

IBM® Network Dispatcher



User's Guide

Version 3.0 for Multiplatforms

IBM® Network Dispatcher



User's Guide

Version 3.0 for Multiplatforms

Note

Before using this information and the product it supports, be sure to read the general information under “Appendix F. Notices” on page 275.

Sixth Edition (May 2000)

© Copyright International Business Machines Corporation 2000. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Tables	ix	What are the components of IBM Network Dispatcher?	28
Figures	xi	Overview of Dispatcher	28
Welcome!	xiii	Overview of Interactive Session Support (ISS)	29
Chapter 1. Getting started...quickly!	1	Overview of Content Based Routing (CBR)	29
What you will need?	2	How about high availability?	30
How do you prepare?	2	Dispatcher	30
Configuring the Dispatcher component	3	ISS	30
Configuring using the command line	3	Which component should I use: Dispatcher, ISS, CBR or all three?	31
Configuring using the configuration wizard	4	Managing local servers with Dispatcher	33
Configuring using the graphical user interface (GUI)	5	Managing local servers with ISS	34
General Instructions for using the GUI	5	Managing local servers with CBR	35
Testing your configuration	6	Managing local servers with ISS and Dispatcher	37
Chapter 2. Installing IBM Network Dispatcher	9	Managing local servers with Server Monitor Agent (SMA) and Dispatcher	38
Requirements for AIX	9	Managing local and remote servers with Dispatcher	39
Installing for AIX	10	Managing local and remote servers with ISS and Dispatcher	40
Before you install	11	Chapter 4. Planning for the Dispatcher component	41
Installation steps	11	Hardware and software requirements	41
Requirements for Red Hat Linux	14	Requirements for AIX	41
Installing for Red Hat Linux	14	Requirements for Red Hat Linux	41
Before you install	14	Requirements for Solaris	42
Installation steps	14	Requirements for Windows 2000 and Windows NT.	42
Requirements for Solaris	17	Planning considerations	43
Installing for Solaris	17	High availability	48
Before you install	17	Simple high availability	48
Installation steps	17	Mutual high availability	49
Requirements for Windows 2000 or Windows NT	19	Using Collocated Servers.	50
Installing for Windows 2000 or Windows NT	19	Load Balancing Dispatcher with ISS	51
Installation Packages	20	Chapter 5. Configuring the Dispatcher component	53
Before you install	20	Overview of configuration tasks	53
Installation steps	21	Methods of configuration	53
Chapter 3. Introducing IBM Network Dispatcher	23	Command line	53
What is IBM Network Dispatcher?	23	Scripts	53
Why do I need IBM Network Dispatcher?	24	GUI.	54
What's new in this version?	26		

Configuration Wizard	55
Setting up the Dispatcher machine	55
Step 1. Start the server function	56
Step 2. Start the executor function.	56
Step 3. Define the nonforwarding address (if different from hostname).	56
Step 4. Define a cluster and set cluster options.	57
Step 5. Alias the network interface card	57
Step 6. Define ports and set port options	59
Step 7. Define load-balanced server machines	59
Step 8. Start the manager function (optional)	60
Step 9. Start the advisor function (optional)	60
Step 10. Set manager proportions as required	60
Setting up server machines for load balancing	60
Step 1. Alias the loopback device	61
Step 2. Check for an extra route	64
Step 3. Delete any extra route	65
Installing the Linux kernel patch (for aliasing the loopback device)	65

Chapter 6. Planning for the Content Based

Routing component	67
Hardware and software requirements	67
Requirements for AIX.	67
Requirements for Red Hat Linux	67
Requirements for Solaris	68
Requirements for Windows 2000 and Windows NT.	68
Planning considerations	69
CBR proxy (for IMAP or POP3)	70
CBR with WTE (for HTTP)	71

Chapter 7. Configuring the Content Based

Routing component	73
Overview of configuration tasks	73
Methods of configuration	73
Command line	73
Scripts	74
GUI.	74
Configuration wizard	75
Setting up the CBR machine	75
Step 1. Configure WTE to use CBR (for HTTP only)	76
Step 2. Start WTE (for HTTP only)	78
Step 3. Define a cluster and set cluster options.	78

Step 4. Define ports and set port options	78
Step 5. Define load balanced server machines	79
Step 6. Add rules to your configuration (for HTTP only).	79
Step 7. Add servers to your rules (for HTTP only)	79
Step 8. Start the manager function (optional)	80
Step 9. Start the advisor function (optional)	80
Step 10. Set manager proportions as required	80
CBR configuration example	80

Chapter 8. Advanced Dispatcher and CBR

Functions	83
Optimizing the load balancing provided by Dispatcher and CBR	84
Proportion of importance given to status information	85
Weights	86
Manager intervals	86
Advisor intervals	87
Advisor timeouts	87
Sensitivity threshold	88
Smoothing index	88
High availability	89
Configure high availability	89
Failure detection capability using heartbeat and reach target.	91
Recovery Strategy	92
Using scripts.	92
Configure wide area Dispatcher support	94
Command Syntax	95
Using remote advisors with wide area support	96
Configuration example	97
Notes	99
ISS	99
Advisors	99
How advisors work	100
Create custom (customizable) advisors.	102
WebSphere Application Server (WAS) advisor	103
Naming Convention	103
Compilation	103
Run	104
Required routines.	104
Search order	105
Naming and path.	105

Sample advisor	105
Workload Manager advisor	105
ISS and SMA Restriction	106
Server Monitor Agent (SMA)	106
WLM Restriction	107
Prerequisites	107
How to Use Server Monitor Agent (SMA)	107
Configure rules-based load balancing	108
How are rules evaluated?	109
Using rules based on the client IP address	109
Using rules based on the time of day	109
Using rules based on the connections per second on a port	110
Using rules based on the active connections total on a port	110
Using rules based on the client port	111
Using rules that are always true	111
Using rules based on type of service (TOS)	111
Using rules based on the request content	112
Adding rules to your configuration	112
Using explicit linking	113
Using a private network configuration	113
Use wildcard cluster to combine server configurations	114
Use wildcard cluster to load balance firewalls	115
Use wildcard cluster with WTE for transparent proxy	116
Use wildcard port to direct unconfigured port traffic	116
How classical ND affinity works	116
Behavior when disable affinity	117
Behavior when enable affinity	117
Server Directed Affinity API to control client-server affinity	117
Cross port affinity	118
Affinity address mask	119
Rule affinity override	119
Cookie affinity	120
How to enable cookie affinity	121
Why use cookie affinity	121
Normal client IP affinity	121
Using binary logging to analyze server statistics	121

Chapter 9. Planning for the Interactive Session Support component	125
Hardware and software requirements	125
Requirements for AIX	125

Requirements for Solaris	126
Requirements for Windows 2000 and Windows NT	126
Planning Considerations	127
Setting up ISS cells	127
Interfaces: an explanation	134
Using ISS and Dispatcher in a two-tiered configuration	136
Configuring highly available Dispatchers in a two-tier configuration	137
Ping Triangulation Considerations	139
Using Affinity (StickyTime) with ISSNameServer	140

Chapter 10. Configuring the Interactive Session Support component	143
Overview of configuration tasks	143
Methods of configuration	143
Modify iss.cfg file or an example file	144
ISScontrol commands	144
ISS GUI	144
Updating the ISS configuration file	146
How a configuration file is created	146
Defining a Cell	146
Defining a Node	147
Defining a Resource	148
Defining a Service	149
Defining an Observer	150
Migrating from an earlier version	150
Setting up the TCP/IP name server	151
Setting up a subdomain for your ISS service DNS names	152
Configuring the ISS client machines	156
Do I need an ISS backup?	157
Starting ISS	157

Chapter 11. Operating and managing IBM Network Dispatcher	159
Using the Dispatcher component	159
Starting Dispatcher	159
Stopping the Dispatcher	159
Using FIN count to control garbage collection	159
Using Dispatcher logs	160
Reporting GUI	161
Remote Authenticated Administration	161
Using Simple Network Management Protocol with the Dispatcher	162
SNMP commands and protocol	163
Enabling SNMP on AIX and Solaris	164

Enabling SNMP on Windows	165
Providing a security password for SNMP	166
Traps	166
Turning the SNMP support on and off from the ndcontrol command	167
Using the Interactive Session Support component	167
Operating and managing Interactive Session Support	167
Controlling ISS.	171
Using the Content Based Routing component	171
Starting and Stopping CBR	171
Using CBR logs	172

Chapter 12. Troubleshooting 173

Troubleshooting tables	173
Checking Dispatcher port numbers	176
Checking ISS port numbers	176
Checking CBR port numbers	177
Solving common problems—Dispatcher	178
Problem: Dispatcher will not run.	178
Problem: Dispatcher and server will not respond	178
Problem: Dispatcher requests are not being balanced.	178
Problem: Dispatcher high-availability function is not working	178
Problem: Unable to add heartbeat (Windows)	178
Problem: Extra routes (Windows only)	179
Problem: Advisors not working correctly	179
Problem: SNMPD does not run correctly (Windows only)	179
Problem: Dispatcher, Microsoft IIS, and SSL do not work (Windows only)	179
Problem: Dispatcher connection to a remote machine	180
Problem: ndcontrol or ndadmin command fails	180
Problem: Network Dispatcher machine locks up when attempting to collocate	180
Problem: "Cannot find the file..." error message when trying to view online Help	180
Solving common problems—CBR	181
Problem: CBR will not run.	181
Problem: Server not responding (CBR w/WTE).	181
Problem: Syntactical or configuration error	181

Problem: The cbrserver command is stopped	182
Problem: Receive CBR proxy error when trying to add a port	182
Solving common problems—ISS	182
Problem: ISS will not run	182
Problem: Starting ISS from the Services panel produces an error.	182
Problem: Unable to get ISS GUI to "Connect to Host"	182
Problem: All interactive sessions for ISS are assigned to one machine	183
Problem: ISS trace shows "Cannot rename 'file 1' to 'file 2'. Permission denied."	183
Problem: ISS is not recognising the isscontrol or GUI commands	183
Problem: Log file reports "Config: Could not find local node in the config file"	184
Problem: ISS is sending loads to Dispatcher, but Dispatcher is reporting the wrong values	184
Using ISS trace and debug facilities.	184
List of some common debug statements	185

Appendix A. How to read a syntax

diagram.	187
Symbols and punctuation	187
Parameters	187
Syntax examples	188

Appendix B. Command reference for

Dispatcher and CBR 191

Configuration differences between CBR and Dispatcher	191
ndcontrol advisor — control the advisor	193
ndcontrol cluster — configure clusters	197
ndcontrol executor — control the executor	200
ndcontrol file — manage configuration files	204
ndcontrol help — display or print help for this command	206
ndcontrol highavailability — control high availability	207
ndcontrol host — configure a remote machine	211
ndcontrol log — control the binary log file	212
ndcontrol manager — control the manager	213
ndcontrol port — configure ports	219
ndcontrol rule — configure rules.	223
ndcontrol server — configure servers	228
ndcontrol set — configure server log	231

ndcontrol status — display whether the manager and advisors are running	232
ndcontrol subagent — configure SNMP subagent.	233
Appendix C. Command reference for ISS	235
start iss	236
isscontrol — control iss	237
Configuration file keywords for ISS.	238
Appendix D. Rule types for CBR	249
Rule (pattern) syntax:	249
Reserved keywords	249
Appendix E. Sample configuration files	251
Sample Network Dispatcher configuration files	251
Dispatcher Configuration file—AIX, Red Hat Linux, and Solaris	251
Dispatcher Configuration file—Windows Sample advisor	255
Sample ISS configuration files.	263
Dispatcher Sample configuration file for ISS.	263
Replace DNS Sample configuration file for ISS	265
A Two Tier Sample configuration file for ISS.	267
Ping Triangulation Sample configuration file for ISS	270
Appendix F. Notices	275
Trademarks	276
Glossary	277
Index	285

Tables

1. AIX installp images.	10	8. Advanced configuration tasks for the Dispatcher and CBR functions	83
2. AIX install commands	12	9. ISS configuration methods	131
3. Guidelines for using the components of IBM Network Dispatcher	31	10. ISS Metric types	132
4. Configuration tasks for the Dispatcher function	53	11. Configuration tasks for the ISS function	143
5. Commands to alias the loopback device (lo0) for Dispatcher.	61	12. Cell Attributes	147
6. Commands to delete any extra route for Dispatcher.	65	13. Guidelines for setting up the name server used by ISS.	151
7. Configuration tasks for the CBR component	73	14. Dispatcher troubleshooting table	173
		15. CBR Troubleshooting table	174
		16. ISS troubleshooting table	175

Figures

1. A simple Dispatcher configuration . . .	1	14. Example of the Dispatcher configured with 2 clusters, each with 2 ports . . .	47
2. The graphical user interface (GUI) . . .	6	15. Example of a Dispatcher using simple high availability	48
3. Example of a logical representation of a site using the Dispatcher to manage local servers	33	16. Example of a Dispatcher using mutual high availability	49
4. Example of a physical representation of a site using the Dispatcher to manage local servers	34	17. Example of the IP addresses needed for the Dispatcher machine	56
5. Example of a site using ISS to manage local servers	35	18. CBR configuration file for AIX . . .	76
6. Example of a site using CBR to manage local servers	36	19. CBR configuration file for Red Hat Linux	76
7. Example of a site using the Dispatcher and ISS to manage local servers . . .	37	20. CBR configuration file for Solaris . . .	77
8. Example of a site using the Dispatcher and SMA to manage local servers . . .	38	21. CBR configuration file for Windows . . .	77
9. Example of a site using Dispatcher to manage local and remote servers . . .	39	22. Example of a configuration consisting of a single LAN segment.	94
10. Example of a site using ISS and Dispatcher to manage local and remote servers	40	23. Example of configuration using local and remote servers	95
11. Dispatcher product components . . .	44	24. Wide area example configuration . . .	97
12. Example of Dispatcher configured with a single cluster and 2 ports	45	25. Example of a private network using Dispatcher	114
13. Example of Dispatcher configured with two clusters, each with one port . . .	46	26. Example of the ISS component . . .	127
		27. A diagram of an ISS interface	134
		28. Example of a two-tiered configuration using IBM Network Dispatcher . . .	136
		29. SNMP commands for AIX and Solaris . . .	164
		30. SNMP commands for Windows . . .	165

Welcome!

This book explains how to plan for, install, configure, use, and troubleshoot IBM® Network Dispatcher for AIX, Red Hat Linux, Solaris, Windows NT, and Windows 2000. Previously, this product was called SecureWay Network Dispatcher, eNetwork Dispatcher, and Interactive Network Dispatcher.

The most current version of this book is available in HTML and PDF formats on the IBM Network Dispatcher Web site. To access the online book, go to the following URL:

<http://www.software.ibm.com/network/dispatcher/library/>

The IBM Network Dispatcher Web site gives you the latest details on how you can use this product to maximize the performance of your servers. Configuration examples and scenarios are included. To access this Web site, go to the following URL:

<http://www.software.ibm.com/network/dispatcher>

Chapter 1. Getting started...quickly!

How quickly can you make IBM Network Dispatcher work for you? Consider the following:

Assume you're the webmaster for the Intersplash Corporation. You manage a local web site with two HTTP servers. You've been using a round-robin approach to manage the load on the two servers, but business has picked up recently and customers are starting to complain about not being able to access the site. What do you do?

Go out to <http://www.software.ibm.com/network/dispatcher> and download the latest version of IBM Network Dispatcher. This product has three components: Dispatcher, Interactive Session Support, and Content Based Routing. For the time being, we'll just discuss the **Dispatcher** component.

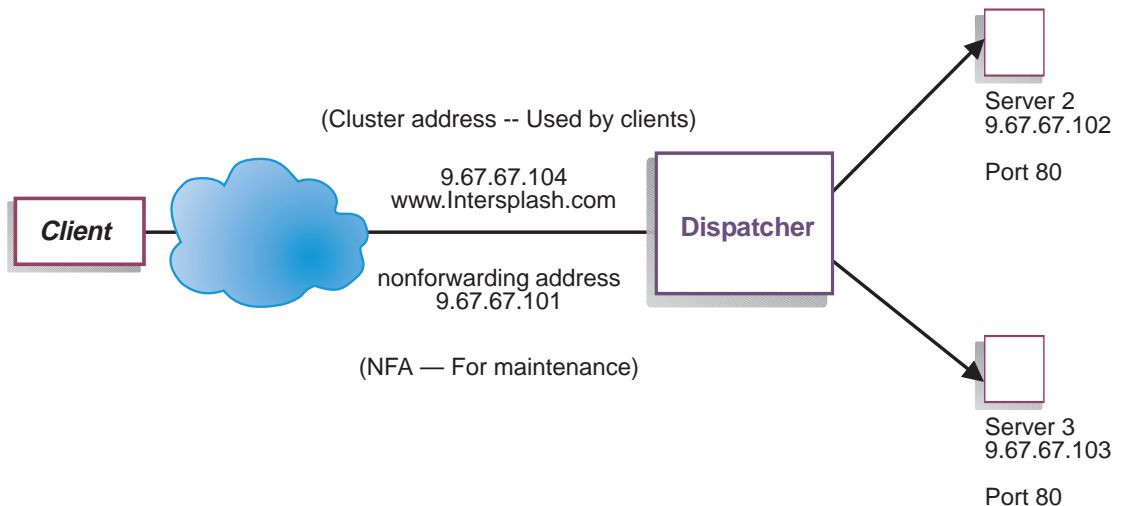


Figure 1. A simple Dispatcher configuration

This quick-start example shows how to configure three workstations using the Dispatcher component to load-balance web traffic between two web servers. The configuration would be essentially the same for balancing any other TCP or stateless UDP application traffic.

Note: With the AIX, Red Hat Linux, or Solaris version of Dispatcher, the configuration could be completed using only two workstations with Dispatcher located on one of the web server workstations. This represents a collocated configuration. Procedures for setting up more complex configurations can be found at “Setting up the Dispatcher machine” on page 55.

What you will need?

For the quick-start example, you will need three workstations and four IP addresses. One workstation will be used as the Dispatcher; the other two workstations will be used as web servers. Each web server requires one IP address. The Dispatcher workstation requires one actual address, and one address to be load balanced.

How do you prepare?

1. Ensure you have the prerequisites, listed in “Chapter 2. Installing IBM Network Dispatcher” on page 9.
2. Set up your workstations so that they are on the same LAN segment. Ensure that network traffic between the three machines does not have to pass through any routers or bridges.
3. Configure the network adapters of the three workstations. For this example, we will assume you have the following network configuration:

Workstation	Name	IP Address
1	server1.intersplash.com	9.67.67.101
2	server2.intersplash.com	9.67.67.102
3	server3.intersplash.com	9.67.67.103
Netmask = 255.255.255.0		

Each of the workstations contains only one standard Ethernet network interface card.

