primary SpectroSERVER.

**Follow these steps:**

1. Save the database using SSdbsave with the -cm option.
2. Change the host name.

3. Reload the database with the save file that you created in the first step. Run SSdbload with the -il option and the -replace option:

```
SSdbload -il -replace precedence savefile
```

This command causes the database to associate the new host name with the precedence value (10) that designates a primary SpectroSERVER.

The change in the host name is communicated to any warm or hot standby SpectroSERVERs the next time the databases are synchronized as a result of Online Backup being run.

In the meantime, however, the host name change prevents the standby SpectroSERVERs from detecting that the primary SpectroSERVER is running. As a result, any SpectroSERVER that is configured as a warm standby starts polling.

4. Load the save file on the warm standby using SSdbload with the -il and -replace options, and specify a higher precedence value (for example, 20) that designates it as a standby.

Now you can change the host name of the secondary SpectroSERVER.

**Follow these steps:**

1. Save the database using SSdbsave with the -cm option.
2. Make the change to the host name.
3. Reload the database with the save file that you created in the first step. Run SSdbload with the -il option and the -replace option:

```
SSdbload -il -replace precedence savefile
```

This command causes the database to associate the new host name with the precedence value (20) that designates a secondary SpectroSERVER.

When you restart the secondary SpectroSERVER, the server communicates the new host name and precedence to the primary SpectroSERVER.

**Note:** For more information, see the *Database Management Guide*.

## Monitor the Changeover Between the Primary and Secondary SpectroSERVERs

You can use watches to monitor the status of your fault tolerant environment. Create a watch that alerts you when either the primary or the secondary SpectroSERVER is ready to take over.

**Follow these steps:**

1. Create a watch on the VNM model to monitor the PercentInitialized (0x11da6) attribute.

When the value of this attribute is equal to 100 percent, the SpectroSERVER has been initialized and is ready to take over.

2. Set up the watch to generate an event or an alarm or run a script when the following expression evaluates to TRUE:

PercentInitialized == 100

3. Set up the watch as an active polled watch.
4. Synchronize the secondary SpectroSERVER with the primary SpectroSERVER to propagate the watch.
5. Specify a value for the Model\_Name (0x1006e) attribute in the watch expression. This attribute notifies you only when the secondary SpectroSERVER is ready to take over.

For example, if the following watch expression evaluates to TRUE, the secondary SpectroSERVER <sec\_server> is ready to take over.

(PercentInitialized == 100) & (Model\_Name= <sec\_server>)

**Note:** For more information, see the *Watches User Guide*.

6. Add the following line to the .vnmrc file on the secondary SpectroSERVER to limit the potential for false events or alarms:

is\_secondary = TRUE

This setting lets the secondary SpectroSERVER drop events unless CA Spectrum determines that the secondary SpectroSERVER has taken over as the primary SpectroSERVER.

## How to Monitor the Secondary SpectroSERVER Status

You can create a watch that alerts you when the secondary SpectroSERVER server is acting as the primary SpectroSERVER.

**Follow these steps:**

1. Create a watch on the VNM model to monitor the value of the secondary SpectroSERVER's PausePolling attribute (0x11b63).

When this attribute is set to FALSE on the secondary SpectroSERVER, the secondary SpectroSERVER is polling and is acting as the primary SpectroSERVER.

For example, if the following watch expression evaluates to TRUE, the secondary SpectroSERVER <sec\_server> is acting as the primary SpectroSERVER.

```
!PausePolling & (Model_Name == <sec_server>)
```

2. Set up the watch as an active polled watch.
3. Synchronize the secondary SpectroSERVER with the primary SpectroSERVER to propagate the watch.

**Note:** For more information, see the *Watches User Guide*.

4. Add the following line to the .vnmrc file on the secondary SpectroSERVER to limit the potential for false events or alarms:

```
is_secondary = TRUE
```

This setting lets the secondary SpectroSERVER drop events unless CA Spectrum determines that the secondary SpectroSERVER has taken over as the primary SpectroSERVER.





