

# **CA Spectrum® Infrastructure Manager**

## **Distributed SpectroSERVER Administrator Guide**

**r9.2.1**



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 © 2011 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

## CA Product References

This document references the CA Spectrum® Infrastructure Manager (CA Spectrum).

## Contact CA Technologies

### Contact CA Support

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

- 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

### Provide Feedback

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

If you would like 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 .....	11
Modeling Catalogs .....	11
User Models .....	11
SpectroSERVER (.vnmrc) Resources .....	12
General SpectroSERVER (.vnmrc) Resources .....	13
Events Archive (.vnmrc) Resources .....	20
Work Thread (.vnmrc) Resources .....	21
Fault Tolerant Alarm Service (.vnmrc) Resources .....	23
How to Pack Up CA Spectrum Utilities and Move Them to Another Computer .....	24
Pack Up CA Spectrum Utilities .....	26
Move CA Spectrum Utilities to Another Computer .....	26

## Chapter 2: Setting Up a Distributed SpectroSERVER Environment 31

Name Resolution Requirements .....	31
CA Spectrum and Multiple Interfaces .....	31
Communication Between Multiple SpectroSERVERs .....	32
Port Conflict Resolution .....	33
DSS Environment Requirements .....	34
Location Servers .....	35
How Location Servers Interact .....	35
Main Location Server Connection .....	36
Designate a New Main Location Server .....	37
Landscape Map Integrity Preservation .....	38
Landscape Handles .....	38
Assign Landscape Handles .....	39
How to Change a Landscape Handle .....	39
Process Daemon (processd) .....	40
Processd Differences in Windows and Solaris Environments .....	41
Change the Windows Password in Processd .....	41
Install Tickets Files .....	42
Stop and Restart Processd .....	48
How the Userconf Process Works During Installation .....	49
How Userconf Works When a User Logs On .....	50
Network Partitioning .....	50

---

Duplicate Models in a Distributed Environment .....	51
Failure to Contact Home Landscape Errors .....	51
Host Resource Configuration File (.hostrc) .....	52
Time Zones in a DSS Environment .....	52
SpectroSERVER Shutdown .....	53
SpectroSERVER Shutdown in a Solaris Environment .....	53
SpectroSERVER Shutdown in a Windows Environment .....	53
Landscape Mapping from Existing DSS Setup .....	54

## **Chapter 3: Communication Across Firewalls in the Distributed SpectroSERVER Environment** **57**

Communication Across Firewalls .....	57
SpectroSERVER and OneClick Web Server Communication Across Firewalls .....	58
OneClick Default Ports and Firewalls .....	59
HTTP Listen Port .....	59
CORBA Listen Port .....	59
Remote SpectroSERVERs and Firewalls .....	60
Primary and Secondary SpectroSERVER Communication Across Firewalls .....	61
CA Spectrum Configuration Files for NAT Firewall Environments .....	62
Default Port Configurations .....	62
Change the SpectroSERVER Port Number .....	63
Change the Archive Manager Port Number and Socket Number .....	63
Change the Location Server Port Number and Socket Number .....	63
Change the Visibroker Naming Service Port Number .....	64
Remote Copy Process Daemon (rcpd) Port Number Configuration .....	64
CLI Daemon (vnmsd) Port Number Configuration .....	65

## **Chapter 4: Fault Tolerance** **67**

About Fault Tolerance .....	67
SpectroSERVER Precedence in a Fault Tolerant Environment .....	68
Data Synchronization .....	68
Secondary SpectroSERVER Readiness Levels .....	74
How to Establish Fault Tolerance .....	75
Validate Fault Tolerance Configuration .....	77
Test Fault Tolerance .....	78
Restart the Primary Archive Manager and Primary SpectroSERVER .....	78
Change the Host Names of the Primary and Secondary SpectroSERVERs .....	79
Monitor the Changeover Between the Primary and Secondary SpectroSERVERs .....	81
How to Monitor the Secondary SpectroSERVER Status .....	82

---

<b>Chapter 5: Working with Trap Director</b>	<b>83</b>
Trap Director .....	83
Trap Data Traffic Consolidation .....	83
How Trap Director Updates the Address Cache .....	84
Trap Director in a Fault-Tolerant Setup .....	84
Trap Storm Settings .....	85
Enable and Disable Trap Director .....	85
Define the Cache Record Retention Period .....	85
<b>Index</b>	<b>87</b>





# 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 11)

[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 24)

## About Distributed SpectroSERVER

Distributed SpectroSERVER (DSS) is a powerful modeling feature that enables the distribution of management for portions of a large-scale network, either geographically, or across multiple servers in a single physical location. DSS can improve CA Spectrum performance when managing a computing infrastructure by distributing the network load introduced by management traffic, and delegating management functions to remote workstations.

Using DSS, you can create a unified representation of the computing infrastructure, composed of multiple landscapes, each with its own local SpectroSERVER. In a DSS environment, a SpectroSERVER client, such as the OneClick Console, can access information from more than one SpectroSERVER at the same time.

DSS includes the following features:

### **Distributed Visibility**

The OneClick console displays information for all deployed SpectroSERVERs. Summary alarm counts are displayed for each SpectroSERVER so the network manager can quickly see where hotspots are.

### **Localized Polling Traffic**

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

### Scalability

As a network expands, it is often less expensive to use low-end workstations as additional, dedicated SpectroSERVER machines, rather than upgrading one workstation to accommodate one SpectroSERVER to manage a large non-distributed environment.

### Fault Tolerance

It is possible to set up fault tolerance between SpectroSERVERs with DSS. A secondary SpectroSERVER can be provided as a fault tolerant backup or standby for a primary SpectroSERVER, so that network management would continue even if the workstation running the primary SpectroSERVER fails.

By simply reloading a given landscape's database onto a secondary SpectroSERVER, the network manager is protected against failures. When a failure occurs that disables the primary SpectroSERVER, the secondary SpectroSERVER automatically takes its place, and all CA Spectrum applications automatically use the secondary SpectroSERVER.

### More information:

[How to Establish Fault Tolerance](#) (see page 75)

## Landscapes

A network domain managed by a single SpectroSERVER is referred to as a landscape. A landscape is composed of the models, associations, attribute values, alarms, events, and statistics belonging to a specific SpectroSERVER. Each landscape contained in a network is unique, and must be identified by a unique landscape handle (ID).

Each landscape can be represented by a landscape icon in OneClick. The landscape icon provides a graphical representation of a SpectroSERVER database. Through double-click zones and menu selections, the landscape icon provides access to remote network models and also presents a rollup of alarm information for the devices 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. If any changes occur in the SpectroSERVER topology (such as a SpectroSERVER being added or removed from the network), an internal discovery mechanism updates the landscape maps of all of the other SpectroSERVERs in the DSS environment.

A landscape model is a container model that provides a means of connecting to other SpectroSERVER 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 (model types, relations, and so on) installed on a particular SpectroSERVER that can be used in creating models. The set of models created through these templates make up that SpectroSERVER’s landscape.

**Note:** When OneClick starts, it queries the default landscape for a list of available model types.

When you model multiple landscapes using DSS, each landscape must contain identical modeling catalogs. This means that all model types that exist in one landscape’s modeling catalog must also exist in the modeling catalog of every other landscape to which it is connected. If you install a new management module on one SpectroSERVER, you must install the same management module on every other SpectroSERVER in the landscape map.

## User Models

A user model contains information about an individual user’s permissions as well as other user data. This information is stored in a landscape. When you first install CA Spectrum, a default user model is created, representing the user you specified in the CA Spectrum installation program’s Installation Owner field.

**Note:** For more information about creating additional users, see the *Administrator Guide*.

In a DSS environment, the same user model must exist in all landscapes for users to be able to connect to remote SpectroSERVERs in the landscape map and to enable the various 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 to see if the user model already exists in other landscapes. If the user model exists in other landscapes, the user models in those landscapes are automatically updated to reflect any changes made to the initial user model.

**Note:** Deleting a user model only deletes the user model in that landscape. Duplicate user models on other SpectroSERVERs in the landscape map must be deleted manually.

## SpectroSERVER (.vnmrc) Resources

SpectroSERVER resources are defined in the `<$SPECROOT>/SPECTRUM/SS/.vnmrc` file. Many default values of .vnmrc 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 entered in the .vnmrc (Virtual Network Machine runtime configuration) file define path names and default settings for the SpectroSERVER. Your CA Spectrum system software includes a .vnmrc file containing default settings for SpectroSERVER resources.

You can modify resources to do the following:

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

All .vnmrc 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 make changes to the .vnmrc file while SpectroSERVER is running, the changes will not take effect until 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 behaviors of the SpectroSERVER. The following describes the general SpectroSERVER (.vnmrc) resources:

### **comm\_port**

Specifies the TCP port number that indicates the port through which the client's user interfaces communicate with SpectroSERVER. The port number is defined during the installation process.

This command has the following format:

```
comm_port=0xBEEF
```

#### **0xBEEF**

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

### **expiration\_date**

Indicates the date your license for CA Spectrum expires. The CA Spectrum key entered during the installation process contains the expiration date. This resource should not be changed after installation.

This command has the following format:

expiration\_date=*mm/dd/yyyy*

***mm***

Is the month

***dd***

Is the day

***yyyy***

Is the year

**password**

Is automatically populated with the spectrum key that is entered during the installation process. This resource should not be changed after installation.

This command has the following format:

password=*password*

***password***

Is supplied by CA

**snmp\_comm\_port**

Specifies a value that can be used to select a port from which SNMP requests can be sent through the SpectroSERVER.

This command has the following format:

snmp\_comm\_port=*port#*

***port#***

Sets to any unsigned 16-bit integer in the range from 0x400 (1,024) to 0xFFFF (65,535). Some implementations (for example, IBM Mainframe MVS system) of SNMP agents treat the port as a signed number. In these cases this resource must be set to a value between 0x400 (1,024) and 0x7FFF (32,768).

**Default:** 0xFFFF

**snmp\_trap\_port\_enabled**

Binds the SpectroSERVER to the SNMP trap port and listen for traps. When set to FALSE, the SpectroSERVER will 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 your LAN's traffic. A large buffer size improves throughput. The resource accepts a numeric value which 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 kilobytes are rounded up to 8K and values above 64 kilobytes are set to 64K.

**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**

Is the file path for your VNM resource files. This resource may be left blank.

**wait\_active**

Determines whether the server will accept connections as soon as possible after all models are loaded, or wait until all models are active. If this resource is set to Yes, the Control Panel's message area displays a running percentage of models 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 will accept 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 will accept whether they are clients or 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 could fail when the maximum number of client connections is exceeded. When you increase the maximum SpectroSERVER connections, you may also have to increase the number of your workstation's open file descriptors to help ensure CA Spectrum performs and operates smoothly. See the vendor's documentation for information about increasing the number of file descriptors on your SpectroSERVER's workstation.

### **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 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 listen 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. You must add it to the file manually, using the following syntax:

```
enable_traps_for_pingables = TRUE
```

**Default:** TRUE

### **device\_limit**

Specifies the maximum number of devices that can be modeled when such a limit applies. The value is derived from the CA Spectrum key entered at the time of installation and is not user-editable. Any managed node that is not a simple host, a pingable, or a proxy agent of a node, will count as a modeled device. If no device limit applies to the CA Spectrum product you installed, there will be no value associated with this resource. (See also max\_device).