4. Ensure that server1.intersplash.com can ping both server2.intersplash.com and server3.intersplash.com.
5. Ensure that server2.intersplash.com and server3.intersplash.com can ping server1.intersplash.com.
6. Ensure that content is identical on the two web servers. This can be done by replicating data on both workstations, by using a shared file system such as NFS, AFS, or DFS, or by any other means appropriate for your site.

7. Ensure that web servers on server2.intersplash.com and server3.intersplash.com are operational. Use a web browser to request pages directly from **http://server2.intersplash.com** and **http://server3.intersplash.com**.
8. Obtain another valid IP address for this LAN segment. This is the address you will provide to clients who wish to access your site. For this example we will use:
Name= www.intersplash.com
IP=9.67.67.104
9. Configure the two web server workstations to accept traffic for www.intersplash.com.
Add an alias for www.intersplash.com to the **loopback** interface on server2.intersplash.com and server3.intersplash.com.
 - For AIX:
ifconfig lo0 alias www.intersplash.com netmask 255.255.255.0
 - For Solaris:
ifconfig lo0:1 www.intersplash.com 127.0.0.1 up
 - For other operating systems see Table 5 on page 61.
10. Delete any extra route that may have been created. See “Step 2. Check for an extra route” on page 64.
You have now completed all configuration steps that are required on the two webserver workstations.

Configuring the Dispatcher component

With Dispatcher, you can create a configuration by using the command line, the configuration wizard, or the graphical user interface (GUI).

Configuring using the command line

If you are using the command line, follow these steps:

1. Start the ndserver on Dispatcher:
 - For AIX, Red Hat Linux, or Solaris, run the following command as root:
ndserver
 - For Windows, ndserver runs as a service that starts automatically.
2. Start the executor function of Dispatcher:
ndcontrol executor start
3. Add the cluster address to the Dispatcher configuration:
ndcontrol cluster add www.intersplash.com
4. Add the http protocol port to the Dispatcher configuration:
ndcontrol port add www.intersplash.com:80

5. Add each of the web servers to the Dispatcher configuration:
ndcontrol server add www.intersplash.com:80:server2.intersplash.com
ndcontrol server add www.intersplash.com:80:server3.intersplash.com
6. Configure the workstation to accept traffic for the cluster address:
ndcontrol cluster configure www.intersplash.com
7. Start the manager function of Dispatcher:
ndcontrol manager start
Dispatcher will now do load balancing based on server performance.
8. Start the advisor function of Dispatcher, and tell the manager to use the advisor information:
ndcontrol advisor start http 80
ndcontrol manager proportions 60 39 1 0

Dispatcher will now make sure that client requests are not sent to a failed web server. Your basic configuration is now complete.

Configuring using the configuration wizard

If you are using the configuration wizard, follow these steps:

1. Start the ndserver on Dispatcher:
 - For AIX, Red Hat Linux, or Solaris, run the following as root:
ndserver
 - For Windows, ndserver runs as a service that starts automatically.
2. Start the wizard function of Dispatcher, **ndwizard**.

The wizard guides you step by step through the process of creating a basic configuration for the Dispatcher component. You will be asked questions about your network. You will be guided through the setup of a cluster for Dispatcher to load balance traffic between a group of servers.

With the configuration wizard, you will see the following panels:

- Introduction to the wizard
- What is going to happen
- Preparing for the setup
- Choosing a host to configure (if necessary)
- Defining a cluster
- Adding a port
- Adding a server
- Starting an advisor
- Server machine setup

Configuring using the graphical user interface (GUI)

For an example of the GUI, see Figure 2 on page 6.

To start the graphical user interface, follow these steps:

1. Ensure **ndserver** is running:
 - For AIX, Red Hat Linux, or Solaris, run the following as root:
ndserver
 - For Windows, **ndserver** runs as a service that starts automatically.
2. Next, do one of the following:
 - Enter **ndadmin** (AIX, Red Hat Linux, or Solaris).
 - Click **Start**, click **Programs**, and click **IBM Network Dispatcher** (Windows).

General Instructions for using the GUI

For AIX, Solaris, and Windows, the left side of the panel displays a tree structure with IBM Network Dispatcher at the top level, and Dispatcher, ISS, and CBR as components.

Note: For Red Hat Linux, the tree structure displays the Dispatcher and CBR components. The system-specific metrics function provided by ISS is replaced by the Server Monitor Agent (SMA), which is unique for Red Hat Linux.

All of the components can be configured from the GUI. You can select elements in the tree structure by clicking mouse button one (normally the left button) and display pop-up menus by clicking mouse button two (normally the right button). The pop-up menus for the tree elements are also accessible from the menu bar located at the top of the panel.

Each item in the tree is marked with a + (plus) or a - (minus). Click on the plus to expand the items within it and the minus to compact the items.

The right side of the panel displays status indicator tabs for the element currently selected. The **Current Statistics** tab presents statistical information about the element. The **Refresh Statistics** button displays the latest statistical data. The **Configuration Settings** tab presents configuration parameters that can be set using the procedures outlined in the configuration chapters for each of the components. The **Update Configuration** button applies the latest changes to the configuration currently running. The **Lists** tab presents additional details about the selected tree element. The **Remove** button deletes highlighted items as selected in a list box. The **Lists** tab does not appear for all elements in the tree structure.

You can access **Help** by clicking the question mark icon in the upper right hand corner of the Network Dispatcher window.

- **Field Help** — describes each field, default values
- **How do I** — lists tasks that can be done from that screen
- **Contents** — a table of contents of all the Help information
- **Index** — an alphabetical index of the help topics

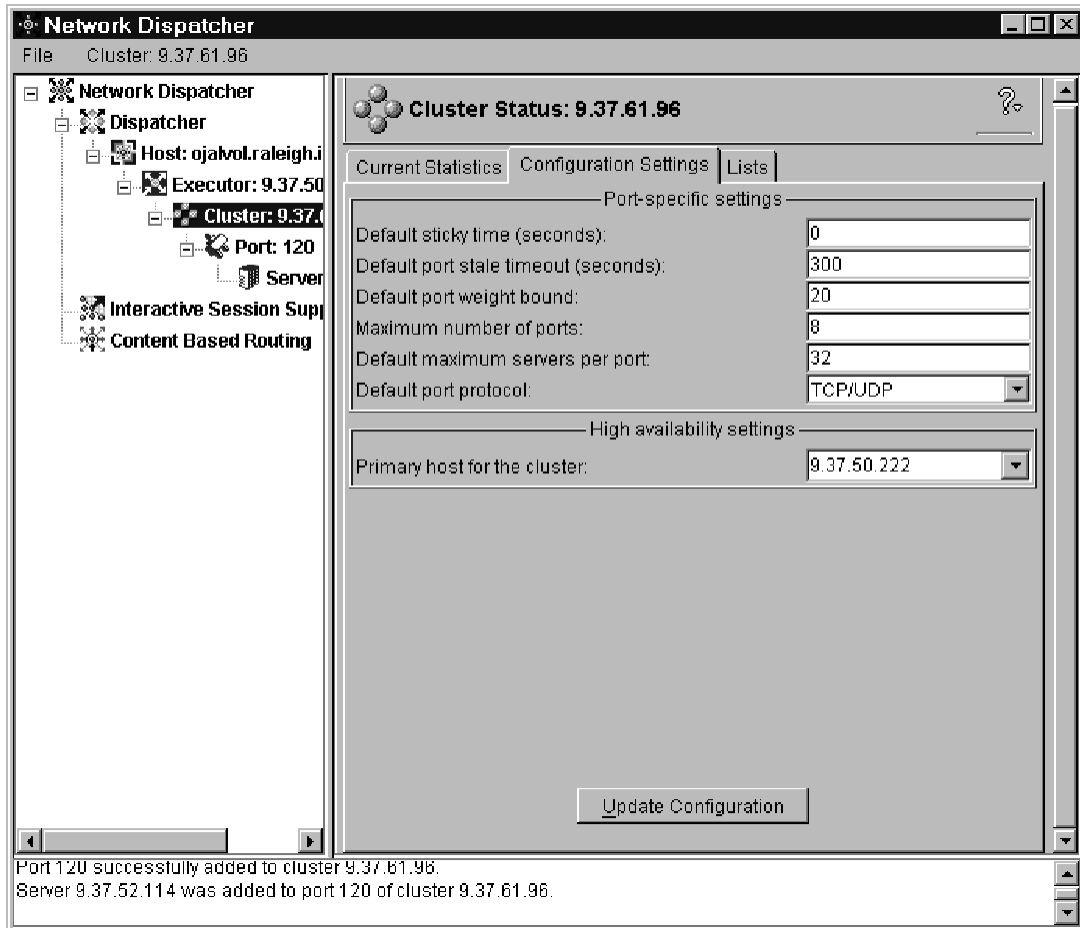


Figure 2. The graphical user interface (GUI)

Testing your configuration

Test to see if the configuration is working.

1. From a web browser, go to location **http://www.intersplash.com**. If a page appears, all is working.
2. Reload the page in the web browser.
3. Look at the results of the following command: **ndcontrol server report www.intersplash.com:80**. The total column of the two servers should add up to "2."

Chapter 2. Installing IBM Network Dispatcher

This chapter instructs you on the hardware requirements and installation of IBM Network Dispatcher on AIX, Red Hat Linux, Solaris, and Windows. Follow these instructions beginning at:

- “Requirements for AIX”
- “Requirements for Red Hat Linux” on page 14
- “Requirements for Solaris” on page 17
- “Requirements for Windows 2000 or Windows NT” on page 19

Note: If you are migrating from a previous version, note that the directory structure has changed and you should be sure to move any of your own configuration files and scripts (such as goIdle and goStandby) into the appropriate `/bin` directory of each component (ISS, CBR or Dispatcher).

Requirements for AIX

- Any IBM RS/6000-based machine
- IBM AIX Version 4.3.2 or higher
- 50 MB of available disk space for installation

Note: Additional disk space will be needed for logs.

- The following Network Interface Cards (NICs) are supported:
 - 16 Mb Token ring
 - 10 Mb Ethernet
 - 100 Mb Ethernet
 - Fiber distributed data interface (FDDI)
- IBM Java development kit (JDK) version 1.1.8 or higher (but not 1.2.x versions)
- IBM Web Traffic Express (WTE) version 2.0 (or higher) if you are using the CBR component for load balancing HTTP traffic.
- Netscape Navigator 4.07 (or higher) or Netscape Communicator 4.61 (or higher) for viewing online Help

Installing for AIX

Table 1 lists the installp images to install for Network Dispatcher for AIX.

Table 1. AIX installp images

Dispatcher (component, administration, license, and messages)	intnd.nd.driver intnd.nd.rte intnd.ndadmin.rte intnd.nd.license intnd.msg.nd.<language>.nd.rte intnd.msg.<language>.ndadmin.rte intnd.admin.rte intnd.msg.<language>.admin.rte
ISS (component, administration, license, and messages)	intnd.iss.rte intnd.issadmin.rte intnd.iss.license intnd.msg.<language>.iss.rte intnd.msg.<language>.issadmin.rte intnd.admin.rte intnd.msg.<language>.admin.rte
CBR (component, administration, license, and messages)	intnd.cbr.rte intnd.cbradmin.rte intnd.cbr.license intnd.msg.<language>.cbr.rte intnd.msg.<language>.cbradmin.rte intnd.admin.rte intnd.msg.<language>.admin.rte
User's Guide	intnd.doc.<language>

where <language> is one of:

- en_US
- de
- es_ES
- fr
- it
- ja_JP
- Ja_JP
- ko_KR
- pt_BR
- zh_CN
- ZH_TW
- zh_TW

If you are downloading an evaluation copy of the product from the Web site, use the installation instructions on (<http://www.software.ibm.com/network/dispatcher/downloads/>).

Before you install

When you install the product, you are given the option of installing any or all of the following:

- Dispatcher component (with messages)
- ISS component (with messages)
- CBR component (with messages)
- Dispatcher, ISS, and CBR components (with messages for all three)
- Dispatcher administration component
- CBR administration component
- ISS administration component
- User's Guide

It is necessary that you install the ISS component on every machine on which you plan to run either the ISS monitor function or the ISS agent function. For more about the monitor and agent functions, see "Chapter 9. Planning for the Interactive Session Support component" on page 125.

Installation steps

Note: If you have an earlier version installed, you should uninstall that copy before installing the current version. First, ensure that both the Dispatcher executor and server are stopped. Then, to uninstall the entire product, enter **installp -u intnd**. To uninstall specific filesets, list them specifically instead of specifying the package name.

Follow these steps to install IBM Network Dispatcher for AIX:

1. Log in as root.
2. Insert the product media, or if you are installing from the Web, copy the install images to a directory.
3. Install the installation image. It is recommended that you use SMIT to install IBM Network Dispatcher for AIX because SMIT will ensure that all messages are installed automatically.

Using **SMIT**:

Select Software Installation and Maintenance

Select Install and Update Software

Select Install Software Products at Latest Level

Select Install and update from all Available Software

Enter The device or directory containing the installp images

Enter On the *SOFTWARE to Install line, the appropriate information to specify options (or select PF4)

Press OK

When the command completes, press **Done**, and then select **Exit Smit** from the Exit menu or press **F12**. If using SMITTY, press **F10** to exit the program.

Using the Command Line:

If installing from a CD, you must enter the following commands to mount the CD:

```
mkdir /cdrom  
mount -v cdrfs -p -r /dev/cd0 /cdrom
```

Refer to the following table to determine which command(s) to enter to install the desired IBM Network Dispatcher for AIX components:

Table 2. AIX install commands

Dispatcher (with msgs)	installp -acXgd <i>device</i> intnd.nd.rte intnd.msg.<language>.nd.rte intnd.msg.<language>.ndadmin.rte intnd.msg.<language>.admin.rte
ISS (with msgs)	installp -acXgd <i>device</i> intnd.iss.rte intnd.msg.<language>.iss.rte intnd.msg.<language>.issadmin.rte intnd.msg.<language>.admin.rte
Content Based Routing (with msgs)	installp -acXgd <i>device</i> intnd.cbr.rte intnd.msg.<language>.cbr.rte intnd.msg.<language>.cbradmin.rte intnd.msg.<language>.admin.rte
Dispatcher, ISS, and Content Based Routing (with msgs)	installp -acXgd <i>device</i> intnd.nd.rte intnd.msg.<language>.nd.rte intnd.msg.<language>.ndadmin.rte intnd .msg.<language>.admin.rte intnd.iss.rte intnd.msg.<language>.iss.rte intnd.msg.<language>.issadmin.rte intnd.cbr.rte intnd.msg.<language>.cbr.rte intnd.msg.<language>.cbradmin.rte
Dispatcher, ISS, and CBR (with msgs and documentation)	installp -acXgd <i>device</i> intnd.nd.rte intnd.msg.<language>.nd.rte intnd.msg.<language>.ndadmin.rte intnd .msg.<language>.admin.rte intnd.iss.rte intnd.msg.<language>.iss.rte intnd.msg.<language>.issadmin.rte intnd.cbr.rte intnd.msg.<language>.cbr.rte intnd.msg.<language>.cbradmin.rte intnd.doc.<language>

Table 2. AIX install commands (continued)

Documentation only	<code>installp -acXgd device intnd.doc.<language></code>
--------------------	--

where *device* is:

- `/cdrom` if you are installing from a CD.
- `/dir` (the directory containing the installp images) if you are installing from a file system.

Ensure that the result column in the summary contains SUCCESS for each part of IBM Network Dispatcher that you are installing (APPLYing). Do not continue until all of the parts you wish to install are successfully applied.

Note that if you are using ISS, it must also be installed on each server machine.

Note: To generate a list of filesets in any installp image, including all available message catalogs, enter

```
installp -ld device
```

where *device* is:

- `/cdrom` if you are installing from a CD.
- `/dir` (the directory containing the installp images) if you are installing from a file system.

To unmount the CD, type:

```
umount /cdrom
```

4. Verify that the product is installed. Enter the following command:

```
lslpp -h | grep intnd
```

If you installed the full product, this command returns the following:

```
intnd.admin.rte
intnd.cbr.rte
intnd.cbradmin.rte
intnd.cbr.license
intnd.iss.rte
intnd.issadmin.rte
intnd.iss.license
intnd.msg.en_US.admin.rte
intnd.msg.en_US.cbr.rte
intnd.msg.en_US.cbradmin.rte
intnd.msg.en_US.iss.rte
intnd.msg.en_US.issadmin.rte
intnd.msg.en_US.nd.rte
```

```
intnd.msg.en_US.ndadmin.rte
intnd.nd.driver
intnd.nd.license
intnd.nd.rte
intnd.ndadmin.rte
intnd.doc.en_US
```

Requirements for Red Hat Linux

- Red Hat Linux version 6.1 (Linux kernel version 2.2.12-20)
- 50 MB of available disk space for installation

Note: Additional disk space will be needed for logs.

- The following Network Interface Cards (NICs) are supported:
 - 10 Mb Ethernet
 - 100 Mb Ethernet
- A version of the Korn Shell (ksh) must be installed
- IBM Java runtime environment (JRE) version 1.1.8 or higher (but not 1.2.x versions)
- The JAVA_HOME and PATH environment variables must be set using the **export** command. The content of JAVA_HOME variable is dependent on where the user has Java installed. Below is an example:
 - JAVA_HOME=/usr/local/jdk1.1.8
 - PATH=\$JAVA_HOME/bin:\$PATH
- IBM Web Traffic Express (WTE) version 3.0 (or higher) if you are using the CBR component for load balancing HTTP traffic
- Netscape Navigator 4.07 (or higher) or Netscape Communicator 4.61 (or higher) for viewing online Help

Installing for Red Hat Linux

This section explains how to install IBM Network Dispatcher on Red Hat Linux using the product CD or the downloaded evaluation copy of the product from the Web site. Installation instructions can be found on the Web site (<http://www.software.ibm.com/network/dispatcher/downloads/>).

Before you install

Before beginning the installation procedure, ensure that you have root authority to install the software.

Installation steps

Note: If you have an earlier version installed, you should uninstall that copy before installing the current version. First, ensure that both the Dispatcher executor and server are stopped. Then to uninstall the entire

product, enter **rpm -e pkgname**. When uninstalling, reverse the order used for package installation ensuring the administration packages are last to be uninstalled.

To install IBM Network Dispatcher:

1. Prepare to install.

- Log in as root.
- Insert the product media or download the product from the Web site and install the installation image using RPM (Red Hat Packaging Manager). The installation image is a file in the format **ndlinux-version.tar**.
- Untar the tar file in a temporary directory by entering the following command: **tar -xf ndlinux-version.tar**. The result will be a set of files with the .rpm extension.