# Chapter 5: Working with Trap Director

---

This section contains the following topics:

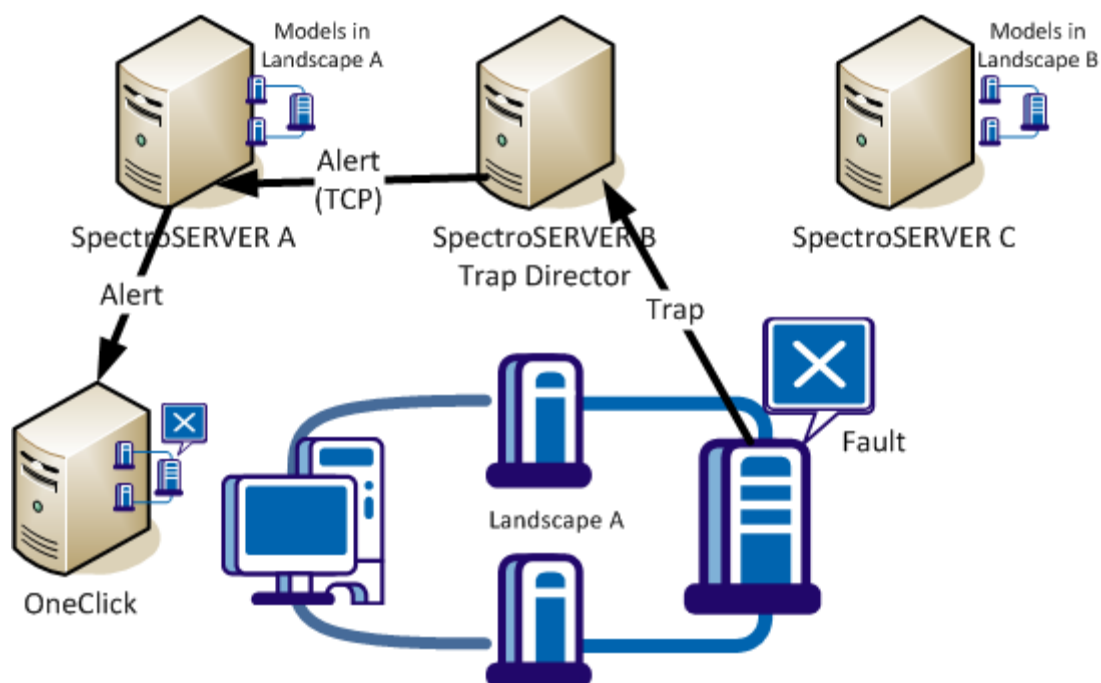
[Trap Director](#) (see page 81)

[Traps and Memory Usage](#) (see page 82)

[Trap Data Traffic Consolidation](#) (see page 82)

## Trap Director

Trap Director is a SpectroSERVER feature that you can enable in a distributed SpectroSERVER environment. Trap Director receives traps and processes them on the Trap Director host server. Trap Director then uses the information in traps to create alerts, which it forwards over a TCP connection to the SpectroSERVER. The main Trap Director mechanism is therefore the CA Spectrum internal alerts that the SpectroSERVER sends to the models on remote hosts for additional processing. The following diagram illustrates the Trap Director architecture:



To use Trap Director, designate one SpectroSERVER as the Trap Director server. [Enable Trap Director](#) (see page 84) on that server. Then configure that server as the network management station (NMS) recipient for traps from devices that are modeled in the remote landscapes.

Trap Director supports encrypted SNMPv3 traps. CA Spectrum attempts to use your SNMPv3 profiles to decrypt incoming encrypted SNMPv3 traps.

The available SpectroSERVER SNMPv3 Profiles are treated as credential templates for unknown SNMPv3 traps. When an encrypted SNMPv3 trap is received, the SNMPv3 profiles are used to decrypt the trap.

**Note:** A SpectroSERVER running as a trap forwarder can process SNMPv3 traps if the Trap Director landscape has an SNMPv3 profile with valid credentials.

## Traps and Memory Usage

When Trap Director is enabled, monitor the memory usage of the SpectroSERVER process. In cases where a large number of traps (more than 100 per second) are forwarded to multiple remote landscapes, a situation can occur in which the SpectroSERVER cannot forward all of the incoming traps. The traps are then queued for further processing. If the situation continues, the memory that the queue uses can exhaust the memory resources that are available for use by the SpectroSERVER process. The SpectroSERVER then shuts down unexpectedly because it lacks sufficient memory.

In addition to the process memory usage, closely monitor the following attribute in the VNM model:

`alert_remote_fwd_queue_length` (Attribute ID: 0x130c3)

Generally, if the queue size continues to grow beyond 1000 elements, we do not recommend enabling Trap Director in your environment.

## Trap Data Traffic Consolidation

As network growth prompts you to apportion models among additional landscapes, you can retain the original landscape host as the trap destination for these models. You can thus avoid reconfiguring new trap destinations on the associated devices. Conversely, you can consolidate multiple trap destinations and can designate a single SpectroSERVER to receive and route traps. If load sharing is an important factor, you can enable Trap Director on multiple servers. Each server handles traps from a set of devices that are configured to send traps to that server.