This command has the following format:

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

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

Are any models derived from both the IPAddress (0x000102bc) model type and from the device (0x0001004b) model type.

In addition, models derived from the DIRIG\_NODE application are counted, as well as the following broadband models: MOTCBLMODEM, SAEXPLORER, and DOCSISCM.

The exception to this rule is:

GEN\_HOST; PINGABLE; PATROL\_PET; and WINDOWS\_PC.

### **max\_device**

Applies to CA Spectrum products for which the 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 and CA Spectrum generates a red alarm when the number of models reaches 100% of the specified device limit. This resource allows you to set a value lower than the actual device limit, which gives you earlier warning that the limit is being approached.

This command has the following format:

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

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

Are any models that have IP addresses and are derived from the device model type AND the specified number is LESS than the value specified for 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 on redundancy, 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 and CA Spectrum automatically retains alarm-related information (such as troubleshooter assignments, status, and so on) when the SpectroSERVER is shut down and brought back up. The alarms that existed prior to shutdown continue to exist. 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 that you do not add the persistent\_alarms\_active resource unless you integrate a third-party application that retains alarm-related information.

The effects of adding this resource are:

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

**unsupported\_attr\_poll\_interval**

Specifies the amount of time that CA Spectrum will wait to poll an external attribute once that attribute 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 64)

## 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 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. The maximum value is limited by the upper limit of your system storage capacity.

**Default:** 20,000

**event\_record\_increment**

Specifies the number of records to be deleted from the Event Log database when the number of Event Log records in the database 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 deleted from the database. The minimum value is 100.

**event\_batch\_max\_size**

Sets the maximum number of events allowed 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 could delay the creation of alarms.

This command has the following format:

```
use_log_queue=<blank>
```

**Default:** <blank>

## Work Thread (.vnmrc) Resources

SpectroSERVER is a multi-threaded 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 work activity. When the common pool of work threads is exhausted, new work threads are created and the pool grows to meet the needs of increased activity.

Work threads that are no longer needed by subsystems are returned to the common pool for later use. Subsequent needs are serviced by threads taken from the pool (or by newly allocated threads, if the pool is empty). When a thread remains unused for a specified period of time, it is removed from the pool and its resources are returned to the system. This process is called aging.

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

### **max\_total\_work\_threads**

Specifies the maximum number of work threads that may be allocated for all SpectroSERVER subsystems. Since each work thread consumes processing resources and requires a significant block of memory, its value should be set based on the capacity (memory size and speed) of your system.

When the value of this resource is too high, SpectroSERVER could periodically run out of memory. When the value of this resource is too low, SpectroSERVER operations may 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's WorkThreadsMaxAvail attribute is 500.

### **work\_thread\_age**

Work threads that are no longer needed by a subsystem 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.

This resource should be set to a value compatible with subsystem activity. Significant processing overhead is required to create a new work thread. Setting the value of this resource too low taxes your system's resources by creating new threads too often in response to work thread demands, too high and 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 get alarm information from each other. If the initial attempt to connect fails, subsequent attempts can be made. The Fault Tolerant Alarm Service controls alarm synchronization using the following settings:

### **ftasv\_enabled**

Controls if alarm synchronization is enabled. This command must be included in the .vnmrc file for both the primary and secondary servers, and it must be set to the same value on both servers.

This command has the following format:

```
ftasv_enabled=true
```

**Default:** True

**Important!** Having `ftasv_enabled` set to true for one server and set to false for the other server is not a supported configuration; the same value must be used on both servers.

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

Specifies the number of times the primary server will attempt to connect to the secondary server for synchronization after an initial connection attempt has failed. This parameter must be used in the .vnmrc file of the primary SpectroSERVER.

**Note:** This parameter is used by the primary server when attempting to connect to the secondary only; 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:** 4 retries (for a total of 5 synchronization attempts with the secondary)

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

Specifies the number of seconds between synchronization attempts by the primary server to the secondary. This parameter must be used in the .vnmrc file of the primary SpectroSERVER.

**Note:** This parameter is used by the primary server when attempting to connect to the secondary only; it 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**

Specifies that debug output for alarm synchronization activity be generated. This command must be included in the .vnmrc file for both the primary and secondary servers. Debug output is written to the VNM.OUT file of each server, with each message beginning 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 attempts to connect to it, the primary will exhaust its synchronization attempts and cause a delay in starting the primary. When using default settings for `ftasv_max_conn_retry_count` and `ftasv_conn_retry_interval`, this would be a delay of 2 minutes (4 retries with 30 seconds between them). This delay must be endured because it happens before model activation. If this is not acceptable, make sure the secondary is running when the primary starts, or decrease the retry count or interval size to reduce the potential delay.

### **More information:**

[Fault Tolerance](#) (see page 67)

[SpectroSERVER Alarm Synchronization](#) (see page 70)

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

Packtool.pl is a script that packs up the CA Spectrum Command Line Interface (CLI), CA Spectrum SS Logger, AlarmNotifier, the modelinggateway tool, and the sbgwimport tool so that these utilities can be sent by FTP to another computer. The script helps ensure that the relative directory structure of the utilities and their support files are retained when the files are moved.

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



Consider the following 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.
- If the CA Spectrum utilities are moved to a computer that only has the OneClick server installed, consider the following:
  - The version and patch version of the OneClick server on each computer must match. Each time a patch is installed, it must be installed on each computer, the packtool.pl script must be re-run, and the utilities must be moved again.
  - The installation user on the computer where the utilities are being moved to should be the same as the installation user on the computer where the files are being packed up.
  - If a debug patch or PTF is installed on the computer where you are moving the utilities to, the debug patch or PTF must be reinstalled after the utilities are moved.
- All CA Spectrum processes must be stopped on the computer where the utilities are being packed up.
- All CA Spectrum processes must be stopped on the computer you are moving the utilities to before you extract the tools\_bundle file.
- You must be a root user on Solaris and Linux platforms, or a user with administrative privileges on Windows platforms to extract the tools\_bundle file.

**Important!** To run the CA Spectrum utilities, you must be logged on to the computer you are moving the utilities to as a valid CA Spectrum user or you will receive an error. For example, if you are logged on to a Linux workstation as "root," and there is no user model in CA Spectrum named "root," you will receive an error saying, "Error: NO\_USER."

To pack up CA Spectrum utilities and move them to another computer, do the following:

1. [Pack Up the SPECTRUM Utilities using the packtool.pl script.](#) (see page 26)
2. [Extract the tools\\_bundle file on another computer](#) (see page 26).

## Pack Up CA Spectrum Utilities

Before you can move the CLI, SSLogger, AlarmNotifier, the modelinggateway tool, and the sbgwimport tool to another computer, you must pack them up.

### To pack up CA Spectrum utilities

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

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

```
<$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 or you can move CA Spectrum utilities to a computer that is not running CA Spectrum.

### To move CA Spectrum utilities to a computer that is only running the OneClick server

1. Pack up the CA Spectrum utilities.
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 packed up and moved AlarmNotifier and event and pcause files, verify that the paths in `<$SPECROOT>/SG-Support/CsResource/preferences/*.prf` have the correct locations for the event and pcause files. Also, verify that the `ui=` and the `lhandle=` options are correct.
6. Set the `<$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 has the local machine name and the machine name of the computer you moved the utilities from in it.
10. Verify that the `.hostrc` file on the main location server contains the host name of the computer the `tools_bundle` file is extracted on.
11. Verify that the `.LocalRegFile` has the correct Main Location Server in it.
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, which 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.

**To move CA Spectrum utilities to a computer that is not running CA Spectrum**

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

**Windows:**

```
c:\win32app\spectrum
```

**Solaris and Linux:**

```
/usr/spectrum
```

**Note:** Be sure to change the working directory (chdir) to these directories.

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. 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 packed up and moved 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 that the `ui=` and the `lhandle=` options are correct.
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 has the local machine name and the machine name of the computer you moved the utilities from in it.
10. Verify that the `.LocalRegFile` has the correct Main Location Server in it.
11. Verify that `vnmsd/.vnmsdrc` has the correct main SpectroSERVER name in it.
12. Install the `processd` service:

**Windows:**

- If SRAdmin is not installed, install it by entering the following:

```
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 hostname of the machine the tools\_bundle file is extracted on.

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, which 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 26)

# Chapter 2: Setting Up a Distributed SpectroSERVER Environment

---

This section contains the following topics:

- [Name Resolution Requirements](#) (see page 31)
- [CA Spectrum and Multiple Interfaces](#) (see page 31)
- [Communication Between Multiple SpectroSERVERs](#) (see page 32)
- [Port Conflict Resolution](#) (see page 33)
- [DSS Environment Requirements](#) (see page 34)
- [Location Servers](#) (see page 35)
- [Landscape Handles](#) (see page 38)
- [Process Daemon \(processd\)](#) (see page 40)
- [Network Partitioning](#) (see page 50)
- [Duplicate Models in a Distributed Environment](#) (see page 51)
- [Host Resource Configuration File \(.hostrc\)](#) (see page 52)
- [Time Zones in a DSS Environment](#) (see page 52)
- [SpectroSERVER Shutdown](#) (see page 53)
- [Landscape Mapping from Existing DSS Setup](#) (see page 54)

## Name Resolution Requirements

For the SpectroSERVER system or OneClick web server system to communicate with a SpectroSERVER, the SpectroSERVER system or OneClick web server system must be able to resolve the non-fully-qualified host name of the SpectroSERVER to an IP that can be used to reach the SpectroSERVER.

We recommend that hosts files be used for the name resolution of SpectroSERVER host names to help ensure that name resolution will not be impacted by a network failure.

## CA Spectrum and Multiple Interfaces

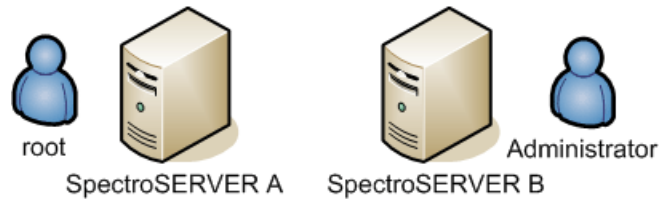
All CA Spectrum servers will bind to and listen on all available interfaces, and advertise themselves by non-fully-qualified host names. This should allow maximum flexibility when configuring a management topology. When establishing connections, all interfaces are tried in the order returned by the operating system, until the connection is successful.

## Communication Between Multiple SpectroSERVERs

SpectroSERVERs in a distributed environment should be installed and run as the same user. If all SpectroSERVERs are installed and run as the same user, no further user model configuration is required. If two SpectroSERVERs are installed and run as different users, they cannot communicate until both users are configured on both SpectroSERVERs.