The following is a list of the RPM installable packages.

- *ibmnd-adm-release-version.i386.rpm* (Base administration).
- *ibmnd-cbr-release-version.i386.rpm* (CBR).
- *ibmnd-cbradm-release-version.i386.rpm* (CBR administration).
- *ibmnd-cbrlic-release-version.i386.rpm* (CBR license).
- *ibmnd-doc-release-version.i386.rpm* (Documentation).
- *ibmnd-dsp-release-version.i386.rpm* (Dispatcher).
- *ibmnd-dspadm-release-version.i386.rpm* (Dispatcher administration).
- *ibmnd-dsplic-release-version.i386.rpm* (Dispatcher license).
- *ibmnd-sma-release-version.i386.rpm* (Server Monitor Agent).
- The order in which the packages are installed is important. Below is a list of packages needed for each component and the order in which they should be installed:
 - Administration
 - Base administration
 - CBR administration and/or Dispatcher administration
 - Content Based Routing
 - Administration (described above)
 - CBR license
 - CBR
 - Dispatcher
 - Administration (described above)
 - Dispatcher license
 - Dispatcher
 - Other

- Documentation (*User's Guide*)
- Server Monitor Agent (for servers only)

The command to install the packages should be issued from the same directory where the RPM files reside. Issue the following command to install each package: **rpm -i package.rpm**.

Note: At least one of the RPM files requires that Java be installed and registered in the RPM database. If Java is installed but not registered in the RPM database, use the install command with a 'no dependencies' option as follows:

rpm -i --nodeps package.rpm

- The installed components reside in the following directories:
 - Administration - **/opt/nd/admin**
 - Dispatcher - **/opt/nd/dispatcher**
 - CBR - **/opt/nd/cbr**
 - Server Monitor Agent (SMA) - **/opt/nd/sma**
 - Documentation (*User's Guide*) - **/opt/nd/documentation**
 - To uninstall the packages, reverse the order used for package installation ensuring the administration packages are last to be uninstalled.
2. Verify that the product is installed. Enter the following command:

rpm -qa | grep ibmnd

Installing the full product should produce a listing like the following:

- *ibmnd-adm-release-version*
- *ibmnd-cbradm-release-version*
- *ibmnd-cbrlic-release-version*
- *ibmnd-dspadm-release-version*
- *ibmnd-cbr-release-version*
- *ibmnd-dsplic-release-version*
- *ibmnd-dsp-release-version*
- *ibmnd-doc-release-version*
- *ibmnd-sma-release-version*

Requirements for Solaris

- Any SPARC workstation supported by Solaris Version 2.6 and Solaris Version 7 (32 bit mode)
- 50 MB of available disk space for installation

Note: Additional disk space will be needed for logs.

- The following Network Interface Cards (NICs) are supported:
 - 10 Mb Ethernet
 - 100 Mb Ethernet
- IBM Java runtime environment (JRE) version 1.1.8 or higher (but not 1.2.x versions)
- IBM Web Traffic Express (WTE) version 2.0 (or higher) if using the CBR component for load balancing HTTP traffic
- Sun Microsystems HotJava Browser 1.0.1 or higher for viewing online Help

Installing for Solaris

This section explains how to install IBM Network Dispatcher on Solaris using the product CD. If you are downloading an evaluation copy of the product from the internet, use the installation instructions on the Web site (<http://www.software.ibm.com/network/dispatcher/downloads/>).

Before you install

Before beginning the installation procedure, ensure that you have root authority to install the software.

Installation steps

Note: If you have an earlier version installed, you should uninstall that copy before installing the current version. First, ensure that you have stopped both the executor and the server. Then, to uninstall Network Dispatcher enter **pkgrm *pkgname***.

To install IBM Network Dispatcher:

1. Prepare to install.
 - Log in as root.
 - Insert the CD-ROM that contains the IBM Network Dispatcher software into the appropriate drive.

At the command prompt, enter **pkgadd -d *pathname***, where *-d pathname* is the device name of the CD-ROM drive or the directory on the hard drive where the package is located; for example: **pkgadd -d /cdrom/cdrom0/**.

You will be given a list of packages to install. They are:

- **ibmdsp** IBM ND Dispatcher for Solaris

- ibmdspadm IBM ND Dispatcher Administration for Solaris
- ibmdsplic IBM ND Dispatcher License for Solaris
- ibmiss IBM ND Interactive Session Support for Solaris
- ibmissadm IBM ND Interactive Session Support Administration for Solaris
- ibmisslic IBM ND Interactive Session Support License for Solaris
- ibmcbr IBM ND Content Based Routing for Solaris
- ibmcbradm IBM ND Content Based Routing Administration for Solaris
- ibmcbrlic IBM ND Content Based Routing License for Solaris
- ibmndadm IBM ND Base Administration for Solaris
- ibmnddoc IBM ND Documentation

If you would like to install all of the packages, simply type 'All' and press return. If you would like to install some of the components, enter the number(s) corresponding to the packages to be installed separated by a space or comma and press return. You may be prompted to change permissions on existing directories or files. Simply press return, or answer "yes." You need to install prerequisite packages (because it installs in alphabetical order not prerequisite order). If you say 'all' then just answer 'yes' to all prompting and the install will complete successfully.

All of the packages depend on the common ND package, ibmndadm. This common package must be installed along with any of the other packages.

If you want to install the entire Dispatcher component, you must install four pieces: ibmndadm, ibmdsp, ibmdspadm, and ibmdsplic. If you want to install the remote administration for the Dispatcher component, you only have to install three pieces: ibmndadm, ibmdsplic, and ibmdspadm. The same is true for the packages ISS and CBR.

The installed components reside in the following directories:

- Dispatcher - **/opt/nd/dispatcher**
 - ISS - **/opt/nd/iss**
 - CBR - **/opt/nd/cbr**
2. This *User's Guide* will be installed in the directory **/opt/nd/documentation**
 3. Verify that the product is installed. Enter the following command: **pkginfo | grep ibm.**

Installing the full product should produce a listing like the following:

- ibmdsp
- ibmdspadm
- ibmdsplic

- ibmiss
- ibmissadm
- ibmisslic
- ibmcbr
- ibmcbradm
- ibmcbrlic
- ibmndadm
- ibmnddoc

Requirements for Windows 2000 or Windows NT

- Any Intel x86 PC supported by Microsoft Windows 2000 or Microsoft Windows NT, version 4.0 with Service Pack 5 (or higher)
- Windows 2000 Professional, Server, or Advanced Server; or Windows NT Workstation or Server, version 4.0 with Service Pack 5 (or higher)
 - The following functions can run on Windows 2000 Professional, Server, or Advanced Server; or Windows NT Workstation or Server:
 - Dispatcher
 - Interactive Session Support
 - Content Based Routing
 - Windows server is required for machines running the NameServer Observer mode of ISS
- 50 MB of available disk space for installation

Note: Additional disk space will be needed for logs.

- The following Network Interface Cards (NICs) are supported:
 - 16 Mb Token ring
 - 10 Mb Ethernet
 - 100 Mb Ethernet
- IBM Java runtime environment (JRE) version 1.1.8 or higher (but not 1.2.x versions)
- IBM Web Traffic Express (WTE) version 2.0 (or higher) if using the CBR component for load balancing HTTP traffic
- Ensure your default browser is either Netscape Navigator 4.07 (or higher), Netscape Communicator 4.61 (or higher), or Internet Explorer 4.0 (or higher). The default browser is used for viewing online Help.

Installing for Windows 2000 or Windows NT

This section explains how to install IBM Network Dispatcher on Windows 2000 or Windows NT using the product CD. If you are downloading an evaluation copy of the product from the Web site, use the installation

instructions on the Web site

(<http://www.software.ibm.com/network/dispatcher/downloads/>).

Note: When downloading from the Web site, you can obtain the Windows 2000 install package and/or the Windows NT install package.

Installation Packages

You will be given a choice of packages to install.

They are:

- Dispatcher Runtime
- Dispatcher Administration
- Dispatcher License
- Interactive Session Support Runtime
- Interactive Session Support Administration
- Interactive Session Support License
- Content Based Routing Runtime
- Content Based Routing Administration
- Content Based Routing License
- User's Guide

Before you install

Windows 2000 version of IBM Network Dispatcher is supported on the following:

- Windows 2000 Professional
- Windows 2000 Server
- Windows 2000 Advanced Server

Windows NT version of IBM Network Dispatcher is supported on the following:

- Windows NT Workstation, version 4.0
- Windows NT Server, version 4.0

Note: The Windows 2000 and Windows NT versions of IBM Network Dispatcher will not run on Windows 98 or any other version of Windows.

Before beginning the installation procedure, ensure you have logged in as the Administrator or as a user with administrative privileges.

Installation steps

If you have an earlier version installed, you should uninstall that copy before installing the current version. To uninstall using the **Add/Remove Program**, do the following:

1. Click **Start**→**Settings**→**Control Panel**
2. Double-click on **Add/Remove Programs**
3. Select *IBM Network Dispatcher*
4. Click **Add/Remove** button

To install IBM Network Dispatcher:

1. Insert the IBM Network Dispatcher CD-ROM into your CD-ROM drive, and the install window should come up automatically.
2. The following step is only required if autorun of the CD did not work on your computer. Using your mouse, click mouse button 1 to perform these tasks:
 - Click **Start**.
 - Select **Run**.
 - Specify the CD-ROM disk drive, followed by setup.exe, for example:
`E:\setup`
3. Select **Language** in which to read the install process.
4. Click **OK**.
5. Follow the instructions of the setup program.
6. If you want to change the drive or directory destination, click **Browse**.
7. You have the choice of selecting "All of the ND product" or "your choice of components," and select what you want.
8. After installation is complete, a message will tell you to reboot your system before using IBM Network Dispatcher. This is necessary to make sure that all files are installed and the IBMNDPATH environment variable is added to the registry.

Chapter 3. Introducing IBM Network Dispatcher

This chapter gives an overview of IBM Network Dispatcher and includes the following sections:

- “What is IBM Network Dispatcher?”
- “Why do I need IBM Network Dispatcher?” on page 24
- “What’s new in this version?” on page 26
- “What are the components of IBM Network Dispatcher?” on page 28
- “How about high availability?” on page 30
- “Which component should I use: Dispatcher, ISS, CBR or all three?” on page 31

What is IBM Network Dispatcher?

The IBM Network Dispatcher is the next generation of server load balancing software. It boosts the performance of servers by directing TCP/IP session requests to different servers within a group of servers; in this way, it balances the requests among all the servers. This load balancing is transparent to users and other applications. IBM Network Dispatcher is useful for applications such as e-mail servers, World Wide Web servers, distributed parallel database queries, and other TCP/IP applications.

When used with Web servers, IBM Network Dispatcher can help maximize the potential of your site by providing a powerful, flexible, and scalable solution to peak-demand problems. If visitors to your site can’t get through at times of greatest demand, IBM Network Dispatcher can automatically find the optimal server to handle incoming requests, thus enhancing your customers’ satisfaction and your profitability.

IBM Network Dispatcher consists of three components that can be used separately or together to provide superior load-balancing results:

- You can use the Dispatcher component by itself to balance the load on servers within a local area network or wide area network using a number of weights and measurements that are dynamically set by Dispatcher. This component provides load balancing at a level of specific services, such as HTTP, FTP, SSL, NNTP, POP3, SMTP, and Telnet. It does not use a domain name server to map domain names to IP addresses.
- You can use the Interactive Session Support (ISS) component by itself to balance the load on servers within a local or wide area network using a domain nameserver (DNS) round-robin approach or a more advanced

user-specified approach. Load balancing is performed at the machine level. ISS can also be used to provide server load information to a Dispatcher machine.

When used for load-balancing, ISS works in conjunction with the DNS name server to map DNS names of ISS services to IP addresses. When used to provide server load information, a name server is not required.

Note: For Red Hat Linux, ISS is not supported. Instead, the Server Monitor Agent (SMA) provides server load information to the Dispatcher in the form of system-specific metrics.

- For HTTP protocol, you can use the Content Based Routing (CBR) component to load balance based on the content of the client request. A client sends a request to Web Traffic Express (WTE), and WTE sends the request to the appropriate server. The chosen server is the result of matching the URL to a specified rule.

For IMAP or POP3 protocols, CBR (without WTE) is a proxy which chooses an appropriate server based on userid and password provided by the client.

For more information, see “Chapter 7. Configuring the Content Based Routing component” on page 73.

- Using either the wide area function of the Dispatcher or a combination of the Dispatcher and ISS allows you to balance the load on servers within both local and remote networks.

For more information on the Dispatcher, ISS and CBR components, see “What are the components of IBM Network Dispatcher?” on page 28.

Why do I need IBM Network Dispatcher?

The number of users and networks connected to the global Internet is growing exponentially. This growth is causing problems of scale that can limit users’ access to popular sites.

Currently, network administrators are using numerous methods to try to maximize access. Some of these methods allow users to choose a different server at random if an earlier choice is slow or not responding. This approach is cumbersome, annoying, and inefficient. Another method is standard round-robin, in which the domain name server selects servers in turn to handle requests. This approach is better, but still inefficient because it blindly forwards traffic without any consideration of the server workload. In addition, even if a server fails, requests continue to be sent to it. For more information on approaches to managing server workload, including animated examples, go to the following URL:

<http://www.software.ibm.com/network/dispatcher/about/features/examples.html>

The need for a more powerful solution has resulted in IBM Network Dispatcher. It offers numerous benefits over earlier and competing solutions:

Scalability

As the number of client requests increases, you can add servers dynamically, providing support for tens of millions of requests per day, on tens or even hundreds of servers.

Efficient use of equipment

Load balancing ensures that each group of servers makes optimum use of its hardware by minimizing the hot-spots that frequently occur with a standard round-robin method.

Easy integration

IBM Network Dispatcher uses standard TCP/IP protocols. You can add it to your existing network without making any physical changes to the network. It is simple to install and configure.

Low overhead

Dispatcher needs only to look at the inbound client-to-server flows. It does not need to see the outbound server-to-client flows. This significantly reduces its impact on the application compared with other approaches and can result in improved network performance.

Non-invasive technology

Dispatcher does not modify any packets, nor does it require any modifications to the operating system on which it runs.

High availability

Dispatcher offers built-in high availability, utilizing a backup machine that remains ready at all times to take over load balancing should the primary Dispatcher machine fail. Dispatcher also offers mutual high availability which allows two machines to be both active and standby for each other. See "How about high availability?" on page 30.

ISS is intrinsically highly available. All the nodes in an ISS configuration work together to eliminate any single point of failure. For more information, see "Configure high availability" on page 89.

Content Based Routing (CBR)

For HTTP, CBR gives a WTE administrator the ability to proxy requests to specific servers based on the content requested. For example, if a request contains the string `'/cgi-bin/'` in the directory portion of the URL, and the server name is a local server, CBR can direct the request to the best server in a set of servers specifically allocated to handle cgi requests.

For IMAP or POP3, CBR (without WTE) is a proxy which chooses an appropriate server based on userid and password provided by the client.

What's new in this version?

Version 3.0 of IBM Network Dispatcher contains a number of new features. The most significant new features are listed here.

- **Support for the Windows 2000 operating system**

This feature applies to the Dispatcher, CBR, and ISS components.

Version 3.0 introduces support for installing and running IBM Network Dispatcher on a new platform — Windows 2000. Support is provided for both the Windows 2000 Professional, Windows 2000 Server, and Windows 2000 Advanced Server versions of this platform. See “Requirements for Windows 2000 or Windows NT” on page 19, for more information.

- **Content Based Routing (CBR) configuration wizard**

This feature applies to the CBR component.

The CBR wizard allows you to create a CBR configuration quickly. When you have completed the wizard, you will be able to load balance HTTP or IMAP and POP3 traffic. See “Configuration wizard” on page 75, for more information.

- **IMAP/POP3 CBR proxy support**

This feature applies to the CBR component.

The CBR component is now able to support proxying IMAP and POP3 protocols. See “Overview of Content Based Routing (CBR)” on page 29 and “CBR proxy (for IMAP or POP3)” on page 70, for more information.

- **Mutual high availability**

This feature applies to the Dispatcher component.

Mutual high availability is a feature that allows both Dispatcher machines to actively load balance a portion of the client traffic, while also backing up each other. See “Mutual high availability” on page 49, for more information.

- **Cross port affinity**

This feature applies to the Dispatcher component.

Cross port affinity is a sticky feature enhancement that allows different ports to direct client requests to the same server. See “Cross port affinity” on page 118, for more information.

- **Affinity address mask**

This feature applies to the Dispatcher component.

Affinity address mask is a sticky feature enhancement to group clients based upon common subnet address. A mechanism allows you to mask the common high-order bits of the IP address. When this feature is enabled, for

all subsequent connections coming in from the same subnet address, the same server will be selected. See “Affinity address mask” on page 119, for more information.

- **Rule affinity override**

This feature applies to the Dispatcher and CBR components.

With rule affinity override, you can override the stickiness of a port for a specific server. For example, you are using a rule to limit the amount of connections to each application server, and you have an overflow server with an always true rule that says “please try again later” for that application. However, you don’t want the client to be sticky to that server, so you can now change the overflow server to override that affinity normally associated with that port. The next time the client requests the cluster, it is load balanced to the best available application server, not the overflow server. See “Rule affinity override” on page 119, for more information.

- **Type of service (TOS) rule**

This feature applies to the Dispatcher component.

TOS rule allows you to load balance across servers based on the content of the “type of service” field in the IP header. See “Using rules based on type of service (TOS)” on page 111, for more information.

- **Manager fixed weights**

This feature applies to the Dispatcher and CBR components.

Manager fixed weight provides you the option to run the advisors, without the function of having the manager update the server weights. See “Manager fixed weights” on page 86, for more information.

- **Multiple address collocation**

This feature applies to the Dispatcher component for all platforms with the exception of Windows.

Multiple address collocation allows you to set up the address of the collocated server to an address other than the NFA. Collocation refers to the Network Dispatcher residing on the same server that it is load balancing. See “Using Collocated Servers” on page 50, for more information on collocated servers. See page 59, for more information on multiple address collocation.

- **WebSphere Application Server (WAS) advisor**

This feature applies to the Dispatcher and CBR components.

A new sample advisor specifically for monitoring the state of WAS servers being load balanced is provided. See “WebSphere Application Server (WAS) advisor” on page 103, for more information.

- **IMAP advisor**

This feature applies to the Dispatcher and CBR components.

This is a new advisor for monitoring IMAP protocol. See “List of advisors” on page 100, for more information on the IMAP advisor.

What are the components of IBM Network Dispatcher?