Trap Director maintains an up-to-date model address cache. This cache lets Trap Director match trap sources with models. Trap Director uses matching to forward traps to destination models on remote landscapes. By keeping cache information current, Trap Director ensures that CA Spectrum can generate events for models regardless of their location.

**Note:** Trap Director performance can be affected when new landscapes are added to the distributed environment. Performance can also be affected when large numbers of traps are processed for many new models. Trap notification latency can increase in these circumstances.

## How Trap Director Updates the Address Cache

Trap Director maintains a cache of the IP addresses and locations of models in the distributed environment. The address cache serves as an index to determine where to forward alerts. Models can be regularly added to landscapes, removed from landscapes, and moved between landscapes. Therefore, Trap Director also regularly updates the cache to keep its content current. Records that meet a retention period (or aging) threshold that you can specify are removed. Trap Director also performs cross-landscape searches for model IP addresses that are not available in the cache when it receives traps from those IP addresses.

The following steps describe how Trap Director determines the destination model for a trap:

1. The Trap Director server receives a trap.
2. Trap Director compares the IP address that was included in the trap to IP addresses in cache records.
3. If Trap Director finds a match, it forwards an alert to the model on the remote landscape.

Otherwise, Trap Director performs the following tasks to determine the destination model and forward the alert:

- Trap Director searches known landscapes for a matching IP address.
  - When it finds the destination model that is associated with the IP address on the remote landscape, Trap Director updates the cache with model information. It forwards an alert to the model.
4. CA Spectrum generates an event on the VNM model on the Trap Director server if the trap is dropped because the cross-landscape search did not find a matching address. The event indicates that the destination model was not found.

A match is not found if, for example, the model has been deleted.

## Trap Director in a Fault-Tolerant Setup

If you want to implement a fault-tolerant Trap Director setup, devices must be configured to forward traps to both the primary and the secondary SpectroSERVERs. The secondary server routes traps only when it detects that the primary server has failed. The secondary SpectroSERVER receives Trap Director settings during the primary backup, or synchronization process.

### More information:

[About SpectroSERVER Fault Tolerance](#) (see page 63)

## Trap Storm Settings

To handle trap storms, Trap Director uses trap storm settings that are configured on modeled devices. When Trap Director detects a trap storm, it stops forwarding alerts to models on remote landscapes. Trap Director also asserts trap storm alarms for the models.

Trap storms can originate from devices that are not modeled in CA Spectrum. For these storms, Trap Director uses the trap storm handling settings that are configured for the VNM model on the Trap Director server.

## Enable and Disable Trap Director

You can enable and disable Trap Director on a SpectroSERVER.

**Note:** Use the OneClick Attribute Editor or the CA Spectrum Command Line Interface to set attribute values on the VNM model for a server.

### Follow these steps:

1. Expand the Trap Management subview in the VNM model Information tab.
2. Click set in the Enable Trap Director field, and select Enabled.  
Trap Director is enabled.
3. (Optional) To disable Trap Director, click set in the Enable Trap Director field, and select Disabled.

Trap Director is disabled.

## Define the Cache Record Retention Period

You can control how frequently the Trap Director trap cache ages out. You can define the cache record retention period.

To define the cache record retention period, define a retention period for the following attribute:

`trap_cache_age_out_minutes` (0x12ad5)

**Default:** 180 (minutes)



# Index

---

.

.hostrc file • 34, 49  
.LocalRegFile • 34  
.locrc file • 34, 61  
.vnmrc file • 12, 79

## A

address cache • 83  
ADMINPRIVS • 40  
alarm synchronization • 68  
alarms • 66  
alerts, forwarding • 81  
APPNAME • 40  
ARGV • 40  
AUTOBOOTSTART • 40  
AutoDiscovery • 47  
AUTORESTART • 40

## B

bind\_retry\_interval • 13

## C

CA Spectrum  
    distributed installation • 55  
    domain • 36  
changing  
    Archive Manager port number and socket number • 61  
    landscape handles • 37  
    location server port number and socket number • 61  
    SpectroSERVER host names • 76  
    SpectroSERVER port number • 60  
    VisiBroker Naming Service port number • 62  
    Windows password in processd • 39  
CLI Daemon (.vnmshd) • 62  
cold • 67  
comm\_port • 13  
connect\_time\_limit • 13

## D

default port numbers • 30, 60  
default socket numbers • 30, 60

device\_limit • 13  
disable\_redundancy\_when\_using\_loopback • 13  
Distributed SpectroSERVER  
    about • 9  
    requirements • 32  
duplicate models • 48