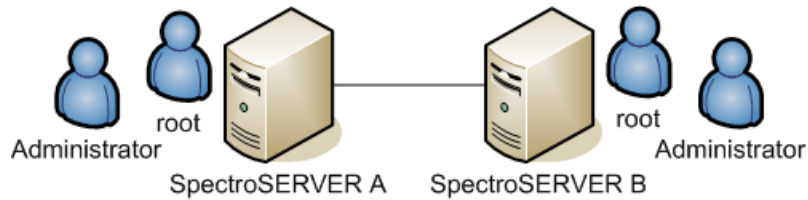
### Example: SpectroSERVERs A and B Cannot Communicate

This example shows SpectroSERVER A installed as "root" and running as root. SpectroSERVER B is installed as "Administrator" and runs as Administrator. A connection cannot be established between the two SpectroSERVERs because they cannot pass through each other's user security, as shown in the following illustration:



### Example: Configuration for SpectroSERVER to SpectroSERVER Connection

This example shows a user "root" created on SpectroSERVER B, and a user "Administrator" created on SpectroSERVER A. This configuration enables the two SpectroSERVERs to communicate with each other, as shown in the following illustration:





## Port Conflict Resolution

If you are concerned about a port conflict with another application, you can change the default port numbers used by CA Spectrum processes and services.

The following table describes the default port numbers used by CA Spectrum processes and services.

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

Port	CA Spectrum Components	Protocols	Notes
80	OneClick/SRM	HTTP	None
162	SpectroSERVER	UDP	Port used to listen for SNMP traps.
3306	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	mysql for BOXI on Solaris/Linux	TCP	Not used for remote connections
6844	SDC	TCP	None
6400	BOXI CMS	TCP	Not used for remote connections
7777	CLI vnmsd	TCP	See <a href="#">CLI Daemon(vnmsd)</a> (see page 65)
14001	OneClick Server	TCP/CORBA	See <a href="#">OneClick WebServer Host</a> (see page 59)
14002	SpectroSERVER	TCP/CORBA	See <a href="#">SpectroSERVER</a> (see page 63)
14003	ArchMgr	TCP/CORBA	See <a href="#">Archive Manager</a> (see page 63)
14004	LocServ	TCP/CORBA	See <a href="#">Location Server</a> (see page 63)
14006	Naming service	TCP/CORBA	None
14010	ArchMgr	TCP/CORBA	See <a href="#">Archive Manager</a>

Port	CA Spectrum Components	Protocols	Notes
			(see page 63)
31415	Telnet through SpectroSERVER		None
46517	sradmin	TCP	Remote Administration Daemon <b>Note:</b> For more information about the sradmin process, see the <i>Installation Guide</i> .
47870	ArchMgr	TCP/SSAPI	See <a href="#">Archive Manager</a> (see page 63)
48879	SpectroSERVER	TCP/SSAPI	See <a href="#">SpectroSERVER</a> (see page 63)
51966	rcpd	TCP	See <a href="#">Remote Copy Process Daemon (rcpd)</a> (see page 64)
56063	LocServ	TCP	See <a href="#">Location Server</a> (see page 63)
61904	processd	TCP	This port is not configurable.
64222	TL1d	TCP/EPI	TL1 gateway agent
65259	processd	TCP	This port is not configurable.

## DSS Environment Requirements

When you create a distributed environment, the following conditions must be met when installing multiple SpectroSERVERs and modeling multiple landscapes to represent those servers:

- Each landscape must contain an identical modeling catalog containing a database's model types and relations. This provides a consistent base for CA Spectrum intelligence.
- The same user models must exist in each landscape. User models can be seen in the Users tab in OneClick.
- The number of connections between subnets modeled in different landscapes must be minimized.

- The number of identical devices that are modeled in more than one landscape must be limited.
- Each landscape is identified by a unique landscape handle. You must assign each modeled landscape a unique landscape handle so that CA Spectrum can distinguish it from other landscapes in the DSS environment.

**Note:** For information about installing CA Spectrum see the *Installation Guide*.

**More information:**

[Assign Landscape Handles](#) (see page 39)

## Location Servers

When you install a SpectroSERVER, you also automatically install a *location server*. The location server is used to locate other CA Spectrum services on the network. CA Spectrum processes, such as Archive Manager, 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 the CA Spectrum installation, one of the location servers in a landscape map (or CA Spectrum domain) must be manually designated as the *main location server*.

Ideally, the main location server should be installed on a highly reliable and available machine. The main location server propagates service advertisements between location servers and communicates these service locations between other location servers on the network. You must specify a main location server machine name every time you install CA Spectrum.

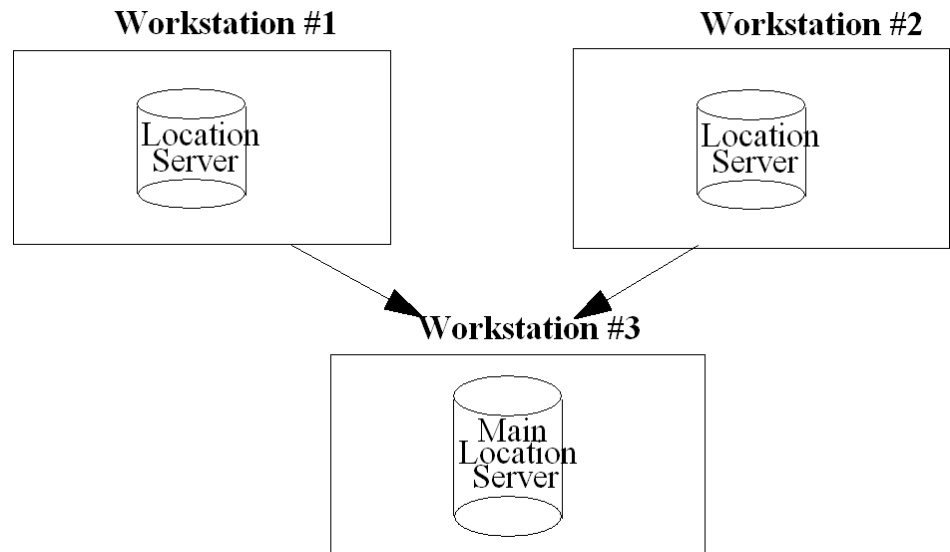
### How Location Servers Interact

Every CA Spectrum installation includes the following two files, which determine the location server hierarchy and the interaction between each location server and the main location server:

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

In a distributed SpectroSERVER network, 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 location server is designated as the main location server.
3. The local location server connects to main location server to determine the location of network services, as shown in the following illustration:



If the main location server goes down, other location servers have their service information cached in memory until the main location server is back up.

## Main Location Server Connection

Each SpectroSERVER must have a connection to the SpectroSERVER running on the machine designated as the MLS, as specified in its `<$SPECROOT>/LS/.locrc` file.

This connection requires the following:

- Each SpectroSERVER must allow the other SpectroSERVER host to connect to it by having the proper entries in each other's .hostrc files.
- Each SpectroSERVER must contain a CA Spectrum user model for the user which the other SpectroSERVER is running as.

**Note:** The main landscape (as specified by the value of MAIN\_LOCATION\_HOST\_NAME in the .locrc file) is modeled in CA Spectrum and placed in the VNM topology for each SpectroSERVER (if not already modeled elsewhere). If the connection to the main landscape is lost, an alarm is generated on this model.

## 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 updates the name of the main location server to which the OneClick web server will connect.

**Note:** If you change the main location server on the SpectroSERVER and you do not update the OneClick web server, CA Spectrum does not function properly. For more information about the OneClick Administration pages, see the *Administrator Guide*.

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

Assume a distributed network is comprised of computers 1 through 4, and computer 1 is designated as the main location server. If you want to have computer 4 as the main location server instead, you would first have to reconfigure computer 4 as the new main location server and then reconfigure computers 1, 2, and 3 to point to computer 4 as the main location server.

#### To designate a new main location server from the .locrc file

1. Navigate to the <\$SPECROOT>/LS directory.
2. Change the following entry in the .locrc file to point to the new main location server (computer 4):

```
MAIN_LOCATION_HOST_NAME=Computer 4
```

**To designate a new main location server from the Location Server Configuration dialog**

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 of the dialog to point to computer 4.

**More information:**

[How to Establish Fault Tolerance](#) (see page 75)

## Landscape Map Integrity Preservation

It is common to have separate production and test environments and to want to maintain the integrity of these environments. As long as no SpectroSERVER in the test environment is configured to connect to a production environment SpectroSERVER as its main location server or conversely, the landscape map integrity will be maintained.

## Landscape Handles

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

**Note:** We recommend that you use a sequential numbering scheme when establishing landscape handles. You can associate a landscape handle with a significant feature of the landscape it represents, 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, and the result may not appear to relate to the landscape feature you wanted.

## 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!** You must run the lh\_set utility before you run the SpectroSERVER for the first time. If you run the SpectroSERVER before running lh\_set, CA Spectrum assigns a default landscape handle that is the same whenever and wherever it is assigned by CA Spectrum. This creates the potential for duplicate landscape handles when multiple landscapes are configured. This means that these landscapes can never be accessed simultaneously from the same application.

### To assign a landscape handle with the lh\_set utility

1. Navigate to CA Spectrum's SS directory.
2. Enter the following:

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

The new landscape handle can be specified 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 about installing CA Spectrum, see the *Installation Guide*.

### More information:

[DSS Environment Requirements](#) (see page 34)

## How to Change a Landscape Handle

If a SpectroSERVER has been previously started, the process for changing the landscape handle of the SpectroSERVER database involves changing the landscape handle of all those models that were created automatically at startup, and updating the landscape handle specified in several resource files. The landscape handle is included in the model handle of each model in the SpectroSERVER database. Changing the landscape handle requires converting all model handles from the old landscape handle to the new landscape handle.

### To change the landscape handle of a SpectroSERVER database

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 about the SSdbload utility, see the *Database Management Guide*.

4. Use the lh\_set utility to assign the new landscape handle.
5. Start the SpectroSERVER.
6. Use the Modeling Gateway tool to import the models into the SpectroSERVER.

**Note:** For more information about the Modeling Gateway tool, 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 that are run on various servers and clients in a DSS environment. This daemon starts processes when requested by an application, such as the CA Spectrum Control Panel. You can also configure processd with install tickets to start processes automatically at system boot up. Install tickets can also be used to automatically restart critical processes should they stop unexpectedly.

The CA Spectrum installation program makes the necessary changes so that processd starts automatically upon system boot on Solaris systems (where it must run as root) and on Windows systems (where it runs as the LocalSystem account).

Processes are launched and tracked with processd only when an application requests such actions. Under normal circumstances, processd operates in the background and is transparent to the user.

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 both the name of the file and the line number on which the error was found.

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 42)

## Processd Differences in Windows and Solaris Environments

There are two main differences in the operation of processd in the Solaris and Windows environments, due to differences in their native security architectures:

- When you request the start of a process from a remote connection in the Windows environment, the process always starts as the user specified in the Windows Service Configuration dialog. In the Solaris environment, the process will start as the user making the request.
- For server processes that need to stay running after a user logs out (or before anyone logs in), a SERVERPROCESS field is used in the Windows environment. In order for Windows to not shut down these processes during logout, these server processes are started as the user specified in the Windows Service Configuration dialog box (not the TICKETUSER specified in the install ticket in the Solaris environment).

**More information:**

[Install Tickets Files](#) (see page 42)

## Change the Windows Password in Processd

When you install CA Spectrum in a Windows environment, you are prompted to enter a username and password in the Windows Service Configuration dialog. This username and password is used by processd to start CA Spectrum processes as the specified user, and allows these processes to run before a user logs in and after they log out.