The three components of IBM Network Dispatcher are Dispatcher, Interactive Session Support (ISS), and Content Based Routing (CBR). IBM Network Dispatcher gives you the flexibility of using these components separately or together depending on your site configuration. This section gives an overview of the Dispatcher, ISS, and CBR.

Note: For Red Hat Linux, ISS is replaced by the Server Monitor Agent (SMA). Similar to ISS, SMA provides a system-specific metrics function, reporting on the health of the servers. SMA is unique to IBM Network Dispatcher for Red Hat Linux and can not be used on any other platform. For information on SMA see “Server Monitor Agent (SMA)” on page 106.

For configuration examples, go to “Which component should I use: Dispatcher, ISS, CBR or all three?” on page 31.

Overview of Dispatcher

The Dispatcher component does not use a domain name server for load balancing. It balances traffic among your servers through a unique combination of load balancing and management software. The Dispatcher can also detect a failed server and forward traffic around it.

All client requests sent to the Dispatcher machine are directed to the most optimal server according to certain dynamically set weights. You can use the default values for those weights or change the values during the configuration process.

The server sends a response back to the client without any involvement of the Dispatcher. No additional code is required on your servers to communicate with the Dispatcher.

The Dispatcher component is the key to stable, efficient management of a large, scalable network of servers. With the Dispatcher, you can link many individual servers into what appears to be a single, virtual server. Your site thus appears as a single IP address to the world. Dispatcher functions independently of a domain name server; all requests are sent to the IP address of the Dispatcher machine.

The Dispatcher brings distinct advantages in balancing traffic load to clustered servers, resulting in stable and efficient management of your site.

Overview of Interactive Session Support (ISS)

ISS periodically monitors the level of activity on a group of servers and detects which server is the least heavily loaded. It can also detect a failed server and forward traffic around it. Once every monitoring period, ISS ensures that the information used by the domain name server or the Dispatcher accurately reflects the load on the servers. The load is a measure of how hard the server is working. The system ISS administrator controls both the type of measurement used to measure the load and the length of the load monitoring period. You can configure ISS to suit your environment, taking into account such factors as frequency of access, the total number of users, and types of access (for example, short queries, long-running queries, or CPU-intensive loads).

In addition to the functions already mentioned, ISS can perform load balancing and can feed information to Dispatcher as it is performing load balancing.

You can use the ISS component with or without a DNS name server:

- If you are using ISS for load balancing, a domain name server is required. This may be either an actual DNS name server or—if you set up a small, separate subdomain for a new nameserver—a replacement name server provided by ISS. Using this approach, ISS runs on a name server machine. A client submits a request for resolution of the DNS name of an ISS service, which has been set up by an administrator. ISS then resolves the name to the IP address of a server in the cell, and forwards this IP address to the client.
- If you are using ISS just to collect server load information, a domain name server is not needed. The ISS monitor collects server load information from the ISS agents running on the individual servers and forwards it to the Dispatcher. The Dispatcher uses this load information, along with other sources of information, to perform load balancing.

Overview of Content Based Routing (CBR)

CBR can be configured with WTE for HTTP servers. Or, it can be configured as a CBR proxy (without WTE) for IMAP and POP3 servers. However, CBR cannot be configured for both in the same Network Dispatcher machine.

CBR with WTE (for processing HTTP traffic):

CBR works with IBM Web Traffic Express (WTE) to proxy client requests to specified servers. WTE is a caching proxy server. It allows you to manipulate caching details for faster Web document retrieval with low network bandwidth requirements. CBR along with WTE filters Web page content using specified rule types.

CBR gives you the ability to specify a set of servers that should handle a request based on regular expression matching of the content of the request. Because CBR allows you to specify multiple servers for each type of request, the requests can be load balanced for optimal client response. CBR will also detect when one server in a set has failed, and stop routing requests to that server. The load balancing algorithm used by the CBR component is identical to the proven algorithm used by the Dispatcher component.

When a request is received by the WTE proxy, it is checked against the rules that have been defined in the CBR component. If a match is found, then one of the servers associated with that rule is chosen to handle the request. WTE then performs its normal processing to proxy the request to the chosen server.

CBR has the same functions as Dispatcher with the exception of high availability, subagent, wide area, and a few other configuration commands. For more information, see “Chapter 7. Configuring the Content Based Routing component” on page 73.

WTE must be running before any CBR configuration can be performed.

CBR proxy (for processing IMAP or POP3 traffic):

CBR (without WTE) can provide a single point of presence for many IMAP or POP3 servers. Each server can have a subset of all user mailboxes serviced by the point of presence. For IMAP and POP3, CBR is a proxy that chooses an appropriate server based on user ID and password provided by the client. The CBR proxy does not support rules-based load balancing.

How about high availability?

Dispatcher

The Dispatcher component offers a built-in high availability feature. This feature involves the use of a second Dispatcher machine that monitors the main, or primary, machine and stands by to take over the task of load balancing should the primary machine fail at any time. The Dispatcher component also offers mutual high availability which allows two machines to be both active and standby for each other. See “Configure high availability” on page 89.

ISS

ISS is intrinsically highly available. All the nodes in an ISS configuration work together to eliminate any single point of failure. Should the monitor machine fail, the survivors elect a new monitor to take over automatically. When using ISS to perform the NameServer replace function, high availability can be enhanced by nominating more than one server to act as a nameserver.

Which component should I use: Dispatcher, ISS, CBR or all three?

Use the questions in Table 3 to help you determine how to use the three components of IBM Network Dispatcher.

Table 3. Guidelines for using the components of IBM Network Dispatcher.

Question	Related information
Do you plan to manage servers on a local network and want to evaluate the Dispatcher component?	If yes, go to “Chapter 1. Getting started...quickly!” on page 1.
Are you managing servers on different subnets?	<p>If yes, you can use the wide area support provided by the Dispatcher. See “Configure wide area Dispatcher support” on page 94.</p> <p>Alternatively, you could use both ISS and Dispatcher, in particular if you make heavy use of Telnet or TN3270. For an example, go to “Managing local and remote servers with ISS and Dispatcher” on page 40.</p> <p>To manage servers on a local area network, you can use either Dispatcher or ISS to perform the load-balancing function. Note that ISS requires a domain name server.</p> <ul style="list-style-type: none">• For a quick start example using Dispatcher, go to “Chapter 1. Getting started...quickly!” on page 1.• For an example using ISS, go to “Managing local servers with ISS” on page 34.
Is high availability important?	Dispatcher has a High Availability feature. See “High availability” on page 48. ISS is intrinsically highly available.
What type of applications are you running?	Dispatcher supports HTTP, FTP, SSL, SMTP, NNTP, IMAP, POP3, Telnet, and any other TCP- or stateless UDP-based application. In addition, it permits you to write your own advisors (see “Create custom (customizable) advisors” on page 102). For any other IP-based application in a domain name server environment, use ISS.
Is application server feedback important?	If yes, use the Dispatcher advisor functions to collect information from HTTP, FTP, SSL, SMTP, NNTP, IMAP, POP3, or Telnet application servers. You can also write your own advisor (see “Create custom (customizable) advisors” on page 102.)
Do you expect a high volume of requests?	If yes, and you are running TCP or stateless UDP applications, use Dispatcher.

Table 3. Guidelines for using the components of IBM Network Dispatcher (continued).

Question	Related information
Is load feedback about the machines on which the servers are running important?	If yes, use Dispatcher for load balancing and run ISS or SMA (for Red Hat Linux) on the server machines to provide load information. For an example w/ISS, go to “Managing local servers with ISS and Dispatcher” on page 37. For an example w/SMA, go to “Managing local servers with Server Monitor Agent (SMA) and Dispatcher” on page 38. To write your own advisors, see “Create custom (customizable) advisors” on page 102.
Is Server Directed Affinity or state preservation important?	If yes, use Dispatcher. You can set up a port that sends the initial and all subsequent requests from a client to the same server by using the ndcontrol port set stickytime command option when configuring the port. For more information on this command, see “ndcontrol port — configure ports” on page 219.
Do you need to send HTTP requests to different servers, depending on the content of the request?	If yes, use the CBR rules. For more information about rules, see “Chapter 7. Configuring the Content Based Routing component” on page 73.
Do you need to send IMAP or POP3 requests to servers based on userid and password?	If yes, use the CBR proxy. For more information about the CBR proxy, see “Chapter 7. Configuring the Content Based Routing component” on page 73.

Managing local servers with Dispatcher

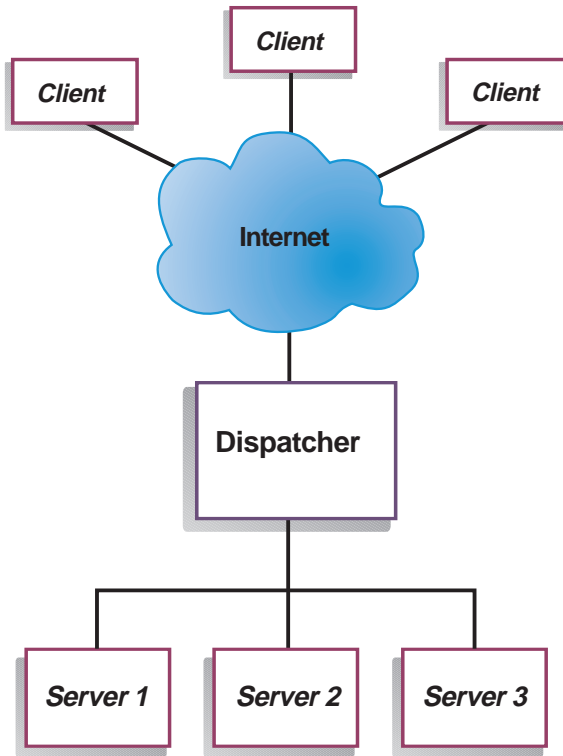


Figure 3. Example of a logical representation of a site using the Dispatcher to manage local servers

Figure 3 shows a logical representation of a site in which all servers are on a local network. The Dispatcher component is used to forward client requests to the servers.

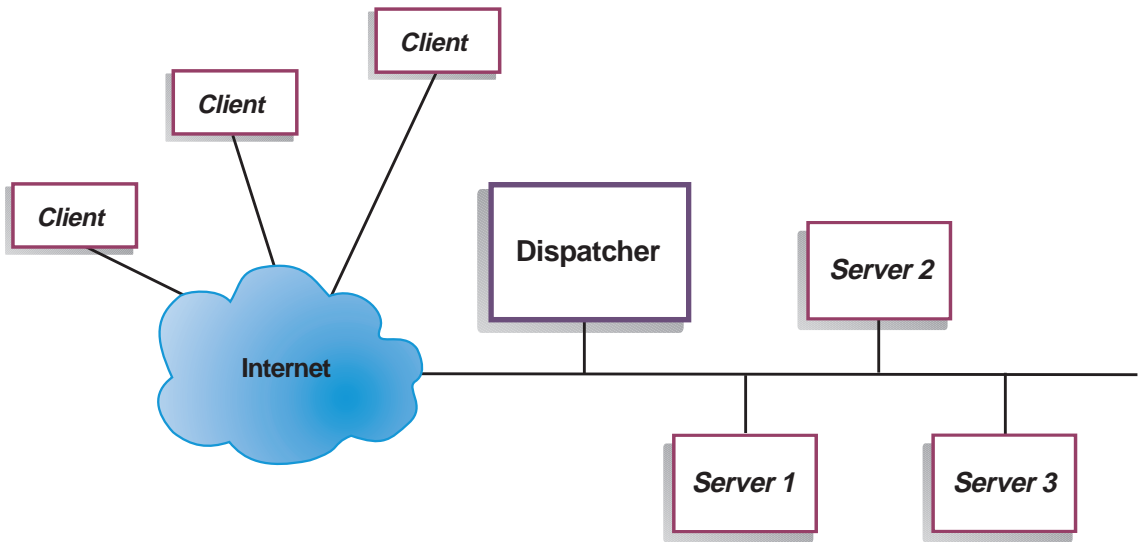


Figure 4. Example of a physical representation of a site using the Dispatcher to manage local servers

Figure 4 shows a physical representation of the site using an Ethernet network configuration. The Dispatcher machine can be installed without making any physical changes to the network. After a client request is directed to the optimal server by the Dispatcher, the response is then sent directly from server to client with no involvement by the Dispatcher.

Managing local servers with ISS

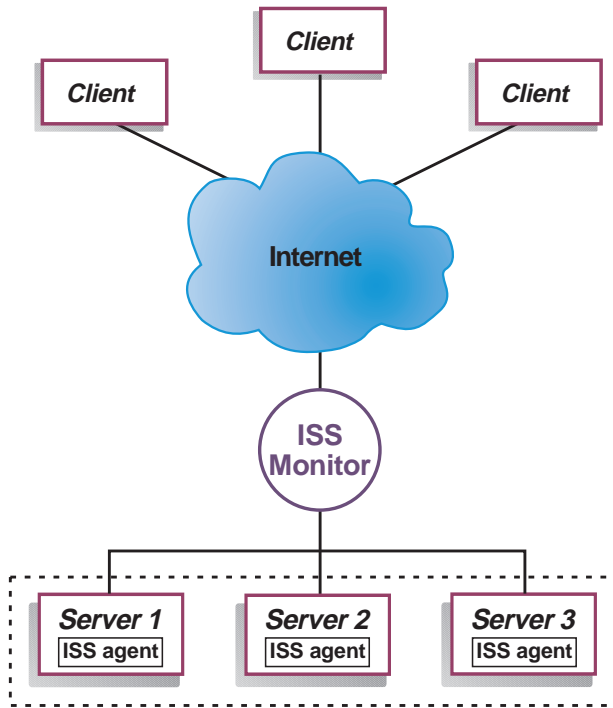


Figure 5. Example of a site using ISS to manage local servers

Figure 5 illustrates a site in which all servers are on a local network. The ISS daemon is installed on all servers. The ISS daemon running as a monitor feeds either the DNS NameServer (ISS in DNS-update mode) or a server which is running the ISS NameServing function (ISS in DNS-replace mode) updated information. When acting as an agent, the ISS daemon provides the monitor with server load information.

Managing local servers with CBR

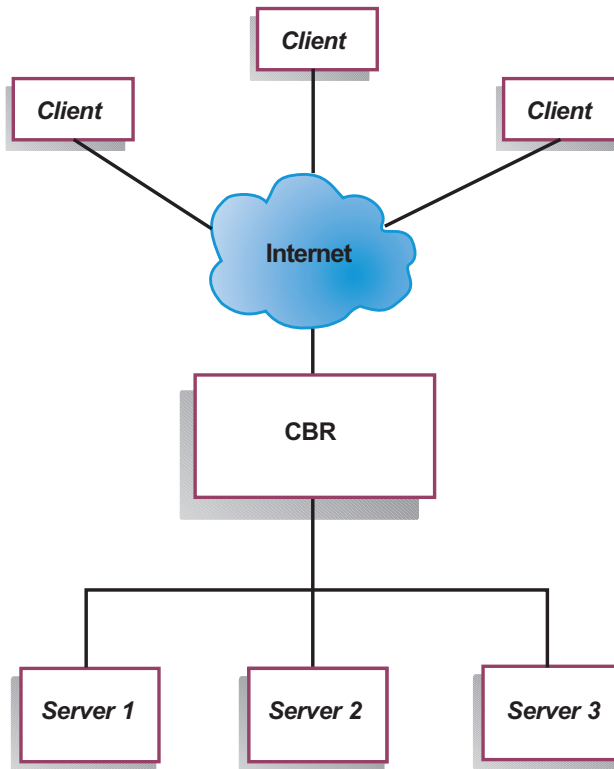


Figure 6. Example of a site using CBR to manage local servers

Figure 6 shows a logical representation of a site in which CBR is being used to proxy some content from local servers. For HTTP requests, the CBR component uses WTE to forward client requests to the servers based on the content of the URL. For IMAP or POP3, the CBR component (without WTE) proxies the client requests to the appropriate server based on userid and password.

Managing local servers with ISS and Dispatcher

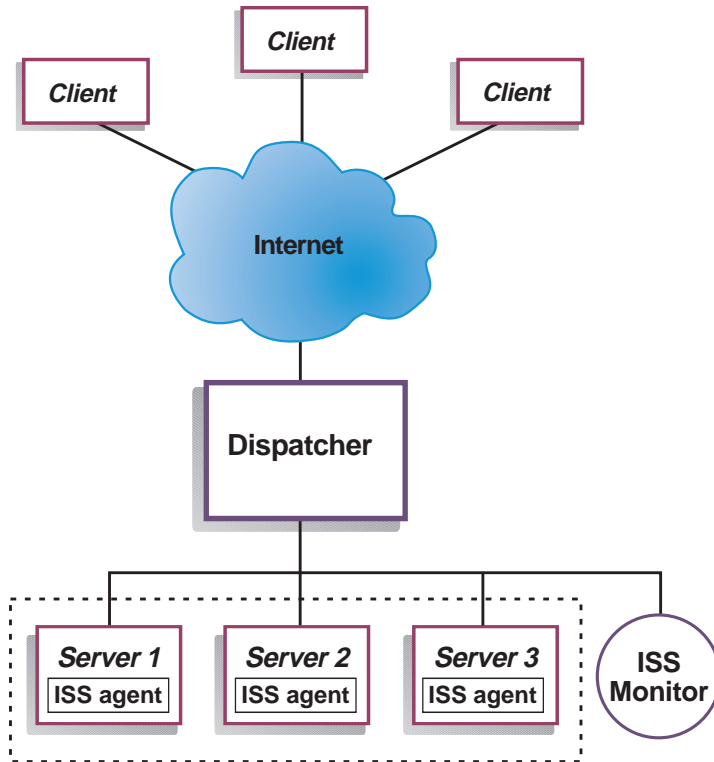


Figure 7. Example of a site using the Dispatcher and ISS to manage local servers

Figure 7 illustrates a site in which all servers are on a local network. The Dispatcher component is used to forward requests and the ISS component is used to provide system load information to the Dispatcher machine. A domain name server is not needed.

In this example, the ISS daemon is installed on each server. The administrator can configure which machine is used as the monitor. There is a priority ordering so that a lower-priority machine will take over as monitor only if the primary monitor has failed.