## E

enable\_traps\_for\_pingables • 13  
ENV • 40  
event\_batch\_max\_size • 18  
event\_batch\_timeout • 18  
event\_record\_increment • 18  
EventDisp file • 66  
EventPair rule • 66  
Events Archive (.vnmrc) Resources • 18  
expiration\_date • 13

## F

fault tolerance  
    about • 63  
    and Trap Director setup • 84  
    environment • 59, 64  
    establishing • 72  
    status monitoring • 78, 79  
    testing • 75  
firewalls  
    and remote SpectroSERVERs • 57  
    communicating across • 55

## G

GAS • 68  
general SpectroSERVER (.vnmrc) resources • 13

## H

handshake\_timeout • 13  
home landscapes • 48  
host security • 49  
hot • 67

## I

IIOP (Internet Inter-ORB protocol) • 56  
install tickets • 40, 42, 45

---

## L

- landscape handles
  - about • 32, 36
  - assigning • 36
  - changing • 37
- landscape map • 10, 11, 36
- landscapes • 10, 32
- lh\_set\_utility • 37
- location servers • 32, 34
- log\_user\_events • 18
- LOGNAMEPATH • 40

## M

- main location server • 32, 34, 35, 48, 57
- MAIN\_LOCATION\_HOST\_NAME • 34
- max\_bind\_retry\_count • 13
- max\_connections • 13
- max\_device • 13
- max\_event\_records • 18
- max\_total\_work\_threads • 19
- min\_client\_version • 13
- Model\_Name attribute • 78
- modeling catalog • 11, 32
- multiple interfaces • 29
- multiple landscapes • 11
- multiple SpectroSERVERs • 30

## N

- name resolution • 29
- Network Address Translation (NAT) • 60
- network models • 47
- network partitioning • 47
- NUMPROCS • 40

## O

- OneClick
  - default ports and firewalls • 57
  - web server • 56, 57
- online backup • 64

## P

- PARTNAME • 40
- partslist directory • 40
- password • 13
- PausePolling attribute • 79
- PercentInitialized attribute • 78
- persistent\_alarms\_active • 13

- port conflicts • 30
- primary SpectroSERVER • 76
- procd\_comm\_port • 13
- processd
  - about • 37
  - and the userconf process • 46, 47
  - changing the Windows password in • 39
  - configuration • 40
  - in Solaris environments • 38
  - in Windows environments • 39, 50
  - restarting • 45
  - shutdown • 49, 50
  - stopping • 45
- processd.pl • 45
- processd\_log file • 37
- processd\_log.bak file • 37
- processd\_shutdown\_timeout • 50
- production environments • 36

## R

- rcpd\_comm\_port • 13, 62
- readiness levels • 67
- remote copy process daemon (rcpd) • 62
- resource\_file\_path • 13
- restarting
  - primary Archive Manager • 76
  - primary SpectroSERVER • 76
- RETRYMAX • 40
- RETRYTIMEOUT • 40

## S

- secondary SpectroSERVER • 67, 76
- secondary\_polling • 67
- SERVERPROCESS • 40
- SERVICE • 40
- snmp\_comm\_port • 13
- snmp\_trap\_port • 13
- snmp\_trap\_port\_enabled • 13
- SpectroSERVERs
  - across firewalls • 59
  - and firewalls • 57
  - and OneClick Web Server communication • 57
  - changeover • 78
  - monitoring • 79
  - multiple • 32
  - precedence • 64
  - shutdown • 49, 50
- SSdbload utility • 37, 64, 76



---

STARTPRIORITY • 40

STATEBASED • 40

## T

tcp\_buffer\_size • 13

test environments • 36

TICKETUSER • 38, 40

time zones • 49

TL1 gateway agent • 30

trap data traffic • 82

Trap Director • 81, 82, 83, 84, 85

    Trap Director, and memory usage • 82

trap storm settings • 84

## U

unsupported\_attr\_poll\_interval • 13

use\_log\_queue • 18

user models • 11, 34

userconf process • 46, 47

Users Group • 46, 47

## V

Visibroker Naming Service • 62

vnm\_close\_timeout • 13

vnm\_file\_path • 13

vnm\_message\_timeout • 13

vsh\_tcp\_port parameter • 62

## W

wait\_active • 13

WaitToKillServiceTimeout • 50

warm • 67

watches • 78, 79

work thread (.vnmrc) resources • 19

work\_thread\_age • 19

WORKPATH • 40