**Note:** See the *Installation Guide* for instructions on entering this username and password.

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

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

**To change the Windows password in processd**

1. Make sure you are logged on as a member of the local Administrator's group. In addition to being a member of the CA Spectrum Users group, you must be an Administrative user to change the processd user name and password.

2. Open a Command Prompt.
3. Navigate to CA Spectrum's /lib/SDPM directory.
4. Enter the following command:

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

***user***

Specifies the user's user name.

***newpassword***

Specifies the new password for this user name.

If the user name or the password contains special characters that may be misinterpreted by the shell, you must enclose the user name or password in quotes when running this command. For example, the following user name contains a domain name and user separated by a backslash (\). The password contains an asterisk (\*). Both of these qualify as special characters, and therefore the user name and password must be enclosed in quotes, 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, the following events will be generated in the Windows event log and processd will not start:

- 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

An administrator can configure processd to automatically start processes at system boot up and to signify that a process should be restarted automatically if the process should somehow stop. This functionality is available through files called install tickets and is supported by 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.

New install tickets can be added to the partslist directory. Install tickets must confirm to the rules of the install ticket format, and processd must be restarted to identify any new or modified install tickets added to the partslist directory. These files are read on 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. The naming scheme for install ticket files is in the form of `<PARTNAME>.idb` where `<PARTNAME>` is a key used internally to distinguish one process from another. When you add new install tickets to the partslist directory, `<PARTNAME>` should be a name that uniquely identifies the process to be controlled by processd.

The following defining fields are used for install tickets. The format is `<fieldname>;<value>;` where `<fieldname>` is defined as one of the names displayed below, and `<value>` is read as a string of one or more characters (no quotes allowed) that provides a definition for the corresponding field name.

**PARTNAME**

Identifies a particular process/application. The value is a multi-character string with no spaces. The install ticket for this application should be named `<PARTNAME>.idb` where `<PARTNAME>` is the process name.

**APPNAME**

Defines the name of the application. This is a multi-character string.

**WORKPATH**

Specifies where you want to run the application from. You must supply a value for this field *before* it can be used as part of the ARGV field described below.

**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, and should be commented out in install tickets that you create.

**AUTORESTART**

Indicates whether a process will be automatically restarted if it dies for any reason. If this field is not included in the install ticket, automatic restart is disabled by default. Values accepted are Y, y, N, n.

**AUTOBOOTSTART**

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

### **STATEBASED**

Indicates whether the process has more than one state when starting up. This usually means the process will send an "is ready" ticket when completely ready to communicate. It is currently reserved for use in Purism-supplied install tickets only. If this field is not included in the install ticket, it is disabled by default. Values accepted are Y, y, N, n.

### **NUMPROCS**

Specifies the number of instances of a process allowed on one platform. Values accepted are numeric values.

### **RETRYTIMEOUT**

Specifies the number of seconds over which processd will keep trying to restart the application after failure.

### **TICKETUSER**

Defines the username of the user allowed to run the process when the AUTOBOOTSTART and/or AUTORESTART fields are set. This field is only required when the AUTOBOOTSTART and/or AUTORESTART fields are included in the install ticket. This field is not used in the Windows environment (as all processes run as the processd install user).

### **RETRYMAX**

Specifies the number of times processd will try to restart an application within the specified RETRYTIMEOUT period before giving up.

### **STARTPRIORITY**

Indicates the relative startup priority of the process in relation to its dependency on other processes.

- The value for standalone processes is 10.
- The value for processes that depend on standalone processes is 20.
- The value for processes that depend on the SpectroSERVER is 30.

### **SERVERPROCESS**

Indicates to processd whether this process should continue to run after the user logs off. When this field is enabled, the process is always started as the user configured to run the processd service. This field is only applicable in the Windows environment. It is set to yes by default for the following processes:

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

**SERVICE**

Denotes whether the ticket represents a Windows service. This allows processd to manage Windows services via the Service Control Manager.

**ENV**

Adds one or more variables to the application's environment. If a variable has multiple values, each is listed on a separate line with the macro <CSPATHSEP> appearing between the value and the semi colon ending the line.

**ARGV**

Defines the argument list of the process. This includes the executable path followed by any number of arguments (spaces allowed).

**More information:**

[Process Daemon \(processd\)](#) (see page 40)

[Processd Differences in Windows and Solaris Environments](#) (see page 41)

**Examples: Install Tickets**

Install ticket syntax conventions are as follows:

- Any line beginning with the pound sign (#) is assumed to be a comment.
- All field names must be followed by a semicolon (;). All values following each field name must also be followed by a semicolon (;).
- Anything following the second semicolon is disregarded.
- There are four macros that can be used. They are <CSEX>, <CSBAT>, <CSCMD>, and <CSPATHSEP>. Based on the platform you are using, the appropriate definition will be substituted. <CSPATHSEP> should be used instead of the actual path separator to avoid parsing conflicts.

The following is an example of an 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 is a second example of an 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<CSEXE>
-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 a new install ticket file, and want to start the process specified in this ticket, you can use the restart option to do so. The restart option eliminates the need to stop and restart processd and all of the processes currently being run by processd.

### To start a new install ticket process on Solaris

1. Become root.
2. Navigate to CA Spectrum's /lib/SDPM directory.
3. Enter the following command:

```
processd.pl restart
```

The process is started.

### To start a new install ticket process on Windows 2000/NT

1. Make sure you are logged on 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 the lib/SDPM/runtime directory has become corrupted, you must stop and restart processd.

### To stop and restart processd on Solaris

1. Become root.
2. Navigate to the /lib/SDPM directory.
3. Enter the following command:
4. Make sure that all CA Spectrum processes are shut down (otherwise problems with the SpectroSERVER may result).
5. Remove all entries in CA Spectrum's /lib/SDPM/runtime directory.
6. Restart processd by entering the following command:

```
processd.pl start
```



**To stop and restart processd on Windows 2000/NT**

1. Make sure you are logged on 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 2000/NT, the processd.pl <start/stop> commands also stop and start the CA Spectrum processes that run as NT services, for example, MySQL and VisiBroker.

**More information:**

[SpectroSERVER Shutdown in a Solaris Environment](#) (see page 53)

## How the Userconf Process Works During Installation

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

The userconf process does the following during a CA Spectrum installation:

1. Runs userconf -install %SPECROOT%, which does the following:
  - 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 is there).
  - Modifies the registry so that userconf (with no arguments) is run every time a user logs on to the machine.
2. Runs userconf -start to start processd. The -start flag forces processd to start without verifying that the user is a member of the CA Spectrum Users Group.

**Note:** The installation program automatically 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. By adding the -start flag, processd starts without verifying that the user is a member of the CA Spectrum Users Group. This allows the installation to continue without forcing 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). This allows `processd` to update PATH environment changes that are part of the Exceed installation.

## How Userconf Works When a User Logs On

The `userconf` process is added to the registry Run key during installation. When a user logs on, `Userconf` starts automatically and checks to see if the user is a member of the CA Spectrum Users group:

- If the user is a member of the CA Spectrum Users group, `userconf` verifies (and reconfigures as needed) `cygwin` and `Exceed` and then starts `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 CA Spectrum Users group, a message appears, stating that CA Spectrum is installed on the machine, but the user is not in the CA Spectrum Users group.

For the user to use CA Spectrum, the user must be added to the CA Spectrum Users group and then the user must log out of CA Spectrum and log back in.

If the user does not want to use CA Spectrum, the user must select the “user doesn't want to run spectrum” check box. If the user selects the check box, `userconf` continues to run when the user logs on, but exits silently.

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

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

## Network Partitioning

In your network model, you can partition your network in whatever way is logical for your network application (this assumes you are modeling a network that is already configured.) However, if you are creating multiple landscapes (or plan to in the future) and you want to use Discovery to create your Topology hierarchy, some methods are more compatible with Discovery than others.

The objective is to minimize the duplication of models in your landscapes and to limit the number of connections between landscapes. Some models should be duplicated to facilitate fault isolation, such as a router that connects two IP subnets.

Your choices for network partitioning include the following:

- Your first choice for partitioning should be to create partitions that are subnets, defined by IP address boundaries. IP address ranges provide the most convenient means of restricting Discovery exploration.

- If IP address boundaries are already established, and they do not provide a means of separating landscapes, then your second choice is to establish unique community names to differentiate devices in each landscape. This method requires more effort, since these names must be added to each device's community names table.
- Your last choice is to create partitions according to some other parameter (time-zone, state, and so on) that consists of a mix of IP subnet addresses. With this choice, you will probably have to model your network manually. However, even with this choice you should maintain as few connections between landscapes as possible and, of course, you must observe modeling rules. When too many connections exist, you begin to compromise the advantages of DSS.

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

## Duplicate Models in a Distributed Environment

In a DSS environment, CA Spectrum keeps track of duplicate models (for example, a user model existing on multiple landscapes) by designating one a "home" model on the SpectroSERVER that is running on the machine hosting 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 reflect the state of the "home" model on the MLS. This is because 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 made to duplicate models that are not the home models are relayed to the home model's landscape, which then distributes the request to all the other duplicate models.

### Failure to Contact Home Landscape Errors

Certain editing operations can fail when the home landscape of the model under edit cannot be contacted. In these cases, OneClick reports relation errors such as the following from the SpectroSERVER:

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

This can occur when the home model on the SpectroSERVER running on the "root" MLS machine 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's SpectroSERVER and the home SpectroSERVER is not running.
- A network problem is preventing SpectroSERVER access.
- A security issue is preventing the SpectroSERVERs from communicating with each other.

## Host Resource Configuration File (.hostrc)

The .hostrc file restricts client application access to each local SpectroSERVER. Client applications will not be able to connect to the local landscape unless the .hostrc file is configured to permit this access. The .hostrc file can be configured with a text editor.

**Note:** For more information about configuring the .hostrc file, 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 allows connections to and from that machine, while adding a "+" symbol permits unrestricted access to and from all remote machines. Client access from individual remote machines can be specified by adding the machine name to the .hostrc file.

CA Spectrum implements host security in many different layers; sometimes by hostname and other times by IP address. We recommended listing hostnames in the .hostrc file, which includes all IP addresses associated with the hostnames. If an IP address is used instead of listing the hostname in the .hostrc file, connections initiated by the hostname may not work properly.

## Time Zones in a DSS Environment

In DSS installations that span multiple time zones, all activities that occur on a SpectroSERVER occur with respect to the SpectroSERVERs local time, 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 landscape that the device is part of.

**Note:** For more information about creating schedules in OneClick, see the *Operator Guide*.

## SpectroSERVER Shutdown

We recommend that you first shut down the SpectroSERVER using the CA Spectrum Control Panel before shutting down the machine that the SpectroSERVER is running on. However, if you choose to use the operating system shut down procedure as a way to shut down the SpectroSERVER, you may want to increase the amount of time allowed for processd to shut down to avoid the SpectroSERVER being shut down prematurely.