Managing local servers with Server Monitor Agent (SMA) and Dispatcher

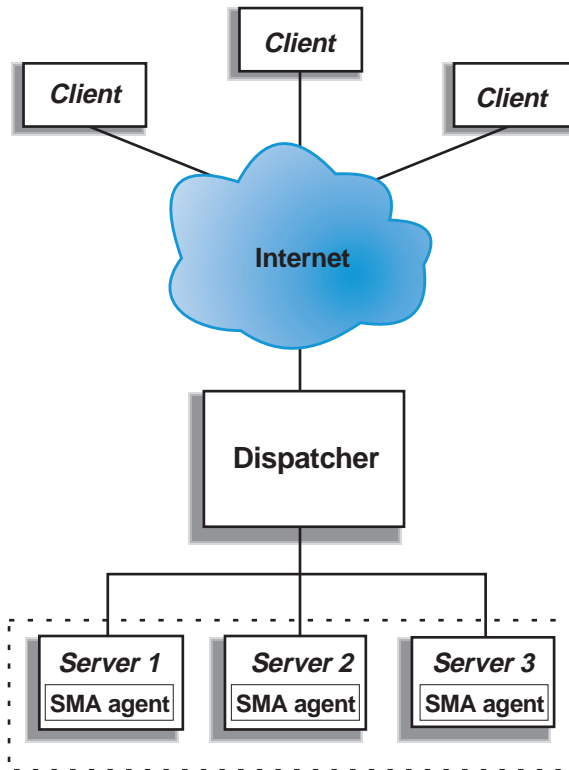


Figure 8. Example of a site using the Dispatcher and SMA to manage local servers

Figure 8 illustrates a site in which all servers are on a local network. The Dispatcher component is used to forward requests and the Server Monitor Agent (SMA) is used to provide system load information to the Dispatcher machine.

Note: In this example, the SMA agent is installed on each server. SMA can only be installed on servers running on a Linux platform. For Network Dispatcher machines running on a Red Hat Linux platform, SMA replaces the ISS system monitoring function for load balancing based on system-specific metrics.

Managing local and remote servers with Dispatcher

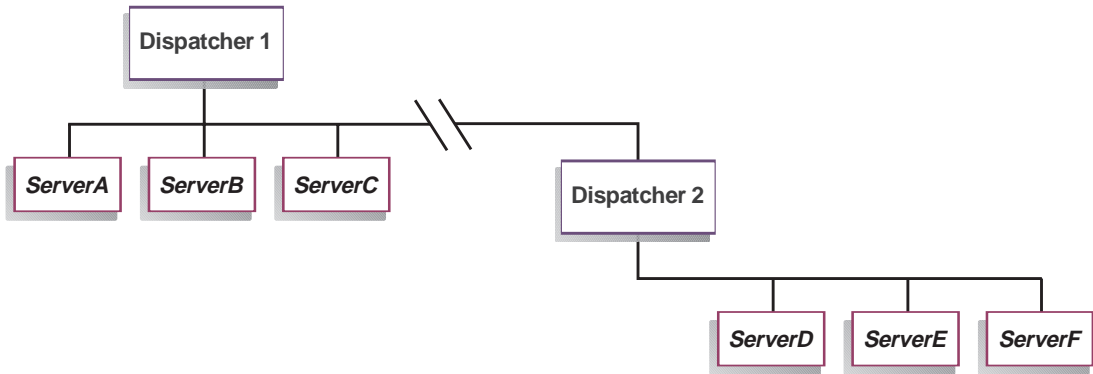


Figure 9. Example of a site using Dispatcher to manage local and remote servers

Wide area support in Dispatcher enables you to use both local and remote servers. Figure 9 shows a configuration where one "local" Dispatcher serves as the entry point for all requests. It distributes these requests among its own local servers (*a*, *b*, and *c*) and to the remote Dispatcher, which will load balance to its local servers (*d*, *e*, and *f*).

Managing local and remote servers with ISS and Dispatcher

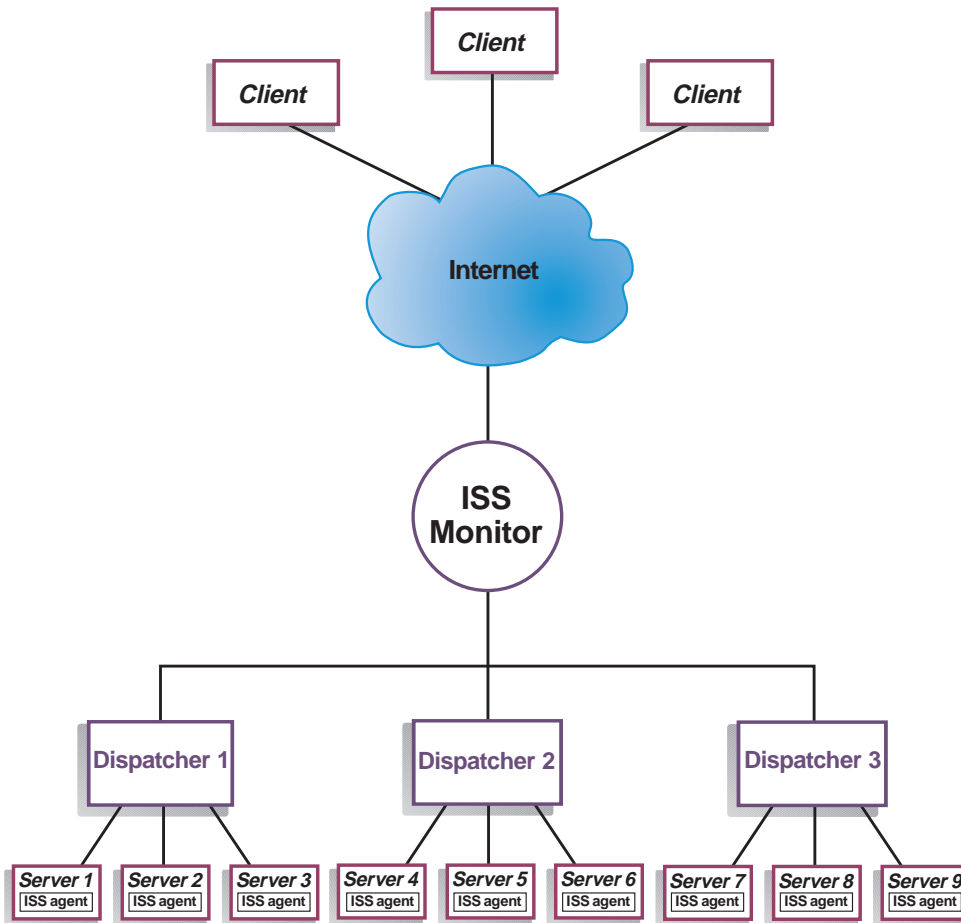


Figure 10. Example of a site using ISS and Dispatcher to manage local and remote servers

Figure 10 illustrates a site in which servers are located on local and remote networks. The ISS component is used, in conjunction with a domain name server, to forward requests to the best Dispatcher machine; the Dispatcher component is used to forward requests to the optimal server.

Optionally, the ISS daemon can be installed on all the servers to provide system load information to the Dispatcher machines. The ISS monitor function must be used to send the ISS agent information to the Dispatcher. The ISS monitor can be a Dispatcher machine or on a different machine.

Chapter 4. Planning for the Dispatcher component

This chapter describes what the network planner should consider before installing and configuring the Dispatcher component. If you want to plan for the Interactive Session Support (ISS) component, see “Chapter 9. Planning for the Interactive Session Support component” on page 125. If you want to plan for the Content Based Routing (CBR) component, see “Chapter 6. Planning for the Content Based Routing component” on page 67. This chapter includes the following sections:

- “Hardware and software requirements”
- “Planning considerations” on page 43
- “High availability” on page 48
- “Using Collocated Servers” on page 50
- “Load Balancing Dispatcher with ISS” on page 51

Hardware and software requirements

Requirements for AIX

- Any IBM RS/6000-based machine
- IBM AIX version 4.3.2 or higher
- 50 MB of available disk space for installation

Note: Additional disk space will be needed for logs.

- The following Network Interface Cards (NICs) are supported:
 - 16 Mb Token ring
 - 10 Mb Ethernet
 - 100 Mb Ethernet
 - Fiber distributed data interface (FDDI)
- IBM Java development kit (JDK) version 1.1.8 or higher (but not 1.2.x versions)
- Netscape Navigator 4.07 (or higher) or Netscape Communicator 4.61 (or higher) for viewing online Help

Requirements for Red Hat Linux

- Red Hat Linux version 6.1 (Linux kernel version 2.2.12-20)
- 50 MB of available disk space for installation

Note: Additional disk space will be needed for logs.

- The following Network Interface Cards (NICs) are supported:
 - 10 Mb Ethernet
 - 100 Mb Ethernet
- A version of the Korn Shell (ksh) must be installed
- IBM Java runtime environment (JRE) version 1.1.8 or higher (but not 1.2.x versions)
- The JAVA_HOME and PATH environment variables must be set using the **export** command. The content of JAVA_HOME variable is dependent on where the user has Java installed. Below is an example:
 - JAVA_HOME=/usr/local/jdk1.1.8
 - PATH=\$JAVA_HOME/bin:\$PATH
- Netscape Navigator 4.07 (or higher) or Netscape Communicator 4.61 (or higher) for viewing online Help

Requirements for Solaris

- Any SPARC workstation supported by Solaris version 2.6 and Solaris version 7 (32 bit mode)
- 50 MB of available disk space for installation

Note: Additional disk space will be needed for logs.

- The following Network Interface Cards (NICs) are supported:
 - 10 Mb Ethernet
 - 100 Mb Ethernet
- IBM Java runtime environment (JRE) version 1.1.8 or higher (but not 1.2.x versions)
- Sun Microsystems HotJava Browser 1.0.1 or higher for viewing online Help

Requirements for Windows 2000 and Windows NT

- Any Intel x86 PC supported by Microsoft Windows 2000; or Microsoft Windows NT, version 4.0 with Service Pack 5 (or higher)
- 50 MB of available disk space for installation

Note: Additional disk space will be needed for logs.

- The following Network Interface Cards (NICs) are supported:
 - 16 Mb Token ring
 - 10 Mb Ethernet
 - 100 Mb Ethernet
- Windows 2000 Professional, Server, or Advanced Server; or Windows NT Workstation or Server, version 4.0
- IBM Java runtime environment (JRE) version 1.1.8 or higher (but not 1.2.x versions)

- Ensure the default browser is either Netscape Navigator 4.07 (or higher), Netscape Communicator 4.61 (or higher), or Internet Explorer 4.0 (or higher). The default browser is used for viewing online Help.

Planning considerations

Dispatcher consists of the following functions:

- **ndserver** handles requests from the command line to the executor, manager, and advisors.
- The **executor** supports port-based load balancing of TCP and UDP connections. It is able to forward connections to servers based on the type of request received (for example, HTTP, FTP, SSL, etc.). The executor always runs when the Dispatcher component is being used.

The attributes of any object in the executor can be configured using **ndcontrol**. Setting the value of a default attribute causes succeeding objects created to have that attribute value, but does not affect the values of existing objects.

- The **manager** sets weights used by the executor based on:
 - Internal counters in the executor
 - Feedback from the servers provided by the advisors
 - Feedback from a system-monitoring program, such as ISS, SMA (for Red Hat Linux only), or WLM.

Using the manager is optional. However, if the manager is not used, load balancing will be performed using weighted round-robin scheduling based on the current server weights, and advisors will not be available.

- The **advisors** query the servers and analyze results by protocol before calling the manager to set weights as appropriate. Currently there are advisors available for HTTP, FTP, SSL, SMTP, NNTP, IMAP, POP3, Telnet, and WTE. You also have the option of writing your own advisors (see “Create custom (customizable) advisors” on page 102). Using the advisors is optional but recommended.
- Both a command line and a graphical user interface are provided to configure and manage the executor, advisors, and manager.
- The **SNMP subagent** allows an SNMP-based management application to monitor the status of the Dispatcher.
- A **sample configuration file** provides a script to use for configuration and administration of the Dispatcher machine. See “Appendix E. Sample configuration files” on page 251. After you have installed the product, this file can be found in the **dispatcher/samples/** subdirectory where IBM Network Dispatcher is located.

The three key functions of Dispatcher (executor, manager, and advisors) interact to balance and dispatch the incoming requests between servers. Along

with load balancing requests, the executor monitors the number of new connections, active connections, and connections in a finished state; does garbage collection of completed or reset connections and supplies this information to the manager.

Note: For AIX, Red Hat Linux, and Solaris, Dispatcher may be set up on a server machine. If you are using Windows, Dispatcher may **not** be set up on a server machine.

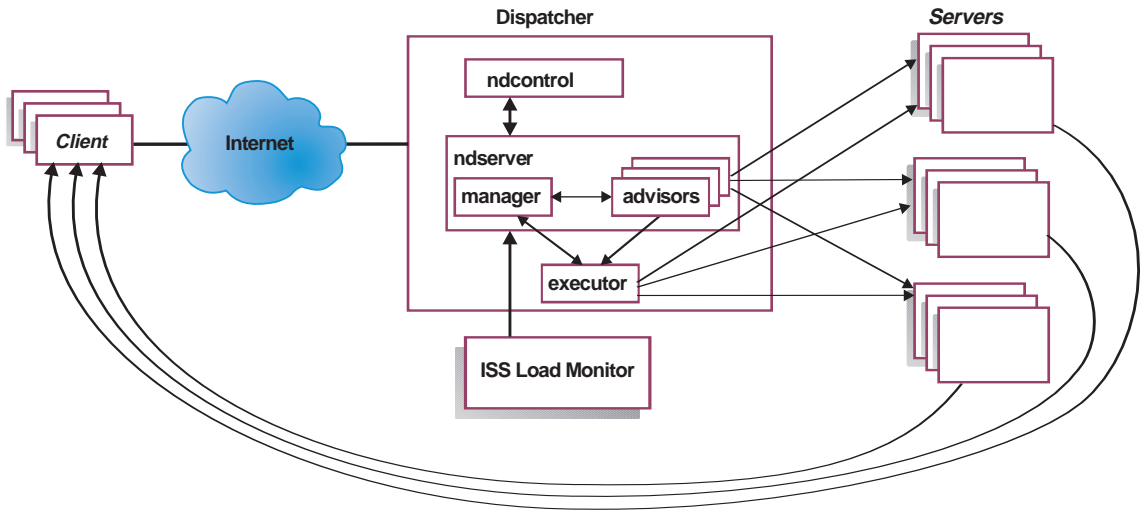


Figure 11. Dispatcher product components

The manager collects information from the executor, the advisors, and a system-monitoring program, such as ISS or SMA (unique to the Red Hat Linux platform). Based on the information the manager receives, it adjusts how the server machines are weighted on each port and gives the executor the new weighting for use in its balancing of new connections.

The advisors monitor each server on the assigned port to determine the server's response time and availability and then give this information to the manager. The advisors also monitor whether a server is up or down. Without the manager and the advisors, the executor does round-robin scheduling based on the current server weights.

There are many ways that you can configure Dispatcher to support your site. If you have only one host name for your site to which all of your customers will connect, you can define a single cluster of servers. For each of these servers, you configure a port through which the Dispatcher communicates. See Figure 12 on page 45.

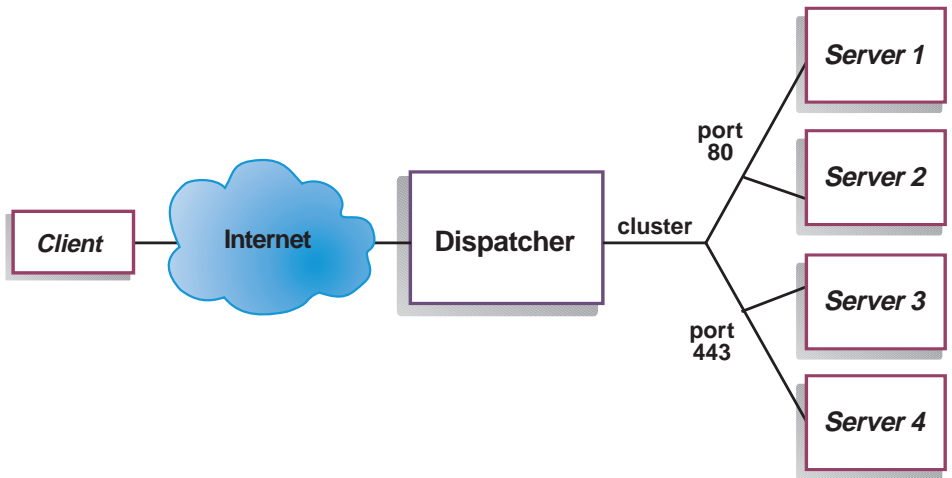


Figure 12. Example of Dispatcher configured with a single cluster and 2 ports

In this example, at www.productworks.com, one cluster is defined. This cluster has two ports: port 80 for HTTP and port 443 for SSL. A client making a request to <http://www.productworks.com> (port 80) would go to a different server than a client requesting <https://www.productworks.com> (port 443).

Another way of configuring Dispatcher would be appropriate if you have a very large site with many servers dedicated to each protocol supported. In this case, you might want to define a cluster for each protocol with a single port but with many servers, as shown in Figure 13 on page 46.

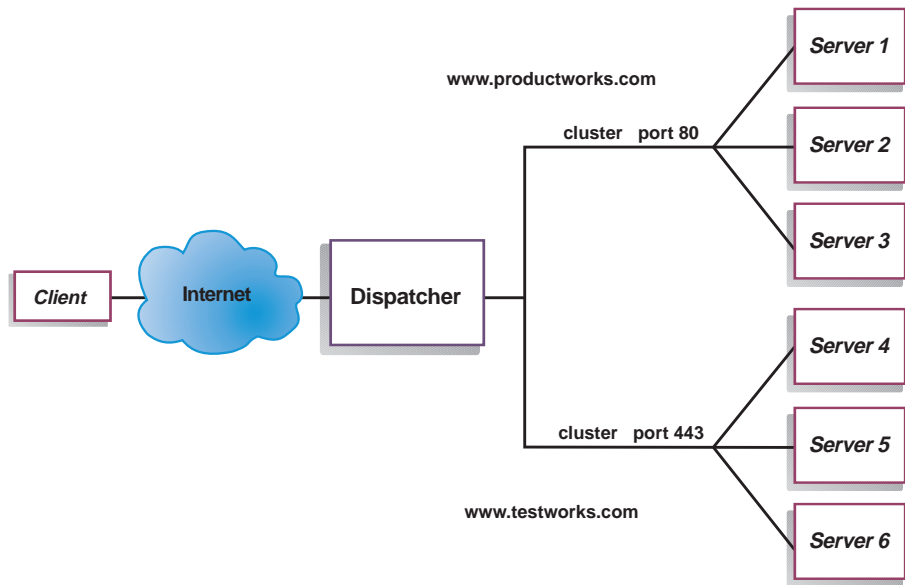


Figure 13. Example of Dispatcher configured with two clusters, each with one port

In this example, two clusters are defined: `www.productworks.com` for port 80 (HTTP) and `www.testworks.com` for port 443 (SSL).

A third way of configuring the Dispatcher would be necessary if your site does content hosting for several companies or departments, each one coming into your site with a different URL. In this case, you might want to define a cluster for each company or department and any ports to which you want to receive connections at that URL, as shown in Figure 14 on page 47.