### SpectroSERVER Shutdown in a Solaris Environment

In the Solaris environment, by default, processd waits 20 seconds for all sub-processes to shut down before it shuts down completely. If the SpectroSERVER takes a long time to shut down, processd waits for a longer time before shutting down. You can change the default value using the `PROCESSD_SHUTDOWN_TIMEOUT` environmental variable. This variable and its appropriate value (in milliseconds) should be added to the `<$SPECROOT>/spectrum60.env` file. For example, if you want processd to wait 60 seconds for all sub-processes to shut down, add the following line to the `spectrum60.env` file:

```
PROCESSD_SHUTDOWN_TIMEOUT=60000
```

Once you have made this entry, you must [stop and restart processd](#) (see page 48) for the changes to take effect.

### SpectroSERVER Shutdown in a Windows Environment

In the Windows environment, processd waits until all sub-processes have shut down before it shuts down completely. However, Windows has a registry setting called `WaitToKillServiceTimeout`, which is the length of time (in milliseconds) that Windows will wait for all services to stop after a Windows shutdown is initiated. If a service (such as processd) is still running after the amount of time specified by `WaitToKillServiceTimeout`, Windows terminates this service. Since the default value is only 20 seconds, this may not give the SpectroSERVER enough time to shut down completely when the system is shut down.

To extend this time, perform the following 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 that pops up, change the value to anything up to 600,000 milliseconds (10 minutes).
6. Click OK.
7. Restart the Windows machine for the change to take effect.

## Landscape Mapping from Existing DSS Setup

### Symptom:

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

### Solution:

The landscape maps are cleaned and copied to the new DSS setup so that there are no references to the existing DSS setup.

### To map the landscape from an existing DSS setup

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

The location server is now pointing to the local machine 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 including all the servers.
2. Save the SpectroSERVER databases.

The landscape maps are clean.
3. Change the MLS of these SpectroSERVER back to 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 31)





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

---

This section contains the following topics:

[Communication Across Firewalls](#) (see page 57)

[SpectroSERVER and OneClick Web Server Communication Across Firewalls](#) (see page 58)

[OneClick Default Ports and Firewalls](#) (see page 59)

[Remote SpectroSERVERs and Firewalls](#) (see page 60)

[Primary and Secondary SpectroSERVER Communication Across Firewalls](#) (see page 61)

[CA Spectrum Configuration Files for NAT Firewall Environments](#) (see page 62)

[Default Port Configurations](#) (see page 62)

## Communication Across Firewalls

Communicating across a firewall can apply in many network environments. The likelihood of firewalls being involved is even greater in a distributed environment.

**Note:** Prior to beginning a distributed CA Spectrum installation, the firewall must be set up to allow traffic on port 46517 (this is the port sradmin).

Depending on the type of firewalls you deploy and other factors such as cost and the level of security required, there are a number of options available for allowing the necessary communications to occur. These include Virtual Private Networks (VPNs), node-to-node tunnels or conduits, and proxies that place wrappers around packets before passing them through the firewall.

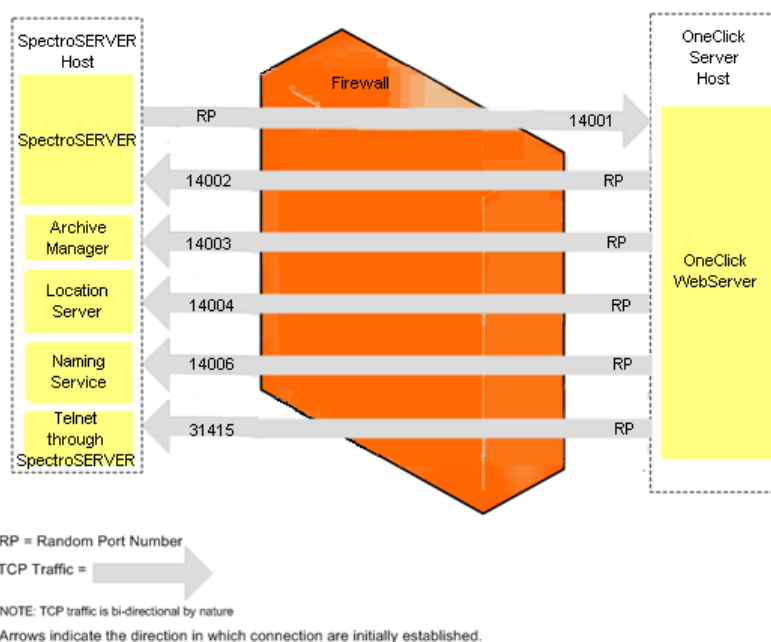
We recommend you work with your network's firewall administrator to determine what measures you need to take to help ensure that CA Spectrum and OneClick traffic can traverse any firewall deployed in your network.

## SpectroSERVER and OneClick Web Server Communication Across Firewalls

The OneClick web server must communicate with processes on the SpectroSERVER host system to gather data for display to the OneClick clients.

This communication is normally initiated by the OneClick web server, which establishes connections to specific SpectroSERVER host-side TCP ports for sending requests and receiving responses. However, there is a single OneClick listen port (default 14001), and the SpectroSERVER will initiate the connection to it, so it may be necessary to modify firewall configuration to allow this. The SpectroSERVER uses bidirectional IIOP (Internet Inter-ORB protocol) to communicate with its CORBA clients.

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



## 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>.

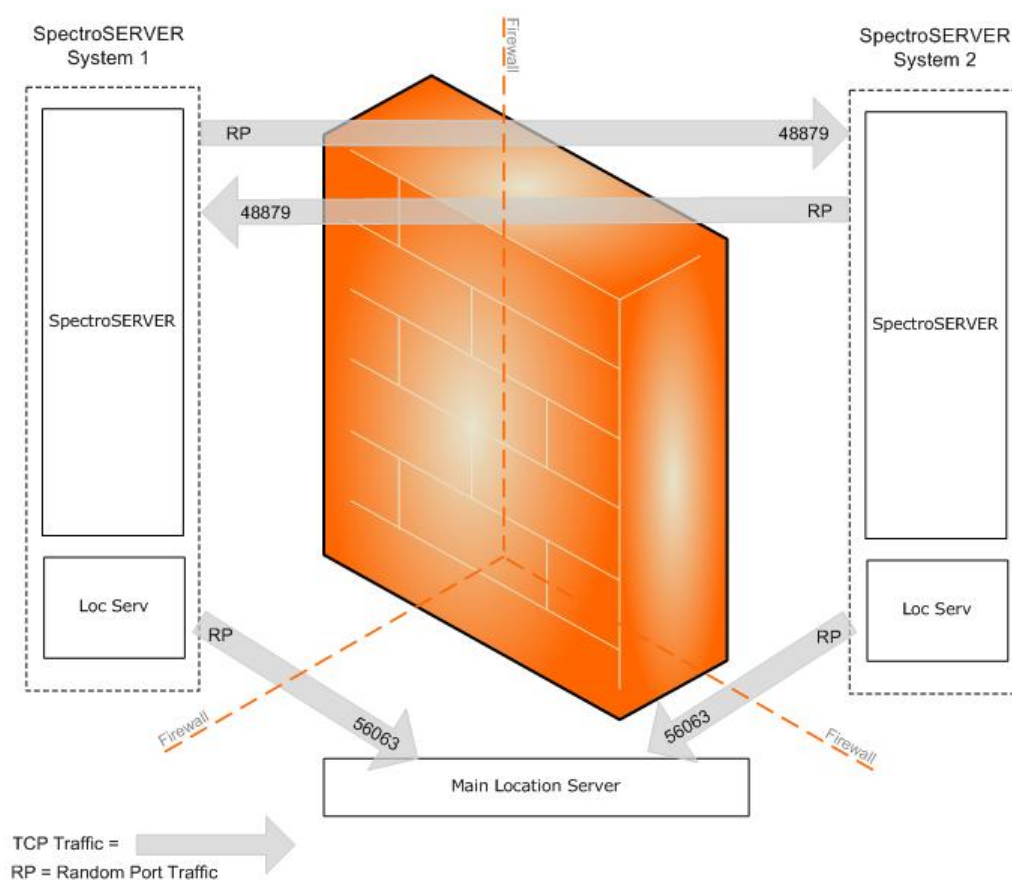
### CORBA Listen Port

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

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

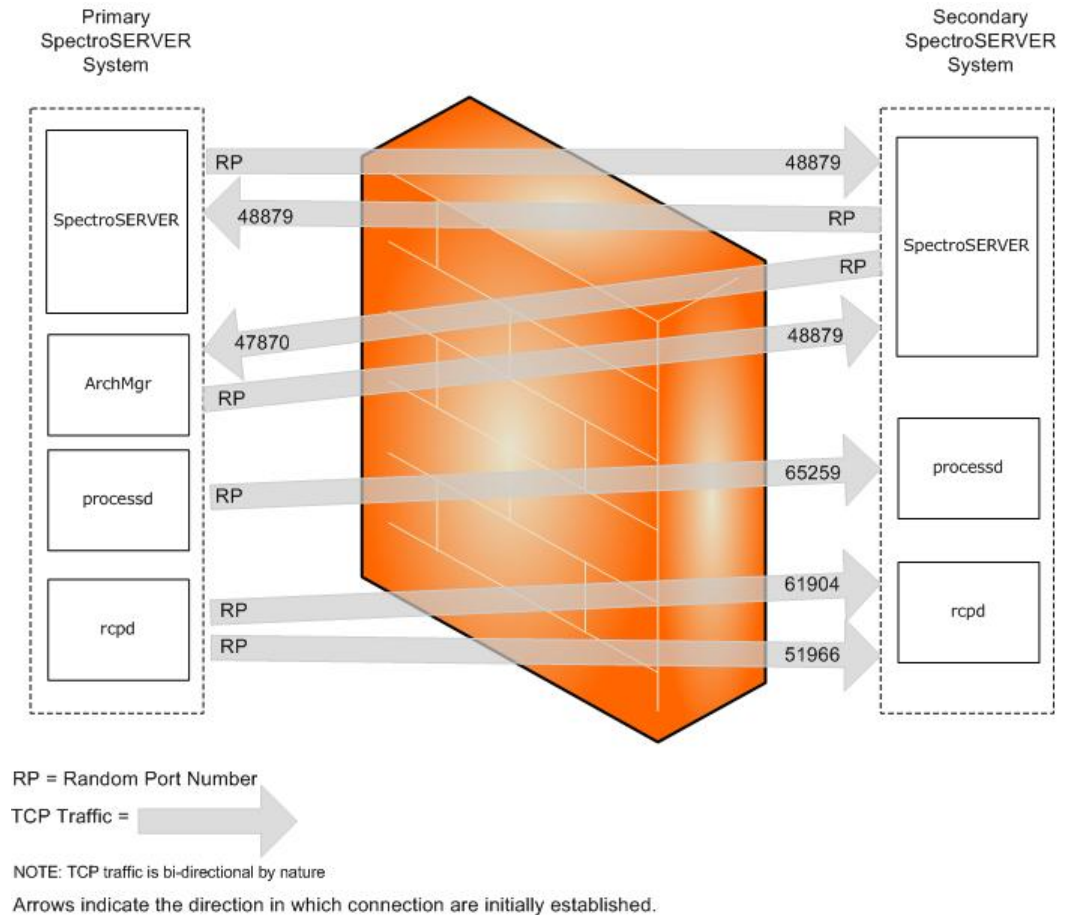
## Remote SpectroSERVERs and Firewalls

The following figure illustrates the IP connectivity required when two remote SpectroSERVERs must communicate through a firewall. In all cases where TCP is used, the connections are established from a random port to a specific/fixed port. The Main Location Server may or may not be located on another SpectroSERVER, and may or may not be situated beyond a firewall.