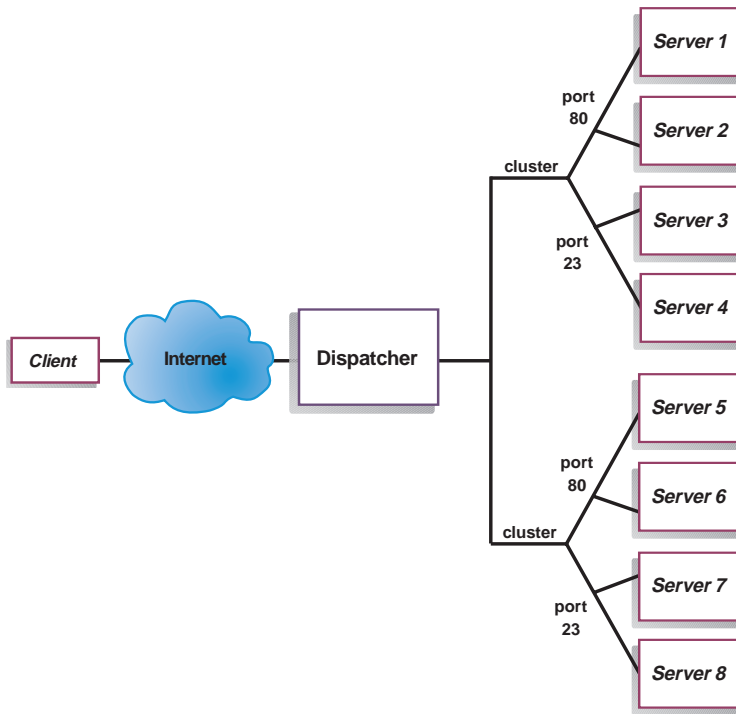


Figure 14. Example of the Dispatcher configured with 2 clusters, each with 2 ports

For example, two clusters are defined with port 80 for HTTP and port 23 for Telnet for each of the sites at www.productworks.com and www.testworks.com.

High availability

Simple high availability

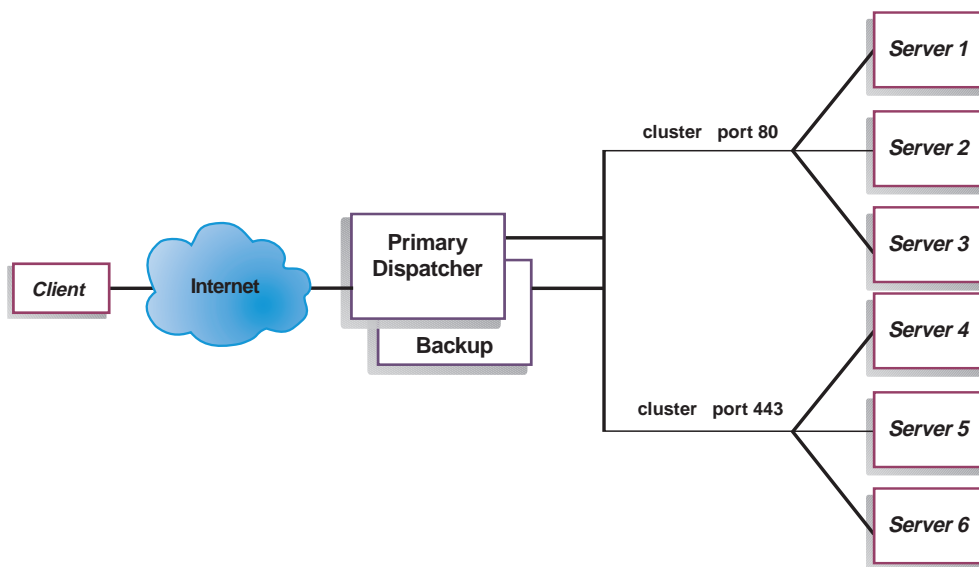


Figure 15. Example of a Dispatcher using simple high availability

The high availability feature involves the use of a second Dispatcher machine. The first Dispatcher machine performs load balancing for all the client traffic as it would in a single Dispatcher configuration. The second Dispatcher machine monitors the “health” of the first, and will take over the task of load balancing if it detects that the first Dispatcher machine has failed.

Each of the two machines is assigned a specific role, either *primary* or *backup*. The primary machine sends connection data to the backup machine on an ongoing basis. While the primary is *active* (load balancing), the backup is in a *standby* state, continually updated and ready to take over, if necessary.

The communication sessions between the two machines are referred to as *heartbeats*. The heartbeats allow each machine to monitor the health of the other.

If the backup machine detects that the active machine has failed, it will take over and begin load balancing. At that point the *statuses* of the two machines are reversed: the backup machine becomes *active* and the primary becomes *standby*.

In the high availability configuration, both primary and backup machines must be on the same subnet.

For information about configuring high availability, see “High availability” on page 89.

Mutual high availability

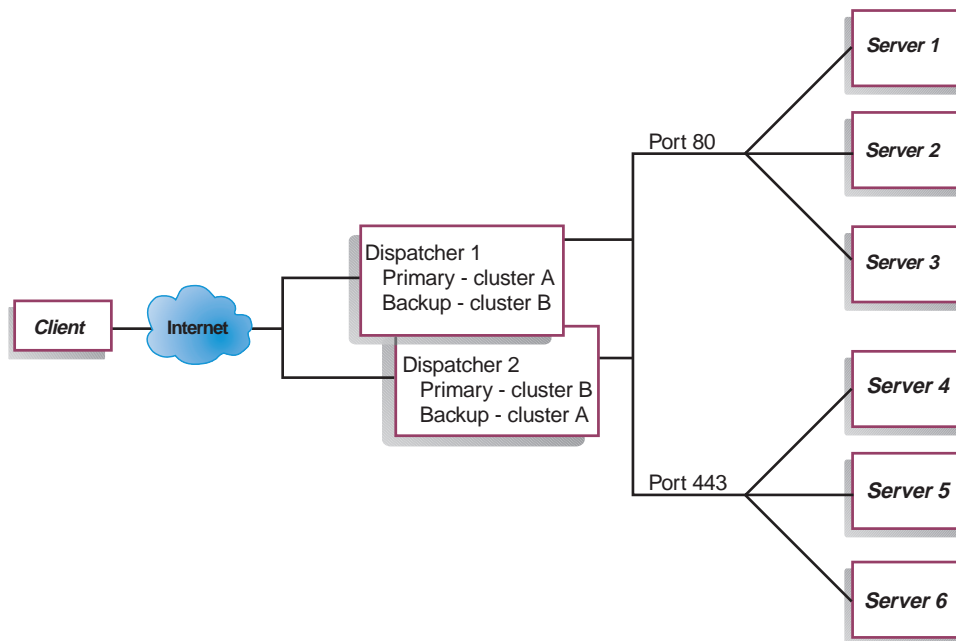


Figure 16. Example of a Dispatcher using mutual high availability

The mutual high availability feature involves the use of two Dispatcher machines. Both machines actively perform load balancing of client traffic, and both machines provide backup for each other. In a simple high availability configuration, only one machine performs load balancing. In a mutual high availability configuration, both machines load balance a portion of the client traffic.

For mutual high availability, client traffic is assigned to each Dispatcher machine on a cluster address basis. Each cluster can be configured with the NFA (nonforwarding address) of its primary Dispatcher. The primary Dispatcher machine normally performs load balancing for that cluster. In the event of a failure, the other machine performs load balancing for both its own cluster and for the failed Dispatcher’s cluster.

For an illustration of a mutual high availability configuration with shared “cluster set A” and shared “cluster set B,” see Figure 16 on page 49. Dispatcher 1 can actively route packets for its primary Cluster A, while Cluster B is considered the backup for the corresponding Cluster B in Dispatcher 2. At the same time, Dispatcher 2 can actively route packets for its primary Cluster B, while Cluster A is considered the backup for the corresponding Cluster A in Dispatcher 1. If Dispatcher 2 were to fail and could no longer actively route packets for its primary Cluster B, Dispatcher 1 could take over routing packets for Cluster B.

Note: Both machines must configure their shared cluster sets the same.

For information about configuring high availability and mutual high availability, see “High availability” on page 89.

Using Collocated Servers

On AIX and Solaris, Network Dispatcher can reside on the same machine as a server for which it is load balancing requests. This is commonly referred to as *collocating* a server. This feature is supported on Red Hat Linux, but only in the absence of using high availability. The Windows platform does not support collocation.

In older releases, it was necessary to specify the collocated server address to be the same as the nonforwarding address (NFA) in the configuration. That restriction has been lifted.

To configure a server to be collocated, the **ndcontrol server** command provides an option called **collocated** which can be set to yes or no. The default is no. The address of the server must be a valid IP address of a network interface card on the machine. See “ndcontrol server — configure servers” on page 228, for more information on ndcontrol server command syntax.

Note: A collocated server competes for resources with Network Dispatcher during times of high traffic. However, in the absence of overloaded machines, using a collocated server offers a reduction in the total number of machines necessary to set up a load-balanced site.

Load Balancing Dispatcher with ISS

The ISS component may be used, in conjunction with a domain name server, to load balance requests among several Dispatcher machines. The Dispatcher machines would then load balance the requests to the optimal server. This creates a two-tiered form of load balancing. For a graphical overview and further explanation, see “Using ISS and Dispatcher in a two-tiered configuration” on page 136.

Chapter 5. Configuring the Dispatcher component

Before following the steps in this chapter, see “Chapter 4. Planning for the Dispatcher component” on page 41. This chapter explains how to create a basic configuration for the Dispatcher component of IBM Network Dispatcher. For more complex configurations or functions, see “Chapter 8. Advanced Dispatcher and CBR Functions” on page 83.

Overview of configuration tasks

Note: Before you begin the configuration steps in this table, ensure that your Dispatcher machine and all server machines are connected to the network, have valid IP addresses, and are able to ping one another.

Table 4. Configuration tasks for the Dispatcher function

Task	Description	Related information
Set up the Dispatcher machine.	Set up your load balancing configuration.	“Setting up the Dispatcher machine” on page 55
Set up machines to be load-balanced.	Alias the loopback device, check for an extra route, and delete any extra routes.	“Setting up server machines for load balancing” on page 60

Methods of configuration

There are four basic methods of configuring the Dispatcher:

- Command line
- Scripts
- Graphical user interface (GUI)
- Configuration wizard

Command line

This is the most direct means of configuring the Dispatcher. The procedures in this manual assume use of the command line. The command is **ndcontrol**. For more information about commands, see “Appendix B. Command reference for Dispatcher and CBR” on page 191.

Scripts

The commands for configuring the Dispatcher can be entered into a configuration script file and executed together. See “Sample Network Dispatcher configuration files” on page 251.

GUI

For an example of the GUI, see Figure 2 on page 6.

To start the graphical user interface (GUI), follow these steps:

1. Ensure ndserver is running.
 - For AIX, Red Hat Linux, or Solaris, run the following as root:
ndserver
 - For Windows, ndserver runs as a service that starts automatically.
2. Next, do one of the following:
 - For AIX, Red Hat Linux, or Solaris: enter **ndadmin**.
 - For Windows: click **Start**, click **Programs**, and click **IBM Network Dispatcher**.

In order to configure the Dispatcher component from the GUI, you must first select **Dispatcher** in the tree structure. You can start the executor and manager once you connect to a Host. You can also create clusters containing port and server, and start advisors for the manager.

The GUI can be used to do anything that you would do with the **ndcontrol** command. For example, to define a cluster using the command line, you would enter **ndcontrol cluster add cluster** command. To define a cluster from the GUI, you would click button number two on the mouse on executor, then in the pop-up menu, click on **add cluster**. Enter the cluster address in the pop-up window, then click **OK**.

Pre-existing Dispatcher configuration files can be loaded using the **Load New Configuration** and **Append to Current Configuration** options presented in the **Host** pop-up menu. You should save your Dispatcher configuration to a file periodically using the **Save Configuration File As** option also presented in the **Host** pop-up menu. The **File** menu located at the top of the GUI will allow you to save your current host connections to a file or restore connections in existing files across all IBM Network Dispatcher components.

The configuration commands can also be run remotely. For more information, see “Remote Authenticated Administration” on page 161.

You can access **Help** by clicking the question mark icon in the upper right hand corner of the Network Dispatcher window.

- **Field Help** — describes each field, default values
- **How do I** — lists tasks that can be done from that screen
- **Contents** — a table of contents of all the Help information
- **Index** — an alphabetical index of the help topics

For more information about using the GUI, see “General Instructions for using the GUI” on page 5.

Configuration Wizard

For more information about using the configuration wizard, see “Configuring using the configuration wizard” on page 4.

Setting up the Dispatcher machine

Before setting up the Dispatcher machine, you must be the root user (for AIX, Red Hat Linux, or Solaris) or the Administrator on Windows.

For AIX, Red Hat Linux, and Solaris only, the Network Dispatcher can have a **collocated** server. This simply means that the Network Dispatcher can physically reside on a server machine which it is load balancing.

You will need at least two valid IP addresses for the Dispatcher machine:

- An IP address specifically for the Dispatcher machine

This IP address is the primary IP address of the Dispatcher machine and is called the nonforwarding address (NFA). This is by default the same address as that returned by the **hostname** command. Use this address to connect to the machine for administrative purposes, such as doing remote configuration via Telnet or accessing the SNMP subagent. If the Dispatcher machine can already ping other machines on the network, you do not need to do anything further to set up the nonforwarding address.

- One IP address for each cluster

A cluster address is an address that is associated with a host name (such as `www.yourcompany.com`). This IP address is used by a client to connect to the servers in a cluster. This is the address that is load balanced by the Dispatcher.

Solaris Only: By default, Dispatcher is configured to load balance traffic on 100Mbps Ethernet network interface cards. To use a 10Mbps Ethernet adapter, you need to edit the `/opt/nd/dispatcher/ibmnd.conf` file. The **ibmnd.conf** file, which specifies the interface cards used to support Dispatcher, provides input to the Solaris **autopush** command and must be compatible with the autopush command. The default 100Mbps Ethernet adapter is specified in `/opt/nd/dispatcher/ibmnd.conf` as `hme`. To use a 10Mbps Ethernet adapter, replace `hme` with `le`. To support multiple types of adapters, replicate the line in the `ibmnd.conf` file and modify each line to match your device type. For example, if you plan to use two 100Mbps Ethernet adapters, the `ibmnd.conf` file should have a single line specifying the `hme` device. If you plan to use one 10Mbps Ethernet adapter and one 100Mbps Ethernet adapter, you will have two lines in the `ibmnd.conf` file: one line specifying the `le` device and one line specifying the `hme` device.

Windows Only: Ensure that IP forwarding is not enabled for the TCP/IP protocol. (See your Windows TCP/IP configuration.)

Figure 17 shows an example of Dispatcher set up with a single cluster, two ports, and three servers.

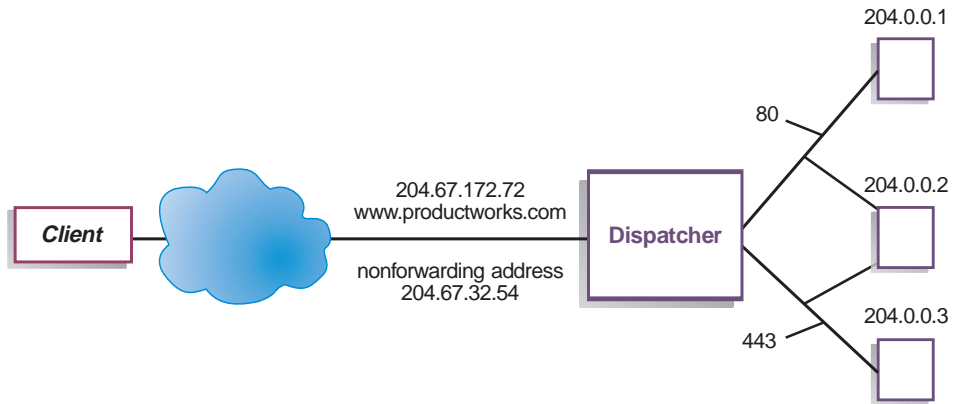


Figure 17. Example of the IP addresses needed for the Dispatcher machine

For help with commands used in this procedure, see “Appendix B. Command reference for Dispatcher and CBR” on page 191.

For a sample configuration file, see “Sample Network Dispatcher configuration files” on page 251.

Step 1. Start the server function

AIX, Red Hat Linux, and Solaris: To start the server function, enter **ndserver**.

Windows: The server function starts automatically as a service.

Note: A default configuration file (default.cfg) gets automatically loaded when starting ndserver. If the user decides to save the Dispatcher configuration in default.cfg, then everything saved in this file will be automatically loaded next time ndserver gets started.

Step 2. Start the executor function

To start the executor function, enter the **ndcontrol executor start** command. You may also change various executor settings at this time. See “Appendix B. Command reference for Dispatcher and CBR” on page 191.

Step 3. Define the nonforwarding address (if different from hostname)

The nonforwarding address is used to connect to the machine for administrative purposes, such as using Telnet or SMTP to this machine. By default, this address is the hostname.

To define the nonforwarding address, enter the **ndcontrol executor set nfa *IP_address*** command or edit the sample configuration file. *IP_address* is either the symbolic name or the dotted-decimal address.

Step 4. Define a cluster and set cluster options

Dispatcher will balance the requests sent to the cluster address to the corresponding servers configured on the ports for that cluster.

The cluster is either the symbolic name, the dotted decimal address, or the special address 0.0.0.0 that defines a wildcard cluster. To define a cluster, issue the command **ndcontrol cluster add**. To set cluster options, issue the command **ndcontrol cluster set** or you can use the GUI to issue commands. Wildcard clusters can be used to match multiple IP addresses for incoming packets to be load balanced. See “Use wildcard cluster to combine server configurations” on page 114, “Use wildcard cluster to load balance firewalls” on page 115, “Use wildcard cluster with WTE for transparent proxy” on page 116 for more information.

Step 5. Alias the network interface card

Once the cluster has been defined, you normally must configure the cluster address on one of the network interface cards of the Dispatcher machine. To do this, issue the command **ndcontrol cluster configure cluster_address**. This will look for an adapter with an existing address that belongs on the same subnet as the cluster address. It will then issue the system’s adapter configuration command for the cluster address, using the adapter found and the netmask for the existing address found on that adapter.

```
ndcontrol cluster configure 204.67.172.72
```

Circumstances where you would not want to configure the cluster address are for clusters added to a standby server in high-availability mode, or clusters added to a wide-area dispatcher acting as a remote server. You also do not need to run the cluster configure command if, in stand-alone mode, you use the sample **goldle** script. For information on the goldle script, see “Using scripts” on page 92.