## Primary and Secondary SpectroSERVER Communication Across Firewalls

The following figure illustrates the IP connectivity required when primary and secondary SpectroSERVERs in a fault-tolerant environment must communicate over 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

Increasingly, Network Address Translation (NAT) is being used to construct networks with consistent internal IP addressing schemes and a single node handling IP address translation to the Internet. CA Spectrum now supports NAT, allowing CA Spectrum to be deployed into a private IP address domain and maintain connectivity to clients outside of NAT firewalls. Therefore, DSS environments can now encompass multiple domains separated by these firewalls.

The only requirement for a NAT environment is that the clients be able to resolve the server by name. On the private-side of the NAT, the hostname must resolve to the private-side IP. On the public-side of the NAT, the hostname must resolve to the public-side IP.

## Default Port Configurations

You may need to change the default port number or socket number used by a CA Spectrum process for the process to work correctly in a firewalled environment.

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

- [OneClick WebServer](#) (see page 59)
- [SpectroSERVER](#) (see page 63)
- [Archive Manager](#) (see page 63)
- [Location Server](#) (see page 63)
- [Naming Service](#) (see page 64)
- [Remote Copy Process Daemon \(rcpd\)](#) (see page 64)
- [CLI Daemon \(vnmsd\)](#) (see page 65)

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

## Change the SpectroSERVER Port Number

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

To change the port number the SpectroSERVER uses for CORBA requests, use a text editor to edit the `.vnmrc` file (located in `<$SPECROOT>/SS/`) to reflect the new port number:

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

### More information:

[Port Conflict Resolution](#) (see page 33)

## Change the Archive Manager Port Number and Socket Number

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

To change the port number the Archive Manager uses for CORBA requests, use a text editor to edit the `.configrc` file (located in `<$SPECROOT>/SS/DDM`) to reflect the new port number:

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

To change the socket number the Archive Manager uses to listen for requests from VNM and SSAPI clients, use a text editor to edit the `.configrc` file (located in `<$SPECROOT>/SS/DDM`) to reflect the following:

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

### More information:

[Port Conflict Resolution](#) (see page 33)

## Change the Location Server Port Number and Socket Number

You can change the port number and the socket number 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 (located in `<$SPECROOT>/LS`) to reflect the new port number:

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

To change the socket number the Location Server uses for VNM and SSAPI requests, use a text editor to edit the .locrc file (located in <\$SPECROOT>/LS) to reflect the following:

LOC\_SERVER\_SOCKET\_NUMBER=<new port number>

**More information:**

[Port Conflict Resolution](#) (see page 33)

## Change the Visibroker Naming Service Port Number

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

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

**To change the Visibroker Naming Service port number on Windows**

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 Solaris, set the environment variable in the spectrum60.env file (located in the /opt/SPECTRUM directory):

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 33)



## 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 33)



# Chapter 4: Fault Tolerance

---

This section contains the following topics:

[About Fault Tolerance](#) (see page 67)

[How to Establish Fault Tolerance](#) (see page 75)

[Validate Fault Tolerance Configuration](#) (see page 77)

[Test Fault Tolerance](#) (see page 78)

[Restart the Primary Archive Manager and Primary SpectroSERVER](#) (see page 78)

[Change the Host Names of the Primary and Secondary SpectroSERVERs](#) (see page 79)

[Monitor the Changeover Between the Primary and Secondary SpectroSERVERs](#) (see page 81)

[How to Monitor the Secondary SpectroSERVER Status](#) (see page 82)

## About Fault Tolerance

Fault tolerance is having more than one SpectroSERVER available for managing a given landscape. A copy of the database for that landscape is loaded on each SpectroSERVER, but only one copy is active at any time. The SpectroSERVER with the active database is called the primary SpectroSERVER. The inactive database resides on a standby SpectroSERVER referred to as the secondary SpectroSERVER (and, if desired, another inactive copy of the database can reside on a tertiary SpectroSERVER).

If the primary SpectroSERVER fails, the secondary SpectroSERVER's database becomes active and the secondary SpectroSERVER starts managing the network. Any applications connected to the primary SpectroSERVER are automatically switched to the secondary SpectroSERVER. When the primary SpectroSERVER comes back into service, the applications are automatically switched back to the primary SpectroSERVER, and the secondary SpectroSERVER becomes inactive again.

**Note:** The fact that applications can switch from one SpectroSERVER to another does *not* mean that all applications will be able to exercise the full range of their capabilities when they are being run from a secondary SpectroSERVER. The main purpose in setting up a fault-tolerant environment is to help ensure continuous monitoring of the network, not to create a full working 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 intended to manage the same landscape must all have the same landscape handle and the same modeling catalog. The primary and its standbys are distinguished from one another on the basis of a numeric precedence value, with the lowest number indicating the primary. SpectroSERVERs are installed with a default precedence value of 10. To designate a SpectroSERVER as a secondary, you must assign it a higher precedence number, for example, 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 75).

If you need to change precedence values later on, you can do so through the Loaded Landscapes subview, which is accessed 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, which is accessed by selecting the VNM in the Navigation panel and then selecting the Information tab in the Component Detail panel.

### More information:

[How to Establish Fault Tolerance](#) (see page 75)

## Data Synchronization

Since only one database is active at any given time in a fault tolerant CA Spectrum environment, the other databases must be updated periodically to reflect new models that have been created and attribute values that have changed in the active database. This synchronization of data is accomplished through CA Spectrum's Online Backup feature, which you can run 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 and automatically load the copy onto each of the designated secondary SpectroSERVERs.

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, but not all the .i files and other elements associated with individual management modules. Therefore, if you install any new management modules on your primary SpectroSERVER, you must also install the same new management modules on any secondary SpectroSERVERs.

**Note:** For more information about configuring Online Backup, see the *Database Management Guide*.

EventDisp and 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 to determine whether it is still up.

### 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. This is because the secondary SpectroSERVER has been brought down in order to load the new database from the primary SpectroSERVER.

If you want to generate this alarm only if the secondary SpectroSERVER is not restarted, you can do so by processing the alarm with an Event rule.

The EventPair rule allows you to specify that if the Contact Lost to Secondary Server (0x00010c0e) event occurs and is not followed by a Contact Established to Secondary Server (0x00010c0f) event within x number of seconds, a new event should be generated. You can then specify that this new event should create an event and an alarm indicating that the secondary SpectroSERVER is still down.

#### To use the EventPair rule to generate the alarm

1. Open the EventDisp file with a text editor.

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

2. 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 you would like to generate if the secondary SpectroSERVER does not come up within the time specified in `<numberofsecondstowait>`.

3. 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 will be logged and it will have a severity value of 50. A 2 indicates that a major alarm will be created. `<generatedalarmcode>` is the alarm code to be generated based on this event.

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

**Note:** For more information about creating a Probable Cause file, EventDisp files, and the EventPair rule, see the *Event Configuration User Guide*.

## SpectroSERVER Alarm Synchronization

The primary and secondary SpectroSERVERs use a Global Alarm Service (GAS) connection to get alarm information from each other. The SpectroSERVERs use the alarm information to synchronize alarms between each server. This synchronization helps to prevent duplicate alarms.

Options for fault tolerant alarm synchronization, including enablement of the service and whether to generate debug information, are controlled by settings in the .vnmrc file. For more information, see [Fault Tolerant Alarm Service \(.vnmrc\) Resources](#) (see page 23).

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:

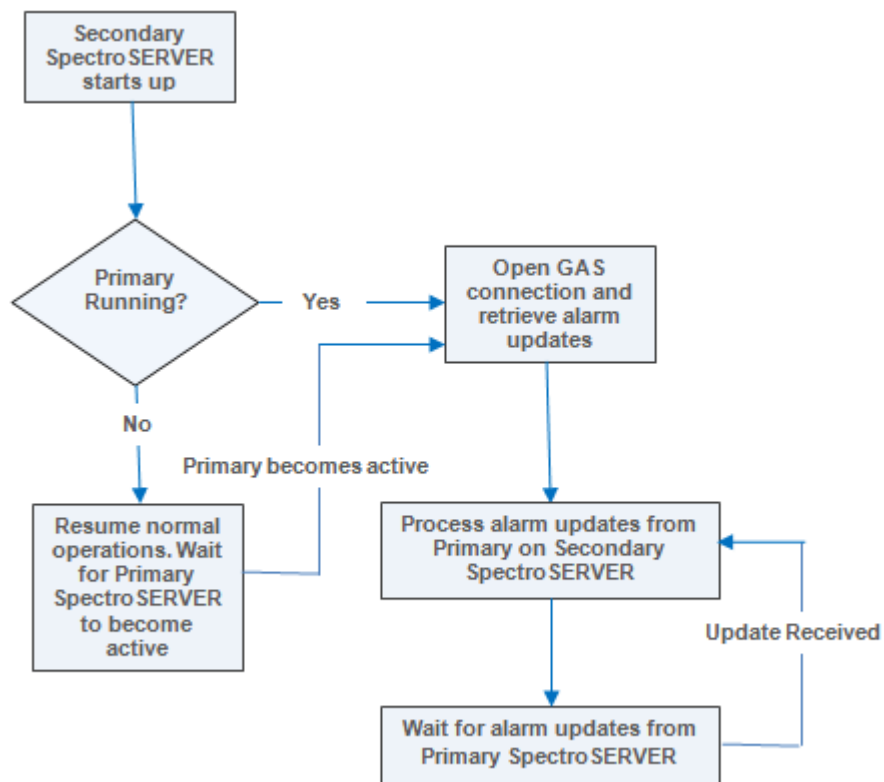
- [Synchronization from the Primary to the Secondary SpectroSERVER](#) (see page 70)
- [Synchronization from the Secondary to the Primary SpectroSERVER](#) (see page 72)

## 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 then remains connected to the primary and continues to receive alarm updates as they happen on the primary. If the connection fails, the secondary SpectroSERVER keeps trying every minute until the primary SpectroSERVER becomes active.

**Note:** For the secondary SpectroSERVER to attempt to connect to the primary SpectroSERVER for alarm synchronization, the Fault Tolerant Alarm Service must be enabled on both the primary and secondary servers. 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 23).

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 'stale'. The new alarm replaces the existing alarm if they are equal. You can determine whether the alarms are equal by verifying the following:
  - Unique Alarm (IDs)
  - Model Handle
  - Probable Cause
  - Alarm Discriminators

Two alarms are considered equal if they have the same ID's. Two alarms are also considered equal if they are on the same model (have the same model handle) and with the same probable cause unless they have different alarm discriminators.