In rare cases you may have a cluster address that does not match any subnet for existing addresses. In this case, use the second form of the cluster configure command and explicitly provide the interface name and netmask. Use **ndcontrol cluster configure cluster_address interface_name netmask**.

Some examples are:

```
ndcontrol cluster configure 204.67.172.72 en0 255.255.0.0
(AIX)
ndcontrol cluster configure 204.67.172.72 eth0:1 255.255.0.0
(Red Hat Linux)
```

```
ndcontrol cluster configure 204.67.172.72 1e0:1 255.255.0.0
(Solaris)
ndcontrol cluster configure 204.67.172.72 en0 255.255.0.0
(Windows)
```

Windows

In order to use the second form of the cluster configure command on Windows, you must determine the interface name to use.

If you have only one Ethernet card in your machine, the interface name will be en0. Likewise, if you have only one Token Ring card, the interface name will be tr0. If you have multiple cards of either type, you will need to determine the mapping of the cards. Use the following steps:

1. Start **regedit** at the command prompt.
2. Click **HKEY_LOCAL_MACHINE**, click **Software**, click **Microsoft**, click **Windows NT**, click **Current Version**.
3. And then, click **Network Cards**.

The network interface adapters are listed under Network Cards. Click each one to determine whether it is an Ethernet or Token Ring interface. The type of interface is listed in the *Description* column. The names assigned by **ndconfig** map to the interface types. For example, the first Ethernet interface in the list is assigned by ndconfig to en0, the second to en1, and so on; the first Token Ring interface is assigned to tr0, the second to tr1, and so on.

Note: The Windows registry begins numbering adapters with **1**, not **0**.

After you obtain this mapping information, you can create an alias on the network interface to the cluster address.

Using ifconfig/ndconfig to configure cluster aliases

The cluster configure command merely runs ifconfig (or ndconfig on NT) commands, so you can still use the ifconfig (ndconfig) commands if you wish.

Windows: The ndconfig command is supplied with the Dispatcher component to configure cluster aliases using the command line. The ndconfig command has the same syntax as a UNIX ifconfig command.

```
ndconfig en0 alias 204.67.172.72 netmask 255.255.0.0
```

Note: The netmask parameter is required. It should be in dotted-decimal (255.255.0.0) or hex (0xffff0000) form.

To determine the interface name, use the same technique as for the second form of the cluster configure command.

Solaris: When using bind-specific server applications that bind to a list of IP addresses that do not contain the server's IP, use **arp publish** command instead of **ifconfig** to dynamically set an IP address on the Network Dispatcher machine. For example:

```
arp -s <cluster> <Network Dispatcher MAC address> publish
```

Step 6. Define ports and set port options

To define a port, enter the **ndcontrol port add** *cluster:port* command, edit the sample configuration file, or use the GUI. *Cluster* is either the symbolic name or the dotted-decimal address. *Port* is the number of the port you are using for that protocol.

Port number 0 (zero) is used to specify a wildcard port. This port will accept traffic for a port that is not destined for any of the defined ports on the cluster. See “Use wildcard port to direct unconfigured port traffic” on page 116 for more information about when you might want to use a wildcard port.

You may also change various port settings at this time. See “Appendix B. Command reference for Dispatcher and CBR” on page 191. You must define and configure all servers for a port. The wildcard port will be used to configure rules and servers for any port. This function could also be used if you have an identical server/rule configuration for multiple ports. The traffic on one port could then affect the load-balancing decisions for traffic on other ports.

Step 7. Define load-balanced server machines

To define a load-balanced server machine, enter the **ndcontrol server add** *cluster:port:server* command, edit the sample configuration file, or use the GUI. *Cluster* and *server* are either the symbolic name or the dotted-decimal address. *Port* is the number of the port you are using for that protocol. You must define more than one server to a port on a cluster in order to perform load balancing.

Multiple address collocation: In a collocated configuration, the address of the collocated server machine does *not* have to be identical to the nonforwarding address (NFA). You can use another address if your machine has been defined with multiple IP addresses. For the Dispatcher component, the collocated server machine must be defined as **collocated** using the **ndcontrol server** command.

For more information on **ndcontrol server** command syntax, see “**ndcontrol server** — configure servers” on page 228.

For more information on collocated servers, see “Using Collocated Servers” on page 50.

Step 8. Start the manager function (optional)

The manager function improves load balancing. To start the manager, enter the **ndcontrol manager start** command, edit the sample configuration file, or use the GUI.

Step 9. Start the advisor function (optional)

The advisors give the manager more information about the ability of the load-balanced server machines to respond to requests. The advisors currently available are listed below along with their default ports:

Advisor Name	Protocol	Port
ftp	FTP	21
telnet	Telnet	23
smtp	SMTP	25
http	HTTP	80
imap	IMAP	143
pop3	POP3	110
nntp	NNTP	119
ssl	SSL	443
Workload Manager	private	10,007
WTE	HTTP	80
PING	ping	0

To start the advisors, enter the **ndcontrol advisor start *name port*** command, edit the sample configuration file, or use the GUI. *name* refers to the protocol that you want to monitor. *port* is the number of the port you are using for that protocol. The FTP advisor should advise only on the FTP control port (21). Do not start an FTP advisor on the FTP data port (20).

Step 10. Set manager proportions as required

If you have started any advisors, you must change the manager proportions to allow the advisor information to be included in the load balancing decisions. You should use the **ndcontrol manager proportions** command. See “Proportion of importance given to status information” on page 85.

Setting up server machines for load balancing

If your server is collocated on your Dispatcher machine, do **not** perform the following procedure.

Follow the steps in this section to set up the load-balanced server machines.

Step 1. Alias the loopback device

For the load-balanced server machines to work, you must set (or preferably alias) the loopback device (often called lo0) to the cluster address. The Dispatcher component does not change the destination IP address in the TCP/IP packet before forwarding the packet to a TCP server machine. By setting or aliasing the loopback device to the cluster address, the load balanced server machines will accept a packet that was addressed to the cluster address.

If you have an operating system that supports network interface aliasing (such as AIX, Linux, Solaris, Windows NT or Windows 2000), you should alias the loopback device to the cluster address. The benefit of using an operating system that supports aliases is that you have the ability to configure the load-balanced server machines to serve multiple cluster addresses.

Note: For **Linux** servers only, in order to alias the loopback device a specific patch is required for Linux kernel versions 2.2.5, 2.2.12, and 2.2.13. See “Installing the Linux kernel patch (for aliasing the loopback device)” on page 65, for more information.

If you have a server with an operating system that does not support aliases, such as HP-UX and OS/2, you must set the loopback device to the cluster address.

Use the command for your operating system as shown in Table 5 to set or alias the loopback device.

Table 5. Commands to alias the loopback device (lo0) for Dispatcher

AIX	ifconfig lo0 alias <i>cluster_address</i> netmask <i>netmask</i>
HP-UX	ifconfig lo0 <i>cluster_address</i>
Linux	ifconfig lo:1 <i>cluster_address</i> netmask 0.0.0.0 up
OS/2	ifconfig lo <i>cluster_address</i>
Solaris	ifconfig lo0:1 <i>cluster_address</i> 127.0.0.1 up

Table 5. Commands to alias the loopback device (lo0) for Dispatcher (continued)

Windows NT	<ol style="list-style-type: none"> 1. Click Start, then click Settings. 2. Click Control Panel, then double-click Network. 3. If you have not done so already, add the MS Loopback Adapter Driver. <ol style="list-style-type: none"> a. In the Network window, click Adapters. b. Select MS Loopback Adapter, then click OK. c. When prompted, insert your installation CD or disks. d. In the Network window, click Protocols. e. Select TCP/IP Protocol, then click Properties. f. Select MS Loopback Adapter, then click OK. 4. Set the loopback address to your cluster address. Accept the default subnet mask (255.0.0.0) and do not enter a gateway address. <p>Note: You may have to exit and reenter Network Settings before the MS Loopback Driver shows up under TCP/IP Configuration.</p>
------------	--

Table 5. Commands to alias the loopback device (lo0) for Dispatcher (continued)

Windows 2000	<ol style="list-style-type: none"> 1. Click Start, click Settings, then click Control Panel. 2. If you have not done so already, add the MS Loopback Adapter Driver. <ol style="list-style-type: none"> a. Double-click Add/Remove Hardware. This launches the Add/Remove Hardware Wizard. b. Click Next, select Add/Troubleshoot a Device, then click Next. c. The screen blinks off/on, then presents the Choose a Hardware Device panel. d. If the MS Loopback Adapter is in the list, it is already installed— click Cancel to exit. e. If the MS Loopback Adapter is <i>not</i> in the list— select Add a New Device and click Next. f. To select the hardware from a list, for the Find New Hardware panel, click No and then click Next. g. Select Network Adapters and click Next. h. On the Select Network Adapter panel, select Microsoft in the Manufacturers list, then select Microsoft Loopback Adapter. i. Click Next, then click Next again to install the default settings (or select Have Disk, then insert CD and install from there). j. Click Finish to complete installation. 3. From the Control Panel, Double-click Network and Dial-up Connections. 4. Select the connection with Device Name “Microsoft Loopback Adapter” and right-click on it. 5. Select Properties from the dropdown. 6. Select Internet Protocol (TCP/IP), then click Properties. 7. Click Use the following IP address. Fill in <i>IP address</i> with the cluster address, and <i>Subnet mask</i> with the default subnet mask (255.0.0.0). Note: Don’t enter a gateway address. Use the localhost as the default DNS server.
--------------	--

Table 5. Commands to alias the loopback device (lo0) for Dispatcher (continued)

OS/390	<p>Configuring a loopback alias on OS/390 system</p> <ul style="list-style-type: none"> In the IP parameter member (file), an Administrator will need to create an entry in the Home address list. For example <pre>HOME ;Address Link 192.168.252.11 tr0 192.168.100.100 ltr1 192.168.252.12 loopback</pre> <ul style="list-style-type: none"> Several addresses can be defined for the loopback. The 127.0.0.1 is configured by default.
--------	--

Using the example shown in Figure 17 on page 56, and setting up a load-balanced server machine that is running Solaris, the command would be:

```
ifconfig lo0:1 204.67.172.72 127.0.0.1 up
```

Step 2. Check for an extra route

On some operating systems, a default route may have been created and needs to be removed.

- Check for an extra route on Windows with the following command:

```
route print
```

- Check for an extra route on all UNIX systems with the following command:

```
netstat -nr
```

Windows Example:

- After **route print** is entered, a table similar to the following will be displayed. (This example shows finding and removing an extra route to cluster 9.67.133.158 with a default netmask of 255.0.0.0.)

Active Routes:

Network Address	Netmask	Gateway Address	Interface	Metric
0.0.0.0	0.0.0.0	9.67.128.1	9.67.133.67	1
9.0.0.0	255.0.0.0	9.67.133.158	9.67.133.158	1
9.67.128.0	255.255.248.0	9.67.133.67	9.67.133.67	1
9.67.133.67	255.255.255.255	127.0.0.1	127.0.0.1	1
9.67.133.158	255.255.255.255	127.0.0.1	127.0.0.1	1
9.255.255.255	255.255.255.255	9.67.133.67	9.67.133.67	1
127.0.0.0	255.0.0.0	127.0.0.1	127.0.0.1	1
224.0.0.0	224.0.0.0	9.67.133.158	9.67.133.158	1
224.0.0.0	224.0.0.0	9.67.133.67	9.67.133.67	1
255.255.255.255	255.255.255.255	9.67.133.67	9.67.133.67	1

- Find your cluster address under the "Gateway Address" column. If you have an extra route, the cluster address will appear twice. In the example given, the cluster address (9.67.133.158) appears in row 2 and row 8.
- Find the network address in each row in which the cluster address appears. You need one of these routes and will need to delete the other

route, which is extraneous. The extra route to be deleted will be the one whose network address begins with the first digit of the cluster address, followed by three zeroes. In the example shown, the extra route is the one in row two, which has a network address of **9.0.0.0**:

```
9.0.0.0      255.0.0.0    9.67.133.158  9.67.133.158      1
```

Step 3. Delete any extra route

You must delete the extra route. Use the command for your operating system shown in Table 6 to delete the extra route.

Example: To delete the extra route as shown in the example, enter:

```
route delete 9.0.0.0 9.67.133.158
```

Table 6. Commands to delete any extra route for Dispatcher

HP-UX	route delete <i>cluster_address cluster_address</i>
Windows 2000/ NT	route delete <i>network_address cluster_address</i> (at an MS-DOS prompt) Note: For Windows, you must delete the extra route every time you reboot the server.

Using the example shown in Figure 17 on page 56, and setting up a server machine that is running AIX, the command would be:

```
route delete -net 204.0.0.0 204.67.172.72
```

Installing the Linux kernel patch (for aliasing the loopback device)

For Linux servers only, in order to alias the loopback device a specific patch (depending on the Linux kernel version) is required. A patch for Linux kernel versions 2.2.5, 2.2.12, and 2.2.13 can be downloaded at the following website: <http://www.ibm.com/developer/linux>.

Note: This Linux kernel patch is *not* code written by IBM. It was written by members of the open source community. IBM cannot warrant this code as suitable for your specific environment, because the conditions of its use are beyond IBM’s control. It was used to test the IBM product and was found to be satisfactory in the IBM testing environment. You should evaluate the usefulness of this code in your own environment, and decide if it meets your needs. This code may or may not be included in future versions of the Linux base source code.

The following are the steps for installing the Linux kernel patch for versions 2.2.12 and 2.2.13 on your Linux server machines:

1. Obtain the loopback patch from <http://www.ibm.com/developer/linux>.
2. Install the kernel source. For installation instructions, refer to the **README.kernel-sources** file in the `/usr/src/linux` directory.

3. Apply the patch by issuing the patch command from /usr/src directory.
For example:

```
patch -p0<patchfile
```
4. Compile the kernel. For compile instructions, refer to the **README** file in /usr/src/linux directory.
5. Install the new kernel and run the **lilo** command. For instructions, refer to the **README** file in /usr/src/linux directory.
6. Reboot with the new kernel.
7. Check for the following file: /proc/sys/net/ipv4/conf/lo/**arp_invisible** . If the file is present then the kernel was patched successfully. If the file is *not* present, then either the patch was unsuccessful, or an unpatched kernel was booted. Check /usr/src/linux/README to make sure all the installation steps were followed correctly.
8. Issue the comand:

```
echo 1 >> /proc/sys/net/ipv4/conf/lo/arp_invisible
```

This command will only last until the machine is rebooted. Once rebooted it will be necessary to follow this and the subsequent steps again.

9. Alias the loopback with a netmask of 0.0.0.0, for example:

```
ifconfig lo:1 cluster netmask 0.0.0.0 up
```
10. Add the server to your cluster.

Chapter 6. Planning for the Content Based Routing component

This chapter describes what the network planner should consider before installing and configuring the CBR component. If you want to plan for the Dispatcher component, see “Chapter 4. Planning for the Dispatcher component” on page 41. If you want to plan for the ISS component, see “Chapter 9. Planning for the Interactive Session Support component” on page 125. This chapter includes:

- “Hardware and software requirements”
- “Planning considerations” on page 69

Hardware and software requirements

Requirements for AIX

- Any IBM RS/6000-based machine
- IBM AIX version 4.3.2 or higher
- 50 MB of available disk space for installation

Note: Additional disk space will be needed for logs.

- The following Network Interface Cards (NICs) are supported:
 - 16 Mb Token ring
 - 10 Mb Ethernet
 - 100 Mb Ethernet
 - Fiber distributed data interface (FDDI)
- IBM Java development kit (JDK) version 1.1.8 or higher
- IBM Web Traffic Express (WTE) version 2.0 (or higher) if you are using the CBR component for load balancing HTTP traffic
- Netscape Navigator 4.07 (or higher) or Netscape Communicator 4.61 (or higher) for viewing online Help

Requirements for Red Hat Linux

- Red Hat Linux version 6.1 (Linux kernel version 2.2.12-20)
- 50 MB of available disk space for installation

Note: Additional disk space will be needed for logs.

- The following Network Interface Cards (NICs) are supported:
 - 10 Mb Ethernet

- 100 Mb Ethernet
- A version of the Korn Shell (ksh) must be installed
- IBM Java runtime environment (JRE) version 1.1.8 or higher (but not 1.2.x versions)
- The JAVA_HOME and PATH environment variables must be set using the **export** command. The content of JAVA_HOME variable is dependent on where the user has Java installed. Below is an example:
 - JAVA_HOME=/usr/local/jdk1.1.8
 - PATH=\$JAVA_HOME/bin:\$PATH
- IBM Web Traffic Express (WTE) version 3.0 (or higher) if you are using the CBR component for load balancing HTTP traffic
- Netscape Navigator 4.07 (or higher) or Netscape Communicator 4.61 (or higher)

Requirements for Solaris

- Any SPARC workstation supported by Solaris version 2.6 and Solaris version 7 (32 bit mode)
- 50 MB of available disk space for installation

Note: Additional disk space will be needed for logs.

- The following Network Interface Cards (NICs) are supported:
 - 10 Mb Ethernet
 - 100 Mb Ethernet
- IBM Java runtime environment (JRE) version 1.1.8 or higher (but not 1.2.x versions)
- IBM Web Traffic Express (WTE) version 2.0 (or higher) if you are using the CBR component for load balancing with HTTP traffic
- Sun Microsystems HotJava Browser 1.0.1 or higher for viewing online Help

Requirements for Windows 2000 and Windows NT

- Any Intel x86 PC supported by Microsoft Windows 2000; or Microsoft Windows NT, version 4.0 with Service Pack 5 (or higher)
- Windows 2000 Professional, Server, or Advanced Server; or Windows NT Workstation or Server
- 50 MB of available disk space for installation

Note: Additional disk space will be needed for logs.

- The following Network Interface Cards (NICs) are supported:
 - 16 Mb Token ring
 - 10 Mb Ethernet
 - 100 Mb Ethernet

- IBM Java runtime environment (JRE) version 1.1.8 or higher (but not 1.2.x versions)
- IBM Web Traffic Express (WTE) version 2.0 (or higher) if you are using the CBR component for load balancing HTTP traffic
- Ensure your default browser is either Netscape Navigator 4.07 (or higher), Netscape Communicator 4.61 (or higher), or Internet Explorer 4.0 (or higher). The default browser is used for viewing online Help.

Planning considerations

The CBR component allows you to:

- Proxy IMAP and POP3 traffic based on user ID and password of the client request (without WTE)
- Or, load balance HTTP traffic using WTE to proxy the request

Note: CBR *cannot* be configured for both (HTTP and IMAP/ POP3) types of traffic on the same Network Dispatcher machine.

CBR can be configured via the command line (cbrcontrol) or the graphical user interface (ndadmin).