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

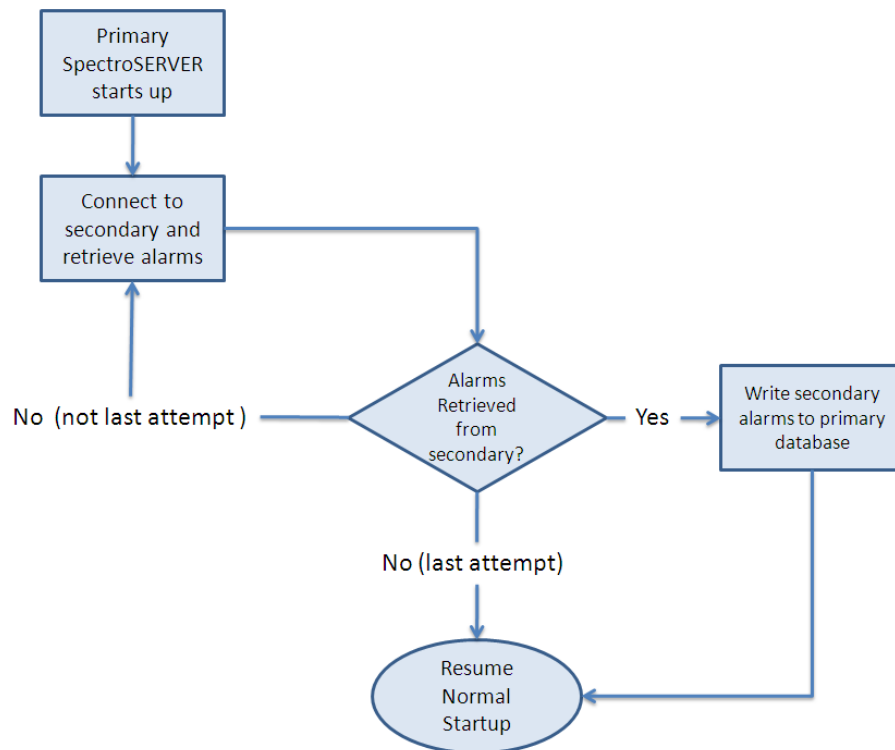
### Synchronization from the Secondary to the Primary SpectroSERVER

When the primary SpectroSERVER comes online after a failure or planned maintenance, it will connect to the secondary SpectroSERVER and get all alarms that exist on the secondary server. If the primary fails to connect on the first attempt, the primary can make multiple connection attempts to the secondary server to do the synchronization. The number and frequency of attempts is controlled by the `ftasv_max_conn_retry_count` and `ftasv_conn_retry_interval` parameters in the `.vnmrc` file of the primary SpectroSERVER. For more information, see [Fault Tolerant Alarm Service \(.vnmrc\) Resources](#) (see page 23).

**Important!** If the secondary SpectroSERVER is not running when the primary attempts to connect to it, the primary will exhaust its synchronization attempts and cause a delay in starting the primary. This delay must be endured because it happens before model activation. If this is not acceptable, make sure the secondary is running when the primary starts, or decrease the retry count or interval size to reduce the potential delay. You can also turn off 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 its alarms.



#### After a Successful Connection:

- After the list of alarms is retrieved, the primary SpectroSERVER writes the alarms to its database. Primary SpectroSERVER startup continues and these alarms are read from the database.
- Alarms currently stored in the primary database are replaced with the alarms from the secondary SpectroSERVER.
- All alarms read from the database are processed normally 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 was cleared on the secondary SpectroSERVER while the primary SpectroSERVER was down.

**Note:** There is no synchronization for blue and gray alarms. However, there is synchronization for brown alarms, just not for WA\_Link models. The WA\_Link model exception is only for brown alarms.

### After a Failed Synchronization:

- If alarm synchronization fails when the primary is attempting to connect to the secondary, the primary will still start up but an event and critical alarm (0x10c35) will be generated on the primary's LocalScope model. The event and alarm will give the reason for the failure of each attempt. If you want to continue running the primary SpectroSERVER after alarm synchronization has failed, it is recommended that you run an Online Backup to synchronize the databases between the primary and secondary SpectroSERVERs.

## Secondary SpectroSERVER Readiness Levels

Depending on how a secondary SpectroSERVER is configured and whether it is up and running, it is considered to be at one of three different levels of readiness.

The readiness levels are defined as follows:

### Hot

The secondary SpectroSERVER is up and 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, you must add the following line to its .vnmrc file:

secondary\_polling=yes. This causes the standby to commence polling and process traps whenever it is started up, regardless of whether it has lost contact with the primary SpectroSERVER.

### Warm

The secondary SpectroSERVER is up and running, but may take a short time to become fully available because it has not been configured to start polling *until* it loses contact with the primary SpectroSERVER (for example, it either has no secondary\_polling entry in its .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 will 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 makes no difference whether the secondary SpectroSERVER is configured for secondary polling.

### More information:

[How to Establish Fault Tolerance](#) (see page 75)

[Validate Fault Tolerance Configuration](#) (see page 77)

## How to 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 addresses a scenario in which you want to set up two SpectroSERVERs: a primary and a secondary. Should you also want to set up a tertiary SpectroSERVER, you would use the same method except you would assign the tertiary SpectroSERVER a higher precedence number than the secondary SpectroSERVER.

**Note:** If you are establishing fault tolerance in an environment that uses a Southbound Gateway integration, see the *Southbound Gateway Toolkit Guide* for more information.

### To establish fault tolerance

1. The same version of CA Spectrum with the same modeling catalog should be installed on both the machine you want to host your primary SpectroSERVER and the machine you want to host your secondary SpectroSERVER. Each of these should have the same landscape handle.
2. Make sure both the primary and secondary SpectroSERVER machines have entries in their .hostrc files giving the SpectroSERVERs access permissions for each other.

**Note:** If you are specifying secure users for the secondary SpectroSERVER in the primary SpectroSERVER machine's .hostrc file, and the secondary SpectroSERVER machine is running in the Windows environment, you must make sure to include the user SYSTEM in the secure user list.

3. Make sure that the MAIN\_LOCATION\_HOST\_NAME parameter in the secondary SpectroSERVER machine's .locrc file points to the same system name that the primary SpectroSERVER is pointing to. If this is not configured properly, the synchronization will fail.
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 will fail or will not run properly after an Online Backup is performed.
5. Make a copy of the primary SpectroSERVER's database, either by running Online Backup or, if the SpectroSERVER is shut down, by using the SSdbsave utility with the -cm argument (to save the modeling catalog as well as any models that have been created).

**Note:** For more information about configuring Online Backup, see the *Database Management Guide*.

6. Using the transport method of your choice, make sure the save file previously created is available to the machine you want to host your secondary SpectroSERVER.
7. On the secondary machine, with SpectroSERVER shut down, navigate to CA Spectrum's 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's default value of 10 (20 is recommended).

**savefile**


Specifies the name of the save file previously created.

8. If you want the secondary SpectroSERVER to function as a ["hot" backup](#) (see page 74), add the line `secondary_polling=yes` to its `.vnmrc` file.
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 information about using this command, see the *Database Management Guide*.

The secondary SpectroSERVER is now available to take over automatically in the event the primary SpectroSERVER fails. If you previously activated secondary polling, the secondary SpectroSERVER will be available immediately. If not, it will begin polling as soon as it detects 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 indicate the "switched" condition. 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"), and all applications switch back to the primary SpectroSERVER. However, any edits you make to the secondary SpectroSERVER while it is active will *not* be automatically copied back to the primary SpectroSERVER. You must manually recreate these modifications on the primary SpectroSERVER.

When you restart the primary SpectroSERVER, connections will be accepted as soon as all models are loaded, but *before* all models are activated. Since it may take a while for all the models to activate, and since the secondary SpectroSERVER will turn off polling as soon as the primary SpectroSERVER is restarted, there can be a gap in your network management coverage.

To avoid this situation, you can edit the primary SpectroSERVER's .vnmrc file so that the wait\_active resource is set to yes. This causes the server to wait until all of the models are activated before accepting any connections. It also causes the message area in the CA Spectrum Control Panel to dynamically display the percentage of models activated throughout the activation process. The SpectroSERVER may appear to take longer to come up, but 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. That way, for a planned shutdown of the primary SpectroSERVER, you will be able to monitor the CA Spectrum Control Panel to see exactly when the secondary SpectroSERVER is ready to take over.

**Note:** For more information about SSdbsave, SSdbload, and other database utilities, see the *Database Management Guide*.

**More information:**

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

[Designate a New Main Location Server](#) (see page 37)

[SpectroSERVER Precedence in a Fault Tolerant Environment](#) (see page 68)

## Validate Fault Tolerance Configuration

After configuring fault-tolerance, it is also important to verify that the OneClick server has access to both primary and secondary SpectroSERVERs, so that in the event of a failover that the OneClick server can properly failover to the secondary.

**To validate Fault Tolerance configuration**

1. Access the OneClick Administration, Landscapes web page.
2. View the 'Secondary Status' column to verify whether OneClick has established contact with secondary SpectroSERVER and whether Fault Tolerance is ready for failover.

The Fault Tolerance configuration is validated.


## Test Fault Tolerance

Upon an initial installation, it is possible for the secondary SpectroSERVER not to have access to all of the devices that the primary SpectroSERVER has access to. If this is the case, the secondary SpectroSERVER will generate false alarms. To avoid the generation of these alarms, you should be sure that the secondary SpectroSERVER can manage your network devices by testing fault tolerance.

**Note:** You should test fault tolerance whenever new devices are added to the primary SpectroSERVER.

### To test fault tolerance

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

The Connection Status icon  displays yellow to indicate the “switched” condition.

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

2. Allow 15 to 20 minutes for the secondary SpectroSERVER to run and then verify the following:
  - The Connection Status icon does not display red.
  - All device models and pingable models do not lose SNMP or ICMP contact. If this occurs, then it is possible that the secondary SpectroSERVER does not have access to manage your devices. This issue should be resolved with a network administrator. Verify that CA Spectrum is managing all devices that have an established contact state. This can be determined by looking to see if device contact or management contact loss alarms are generated by any of the device models.
3. Restart the primary SpectroSERVER.

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

## Restart the Primary Archive Manager and Primary SpectroSERVER

To prevent the loss of any event or statistical data in a fault tolerant SpectroSERVER environment, only one Archive Manager should be running. This primary Archive Manager should reside on the primary SpectroSERVER machine. With this configuration, if a failure occurs, no data will be lost.

There are two possible failure scenarios:

- If the primary SpectroSERVER stops operating, then the secondary SpectroSERVER will forward event and statistical information to the primary Archive Manager running on the primary SpectroSERVER machine. When the primary SpectroSERVER is restarted, no event and statistical data will have been lost.
- If the computer on which the primary SpectroSERVER and the primary Archive Manager reside stops operating completely, the secondary SpectroSERVER caches event and statistical data in its database.

You restart the primary Archive Manager and the primary SpectroSERVER if the machine they are running on goes down, or if the primary SpectroSERVER stops operating.

#### **To restart the primary Archive Manager and the primary SpectroSERVER**

1. Start the CA Spectrum Control Panel on the primary SpectroSERVER machine and select Start Archive Manager from the Control menu.

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

**Important!** If you start the primary SpectroSERVER before the secondary SpectroSERVER has forwarded the cached data, the data will be lost.

2. From the secondary SpectroSERVER, access the Event and Statistic Log Information view by highlighting the VNM icon and selecting Configuration from the Icon Subviews menu and clicking the Event/Statistic Logs button in the Landscape Configuration view.
3. Monitor the Event and Statistic Log Information view.

When the Locally Stored Events and Locally Stored Records fields each display a zero, the events and statistics cached in the secondary SpectroSERVER's database have been forwarded to the primary Archive Manager.

4. Restart the primary SpectroSERVER.

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

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

### To change the host name of the primary SpectroSERVER

1. Save the database using SSdbsave with the -cm option.
2. Make the host name change.
3. Reload the database with the save file created in the first step by running SSdbload with the -il option and the -replace option so that the database will associate the new host name with the precedence value (10) that designates a primary SpectroSERVER:

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

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, therefore, any SpectroSERVER configured as a warm standby would start polling.

4. To prevent this, 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.

### To change the host name of the secondary SpectroSERVER

1. Save the database using SSdbsave with the -cm option.
2. Make the host name change.
3. Reload the database with the save file created in the first step by running SSdbload with the -il option and the -replace option so that the database associates the new host name with the precedence value (20) that designates a secondary SpectroSERVER:

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

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

**Note:** For more information about SSdbsave, SSdbload, and other database utilities, 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. You can create a watch that alerts you when either the primary or the secondary SpectroSERVER is ready to take over.

### To monitor the changeover between the primary and secondary SpectroSERVERs, do the following:

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`

The watch must be set up as an active polled watch. Once it has been created on the primary SpectroSERVER, the primary SpectroSERVER must be synched with the secondary SpectroSERVER so that the watch exists on the secondary server.

3. Specify a value for the Model\_Name (0x1006e) attribute in the watch expression, if you want to be notified 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 about creating a watch, see the *Watches User Guide*.

4. To limit the potential for false events or alarms, add the following line to the secondary SpectroSERVER's .vnmrc file:

`is_secondary = TRUE`

This setting allows the secondary SpectroSERVER to drop events unless CA Spectrum's internal mechanisms have determined 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.

### To monitor the status of the secondary SpectroSERVER

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>)
```

The watch must be set up as an active polled watch. Once it has been created on the primary SpectroSERVER, the primary SpectroSERVER must be synched with the secondary SpectroSERVER so that the watch exists on the secondary server.

**Note:** For more information about creating a watch, see the *Watches User Guide*.

2. To limit the potential for false events or alarms, add the following line to the secondary SpectroSERVER's .vnmrc file:

```
is_secondary = TRUE
```

This setting allows the secondary SpectroSERVER to drop events unless CA Spectrum's internal mechanisms have determined 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 83)

[Trap Data Traffic Consolidation](#) (see page 83)

## Trap Director

Trap Director is a SpectroSERVER feature you can enable when you want a given SpectroSERVER to forward incoming traps to models on remote landscapes in a distributed SpectroSERVER environment. Use of Trap Director requires that the Trap Director server be specified as the NMS (network management station) recipient for traps from those devices modeled in the remote landscapes.

**Note:** SNMPv3 traps will not be processed by a SpectroSERVER running as a trap forwarder because the SNMPv3 security credentials cannot be validated.

## Trap Data Traffic Consolidation

As your managed network grows and you decide to apportion models among additional landscapes, you might want to retain the original landscape host as the trap destination for the models to avoid having to reconfigure new trap destinations on the devices they represent. Conversely, you may simply want to eliminate existing multiple trap destinations and designate a single SpectroSERVER to receive and route traps. If load-sharing considerations are an important influence on how you manage network devices, you can enable Trap Director on multiple servers. Each server in this case handles traps from a set of devices configured to send traps to it.

Trap Director maintains an up-to-date model address cache and uses it to match trap sources with models. It forwards traps to destination models on remote landscapes based on the matches. By keeping its cache information current, Trap Director helps ensure that CA Spectrum is able to generate events for models regardless of where they are located.

**Note:** You may notice decreased Trap Director performance (for example, trap notification latency) when new landscapes are added to the distributed environment or when very large numbers of traps are processed for many new models.

## How Trap Director Updates the Address Cache

Trap Director maintains an address cache that includes the IP addresses and locations of models in the distributed environment. The address cache serves as an index, which Trap Director uses to determine where to forward traps. Because models may be added to, removed from, and moved between landscapes on a regular basis, Trap Director must also update the cache on a regular basis to keep its content current. It does this by removing records that meet a retention period (or aging) threshold that you can specify and by doing cross-landscape searches for model IP addresses that are not included in (and are no longer in) the cache when it receives traps from those IP addresses.

Trap Director determines the destination model for a trap by doing the following:

1. When the Trap Director server receives a trap, it compares the IP address included in the trap to IP addresses in cache records.
2. If it finds a match, it forwards the trap to the model (on the remote landscape).

If it does not find a matching IP address in the cache, Trap Director does the following to determine the destination model and forward the trap:

- Trap Director searches known landscapes for a matching IP address.
  - When it finds the destination model associated with the IP address on the remote landscape, it updates the cache with the model information and forwards the trap to the model.
3. If Trap Director cannot find a matching address (for example, the model has been deleted) during its cross-landscape search and must drop the trap, CA Spectrum generates an event on the VNM model on the Trap Director server indicating that the destination model could not be found.

## 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 Fault Tolerance](#) (see page 67)

## Trap Storm Settings

Trap Director uses trap storm settings configured on modeled devices. When Trap Director detects a trap storm, it stops forwarding traps to models on remote landscapes and asserts trap storm alarms for the models. For trap storms originating from devices not modeled in CA Spectrum, Trap Director asserts the trap storm handling settings 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 server's VNM model.

### To enable Trap Director

1. Expand the SpectroSERVER Control subview in the VNM model's Information tab.
2. Click 'set' in the Enable Trap Director field and select Enabled.

Trap Director is enabled.

### To disable Trap Director

1. Expand the SpectroSERVER Control subview in the VNM model's Information tab.
2. 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 Trap Director's trap cache ages-out by defining 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 • 36, 52  
.LocalRegFile • 35  
.locrc file • 35, 36, 63  
.vnmrc file • 12, 82

## A

address cache • 84  
ADMINPRIVS • 42  
alarm synchronization • 70  
alarms • 69  
APPNAME • 42  
ARGV • 42  
AUTOBOOTSTART • 42  
AutoDiscovery • 50  
AUTORESTART • 42

## B

bind\_retry\_interval • 13

## C

CA Spectrum  
    distributed installation • 57  
    domain • 38  
changing  
    Archive Manager port number and socket  
        number • 63  
    landscape handles • 39  
    location server port number and socket  
        number • 63  
    SpectroSERVER host names • 79  
    SpectroSERVER port number • 63  
    VisiBroker Naming Service port number • 64  
    Windows password in processd • 41  
CLI Daemon (.vnmsd) • 65  
cold • 74  
comm\_port • 13  
connect\_time\_limit • 13

## D

default port numbers • 33, 62  
default socket numbers • 33, 62  
device\_limit • 13

disable\_redundancy\_when\_using\_loopback • 13  
Distributed SpectroSERVER  
    about • 9  
    requirements • 34  
duplicate models • 51

## E

enable\_traps\_for\_pingables • 13  
ENV • 42  
event\_batch\_max\_size • 20  
event\_batch\_timeout • 20  
event\_record\_increment • 20  
EventDisp file • 69  
EventPair rule • 69  
Events Archive (.vnmrc) Resources • 20  
expiration\_date • 13

## F

fault tolerance  
    about • 67  
    and Trap Director setup • 84  
    environment • 61, 68  
    establishing • 75  
    status monitoring • 81, 82  
    testing • 78  
firewalls  
    and remote SpectroSERVERs • 60  
    communicating across • 57

## G

GAS • 70  
general SpectroSERVER (.vnmrc) resources • 13

## H

handshake\_timeout • 13  
home landscapes • 51  
host security • 52  
hot • 74

## I

IIOP (Internet Inter-ORB protocol) • 58  
install tickets • 42, 45, 48

---

## L

- landscape handles
  - about • 34, 38
  - assigning • 39
  - changing • 39
- landscape map • 11, 38
- landscapes • 10, 34
- lh\_set\_utility • 39
- location servers • 35
- log\_user\_events • 20
- LOGNAMEPATH • 42

## M

- main location server • 35, 36, 37, 51, 60
- MAIN\_LOCATION\_HOST\_NAME • 36
- max\_bind\_retry\_count • 13
- max\_connections • 13
- max\_device • 13
- max\_event\_records • 20
- max\_total\_work\_threads • 21
- min\_client\_version • 13
- Model\_Name attribute • 81
- modeling catalog • 11, 34
- multiple interfaces • 31
- multiple landscapes • 11
- multiple SpectroSERVERs • 32

## N

- name resolution • 31
- Network Address Translation (NAT) • 62
- network models • 50
- network partitioning • 50
- NUMPROCS • 42

## O

- OneClick
  - default ports and firewalls • 59
  - web server • 58, 59
- online backup • 68

## P

- PARTNAME • 42
- partslist directory • 42
- password • 13
- PausePolling attribute • 82
- PercentInitialized attribute • 81
- persistent\_alarms\_active • 13
- port conflicts • 33

- primary SpectroSERVER • 78
- procd\_comm\_port • 13
- processd
  - about • 40
  - and the userconf process • 49, 50
  - changing the Windows password in • 41
  - configuration • 42
  - in Solaris environments • 41
  - in Windows environments • 41, 53
  - restarting • 48
  - shutdown • 53
  - stopping • 48
- processd.pl • 48
- processd\_log file • 40
- processd\_log.bak file • 40
- processd\_shutdown\_timeout • 53
- production environments • 38

## R

- rcpd\_comm\_port • 13, 64
- readiness levels • 74
- remote copy process daemon (rcpd) • 64
- resource\_file\_path • 13
- restarting
  - primary Archive Manager • 78
  - primary SpectroSERVER • 78
- RETRYMAX • 42
- RETRYTIMEOUT • 42

## S

- secondary SpectroSERVER • 74, 78
- secondary\_polling • 74
- SERVERPROCESS • 42
- SERVICE • 42
- snmp\_comm\_port • 13
- snmp\_trap\_port • 13
- snmp\_trap\_port\_enabled • 13
- SpectroSERVERs
  - across firewalls • 61
  - and firewalls • 60
  - and OneClick Web Server communication • 59
  - changeover • 81
  - monitoring • 82
  - multiple • 34
  - precedence • 68
  - shutdown • 53
- SSdbload utility • 39, 68, 79
- STARTPRIORITY • 42



---

STATEBASED • 42

## T

tcp\_buffer\_size • 13  
test environments • 38  
TICKETUSER • 41, 42  
time zones • 52  
TL1 gateway agent • 33  
trap data traffic • 83  
Trap Director • 83, 84, 85  
trap storm settings • 85

## U

unsupported\_attr\_poll\_interval • 13  
use\_log\_queue • 20  
user models • 11, 36  
userconf process • 49, 50  
Users Group • 49, 50

## V

Visibroker Naming Service • 64  
vnm\_close\_timeout • 13  
vnm\_file\_path • 13  
vnm\_message\_timeout • 13  
vsh\_tcp\_port parameter • 65

## W

wait\_active • 13  
WaitToKillServiceTimeout • 53  
warm • 74  
watches • 81, 82  
work thread (.vnmrc) resources • 21  
work\_thread\_age • 21  
WORKPATH • 42