CBR is very similar to Dispatcher in its component structure. CBR consists of the following functions:

- The **executor** supports load balancing of client requests. The executor always runs when the CBR component is being used.
- The **manager** sets weights used by the executor based on:
 - Internal counters in the executor
 - Feedback from the servers provided by the advisors
 - Feedback from a system-monitoring program, such as ISS or WLM.

Using the manager is optional. However, if the manager is not used, load balancing will be performed using weighted round-robin scheduling based on the current server weights, and advisors will not be available.

- The **advisors** query the servers and analyze results by protocol before calling the manager to set weights as appropriate. It may not make sense to use some of these advisors in a typical configuration. You also have the option of writing your own advisors. Using the advisors is optional but recommended.
- Both a command line and a graphical user interface are provided to configure and manage the executor, advisors, and manager.

The three key functions of CBR (executor, manager, and advisors) interact to balance and dispatch the incoming requests between servers. Along with load

balancing requests, the executor monitors the number of new connections and active connections and supplies this information to the manager.

CBR proxy (for IMAP or POP3)

To start CBR in proxy mode, issue the **cbrserver** command from the command prompt.

CBR (without WTE) can provide a single point of presence for many IMAP or POP3 servers. Each server can have a subset of all mailboxes serviced by the point of presence. For IMAP and POP3, CBR is a proxy that chooses an appropriate server based on user ID and password provided by the client.

Note: The CBR proxy does *not* support rules-based load balancing.

An example of a method for distributing requests based on client user ID is the following. If you have two (or more) POP3 servers, you can choose to divide the mailboxes alphabetically by user ID. Client requests with user ID's beginning with letters A–I can be distributed to server 1. Client requests with user ID's beginning with letters J–R can be distributed on server 2, and so on.

You can also choose to have each mailbox represented on more than one server. In that case, the content of each mailbox must be available to all servers with that mailbox. In the event of a server failure, another server can still access the mailbox.

In order to have only one address representing multiple POP3 mail servers, the CBR proxy can be configured with a single cluster address that becomes the POP3 mail server address for all clients. The commands to configure this are the following:

```
cbrcontrol cluster add pop3MailServer
cbrcontrol port add pop3MailServer:110 protocol pop3
cbrcontrol server add pop3MailServer:110:pop3Server1+pop3Server2+pop3Server3
```

In this example, *pop3MailServer* represents the cluster address. Port 110 with proxy protocol POP3 is added to the *pop3MailServer*. *Pop3Server1*, *pop3Server2*, and *pop3Server3* represent POP3 mail servers which are added to the port. With this configuration, you can configure your mail clients' incoming POP3 requests with the *pop3MailServer* cluster address.

When a POP3 request arrives at the proxy, the proxy attempts to contact all the configured servers for the port using the client's user ID and password. The client's request is directed to the first server that responds. You should use the sticky/affinity feature in conjunction with the CBR proxy for IMAP or POP3 servers. The affinity feature allows subsequent requests from the same client's user ID to be directed to the same server. Set **stickytime** for the port

to a value greater than zero to enable this affinity feature. For more information on the affinity feature, see “How classical ND affinity works” on page 116.

The inactivity autologout timer for POP3 and IMAP protocols is a minimum of 10 minutes and 30 minutes respectively. This timeout is the number of seconds during which there can be no activity on a connection before that connection is removed. To optimize performance, the CBR proxy overrides the inactivity timeout value to 60 seconds. In order to change the inactivity timeout, change the **staletimeout** value on the **cbrcontrol port** command. For information on configuring this command, see “ndcontrol port — configure ports” on page 219.

CBR with WTE (for HTTP)

The CBR component gives you the ability to specify a set of servers that will handle a request based on regular expression matching the content of the request. CBR allows you to partition your site so that different content or application services can be served by different sets of servers. This partitioning will be transparent to clients accessing your site. Because CBR allows you to specify multiple servers for each type of request, the requests can be load balanced for optimal client response. By allowing multiple servers to be assigned to each type of content, you are protected if one workstation or server fails. CBR will recognize the failure and continue to load balance client requests to the other servers in the set.

One way to divide your site would be to assign some servers to handle only cgi requests, and another set of servers to handle all other requests. This would stop compute intensive cgi scripts from slowing down the servers for normal html traffic, allowing clients to get better overall response time. Using this scheme, you could also assign more powerful workstations for normal requests. This would give clients better response time without the expense of upgrading all your servers.

You could also assign more powerful workstations for cgi requests.

Another possibility for partitioning your site could be to direct clients who are accessing pages requiring registration to one set of servers, and all other requests to a second set of servers. This would keep casual browsers of your site from tying up resources that could be used by clients who have committed to your registration. It would also allow you to use more powerful workstations to service those clients who have registered.

You could of course combine the methods above for even more flexibility, and improved service.

CBR runs as a subprocess of WTE, the caching proxy server, which must be installed on the same machine. CBR along with WTE filters Web page content using specified rule types. When running, WTE accepts client requests and queries the CBR component for the best server. Upon this query, CBR matches the request to a set of prioritized rules. When a rule is matched, an appropriate server is chosen from a preconfigured server set. Finally, CBR informs WTE which server was chosen and the request gets proxied there.

Once you have defined a cluster to be load balanced, you must make sure that all requests to that cluster have a rule that will choose a server. If no rule is found that matches a particular request, the client will receive an error page from WTE. The easiest way to ensure that all request will match some rule, is to create an always true rule at a very high priority number. Make sure that the servers used by this rule can handle all the requests not explicitly handled by the rules with lower priority numbers.

Chapter 7. Configuring the Content Based Routing component

Before following the steps in this chapter, see “Chapter 6. Planning for the Content Based Routing component” on page 67. This chapter explains how to create a basic configuration for the CBR component of IBM Network Dispatcher. For more complex configurations or functions, see “Chapter 8. Advanced Dispatcher and CBR Functions” on page 83.

Overview of configuration tasks

Note: Before you begin the configuration steps in this table, ensure that your CBR machine and all server machines are connected to the network, have valid IP addresses, and are able to ping one another.

Table 7. Configuration tasks for the CBR component

Task	Description	Related information
Set up the CBR machine.	Finding out about the requirements.	“Setting up the CBR machine” on page 75
Set up machines to be load-balanced.	Set up your load balancing configuration.	“Step 5. Define load balanced server machines” on page 79

Methods of configuration

To create a basic configuration for the CBR component of Network Dispatcher, there are three basic methods:

- Command line
- Scripts
- Graphical user interface (GUI)
- Configuration wizard

Command line

This is the most direct means of configuring CBR. The procedures in this manual assume use of the command line. The command is **cbrcontrol**. For more information about commands, see “Appendix B. Command reference for Dispatcher and CBR” on page 191.

Note: To start CBR in proxy mode for IMAP or POP3, first issue the **cbrserver** command from the command prompt. Then issue **cbrcontrol** commands you want to set up your configuration.

Scripts

The commands for configuring CBR can be entered into configuration script file and executed together.

GUI

For a diagram of the GUI, see Figure 2 on page 6.

For IMAP or POP3, prior to starting the GUI, start the CBR proxy by issuing **cbrserver** command from the command prompt.

For HTTP, prior to starting the GUI, Web Traffic Express (WTE) must be running. To start the GUI:

- Enter **ndadmin** (AIX, Red Hat Linux, or Solaris)
- Click **Start**, click **Programs**, and click **IBM Network Dispatcher** (Windows.)

In order to configure the CBR component from the GUI, you must first select **Content Based Routing** in the tree structure. You can start the manager once you connect to a Host. You can also create clusters containing ports and servers, and start advisors for the manager.

The GUI can be used to do anything that you would do with the **cbrcontrol** command. For example, to define a cluster using the command line, you would enter **cbrcontrol cluster add cluster** command. To define a cluster from the GUI, you would click button number two of the mouse on Executor, then in the pop-up menu, click on **add cluster**. Enter the cluster address in the pop-up window, then click **OK**.

Pre-existing CBR configuration files can be loaded using the **Load New Configuration** and **Append to Current Configuration** options presented in the **Host** pop-up menu. You should save your CBR configuration to a file periodically using the **Save Configuration File As** option also presented in the **Host** pop-up menu. The **File** menu located at the top of the GUI will allow you to save your current host connections to a file or restore connections in existing files across all Network Dispatcher components.

You can access **Help** by clicking the question mark icon in the upper right hand corner of the Network Dispatcher window.

- **Field Help** — describes each field, default values
- **How do I** — lists tasks that can be done from that screen
- **Contents** — a table of contents of all the Help information
- **Index** — an alphabetical index of the help topics

For more information about using the GUI, see “General Instructions for using the GUI” on page 5.

Configuration wizard

If you are using the configuration wizard, follow these steps:

1. Determine which type of traffic you want to load balance:
 - If you want to load balance HTTP traffic, you will need to start WTE
 - If you want to load balance IMAP or POP3 traffic, you will need to issue the **cbrserver** command at a Command Prompt
2. Start the wizard function of CBR, **cbrwizard**.

You can launch this wizard from the command prompt by issuing the **cbrwizard**. Or, select the Configuration Wizard from the CBR component menu as presented in the GUI.

The CBR wizard guides you step-by-step through the process of creating a basic configuration for the CBR component. It asks you questions about your network and guides you as you setup a cluster that enables CBR to load balance traffic between a group of servers.

With the CBR configuration wizard, you will see the following panels:

- Introduction to the wizard
- What to expect
- Before you start
- Choosing a host to configure (if necessary)
- Defining a cluster
- Adding a port
- Adding a server
- Adding a rule (HTTP traffic only)
- Starting an advisor

Setting up the CBR machine

Before setting up the CBR machine, you must be the root user (for AIX, Red Hat Linux, or Solaris) or the Administrator on Windows.

You will need one IP address for each cluster of servers that will be set up. A cluster address is an address that is associated with a host name (such as `www.company.com`). This IP address is used by a client to connect to the servers in a cluster. Specifically, this address is found in the URL request from the client. All requests made to the same cluster address are load balanced by CBR.

Step 1. Configure WTE to use CBR (for HTTP only)

WTE must be installed, and the WTE configuration file must be edited to use CBR. There are four entries:

- ServerInit
- PreExit
- PostExit
- ServerTerm

Each entry must be on a single line. The location of these four entries should be after the first instance of "ServerInit," which is commented out in the configuration file.

The comma delimited set of parameters refer to the following:

- client keys directory
- server key directory
- install path
- class path
- rmi port
- log directory
- save directory

Note: This set of parameters cannot contain spaces.

The specific additions to the configuration file for AIX, Red Hat Linux, Solaris, and Windows follow.

Figure 18. CBR configuration file for AIX

```
ServerInit /usr/lpp/nd/cbr/lib/libndcbr.so:ndServerInit
CBR_CLIENT_KEYS_DIRECTORY=/usr/lpp/nd/admin/keys/cbr,
CBR_SERVER_KEYS_DIRECTORY=/usr/lpp/nd/cbr/key,
END_INSTALL_PATH=/usr/lpp/nd,/usr/lpp/nd/cbr/lib:/usr/lpp/
nd/cbr/lib/ibmcbr.jar:/usr/lpp/nd/admin/lib/ChartRuntime.jar,
11099,/usr/lpp/nd/cbr/logs/,/usr/lpp/nd/cbr/
configurations/

PreExit /usr/lpp/nd/cbr/lib/libndcbr.so:ndPreExit

PostExit /usr/lpp/nd/cbr/lib/libndcbr.so:ndPostExit

ServerTerm /usr/lpp/nd/cbr/lib/libndcbr.so:ndServerTerm
```

Figure 19. CBR configuration file for Red Hat Linux

#NOTE: replace /usr/local/J1.1.8 with the directory where Java is installed.

```
ServerInit /opt/nd/cbr/lib/libndcbr.so:ndServerInit
CBR_CLIENT_KEYS_DIRECTORY=/opt/nd/admin/keys/cbr,
CBR_SERVER_KEYS_DIRECTORY=/opt/nd/cbr/key,END_INSTALL_PATH=
/opt/nd,/usr/local/J1.1.8/lib/classes.zip:/opt/nd/cbr/lib:/opt/
nd/cbr/lib/ibmcbr.jar:/opt/nd/admin/lib/ChartRuntime.jar,
11099,/opt/nd/cbr/logs/,/opt/nd/cbr/configurations/

PreExit /opt/nd/cbr/lib/libndcbr.so:ndPreExit

PostExit /opt/nd/cbr/lib/libndcbr.so:ndPostExit

ServerTerm /opt/nd/cbr/lib/libndcbr.so:ndServerTerm
```

Figure 20. CBR configuration file for Solaris

```
ServerInit /opt/nd/cbr/lib/libndcbr.so:ndServerInit
CBR_CLIENT_KEYS_DIRECTORY=/opt/nd/admin/keys/cbr,
CBR_SERVER_KEYS_DIRECTORY=/opt/nd/cbr/key,END_INSTALL_PATH=
/opt/nd,/opt/nd/cbr/lib:/opt/
nd/cbr/lib/ibmcbr.jar:/opt/nd/admin/lib/ChartRuntime.jar,
11099,/opt/nd/cbr/logs/,/opt/nd/cbr/configurations/

PreExit /opt/nd/cbr/lib/libndcbr.so:ndPreExit

PostExit /opt/nd/cbr/lib/libndcbr.so:ndPostExit

ServerTerm /opt/nd/cbr/lib/libndcbr.so:ndServerTerm
```

Figure 21. CBR configuration file for Windows

```
ServerInit c:\Progra~1\IBM\nd\cbr\lib\libndcbr.dll:
ndServerInit CBR_CLIENT_KEYS_DIRECTORY=c:\Progra~1\IBM\nd\
admin\keys\cbr,CBR_SERVER_KEYS_DIRECTORY=
c:\Progra~1\IBM\nd\cbr\key,END_INSTALL_PATH=
c:\Progra~1\IBM\nd,c:\wspp\IBM\nd\cbr\lib;
c:\Progra~1\IBM\nd\cbr\lib
\ibmcbr.jar;c:\Progra~1\IBM\nd\admin\lib\ChartRuntime.jar,
11099, c:\Progra~1\IBM\nd\cbr\logs\,
c:\Progra~1\IBM\nd\cbr\configurations\

PreExit c:\Progra~1\IBM\nd\cbr\lib\libndcbr.dll:ndPreExit

PostExit c:\Progra~1\IBM\nd\cbr\lib\libndcbr.dll:ndPostExit

ServerTerm c:\Progra~1\IBM\nd\cbr\lib\libndcbr.dll:
ndServerTerm
```

Step 2. Start WTE (for HTTP only)

- AIX: Add to your LIBPATH environment variable:
/usr/jdk_base/lib:/usr/jdk_base/lib/aix/native_threads:/usr/lpp/nd/cbr/lib
In the new environment, start WTE.
- Red Hat Linux:
 - Assign to your JAVA_HOME environment variable the directory where Java is installed, for example: /usr/local/J1.1.8
 - Add to your PATH environment variable: \$JAVA_HOME/bin:\$PATH
 - Add to your LD_LIBRARY_PATH environment variable:
opt/jre1.1.8/lib/linux/native_threads:/opt/nd/cbr/libIn the new environment, start WTE.
- Solaris: Add to your LD_LIBRARY_PATH environment variable:
/opt/jre1.1.8/lib/sparc/native_threads:/opt/nd/cbr/lib
In the new environment, start WTE.
- Windows: Add to your PATH environment variable:
c:\jre1.1.8\bin;c:\ProgramFiles\IBM\nd\cbr\lib
From the start button select Start->Programs->IBM Web Traffic Express or start the Web Traffic Express service from the services panel.

Step 3. Define a cluster and set cluster options

CBR will balance the requests sent for the cluster address to the corresponding servers configured on the ports for that cluster.

The cluster address is either a symbolic name or a dotted decimal address. For HTTP, this address will be located in the host portion of the URL.

To define a cluster, issue the following command:

```
cbrcontrol cluster add cluster
```

To set cluster options, issue the following command:

```
cbrcontrol cluster set cluster option value
```

For more information, see “Appendix B. Command reference for Dispatcher and CBR” on page 191.

Step 4. Define ports and set port options

The port number is the port that the server applications are listening on. For CBR with WTE running HTTP traffic, this is typically port 80. For CBR proxy with IMAP traffic, this is typically port 143. And, for CBR proxy with POP3 traffic, this is typically port 110.

For CBR w/WTE, to define a port to the cluster you defined in the previous step, issue the following:

```
cbrcontrol port add cluster:port
```

For CBR proxy, to define a port to the cluster you defined in the previous step, issue the following:

```
cbrcontrol port add cluster:port protocol [pop3|imap]
```

To set port options, issue the following:

```
cbrcontrol port set cluster:port option value
```

For more information, see “Appendix B. Command reference for Dispatcher and CBR” on page 191.

Step 5. Define load balanced server machines

The server machines are the machines running the applications that you want load balanced. The *server* is the symbolic name or dotted decimal address of the server machine. To define a server on the cluster and port from step 3, issue the following command:

```
cbrcontrol server add cluster:port:server
```

You must define more than one server per port on a cluster in order to perform load balancing.

Step 6. Add rules to your configuration (for HTTP only)

This is the key step in configuring CBR w/WTE for HTTP protocol. A rule defines how a URL request will be distinguished and sent to one of the appropriate set of servers. The special rule type used by CBR is called a content rule. To define a content rule, issue the following command:

```
cbrcontrol rule add cluster:port:rule type content pattern=pattern
```

The value *pattern* is the regular expression that will be compared to the URL in each client request. For more information on how to configure the pattern, see “Appendix D. Rule types for CBR” on page 249.

Some other rule types defined in Dispatcher can also be used in CBR for HTTP. For more information, see “Configure rules-based load balancing” on page 108.

Step 7. Add servers to your rules (for HTTP only)

When a rule is matched by a client request, the rule’s set of servers is queried for which server is best. The rule’s server set is a subset of the servers defined in the port. To add servers to a rule’s server set, issue the following command:

```
cbrcontrol rule useserver cluster:port:rule server
```

Step 8. Start the manager function (optional)

The manager function improves load balancing. To start the manager, issue the following command:

```
cbrcontrol manager start
```

Step 9. Start the advisor function (optional)

The advisors give the manager more information about the ability of the load balanced server machines to respond to requests. An advisor is specific to a protocol. The Network Dispatcher supplies HTTP, IMAP, and POP3 advisors for the three CBR supported protocols. For example, to start the HTTP advisor, issue the following command:

```
cbrcontrol advisor start http port
```

Step 10. Set manager proportions as required

If you have started any advisors, you must change the manager proportions to allow the advisor information to be included in the load balancing decisions. To set manager proportions, issue the **cbrcontrol manager proportions** command. For more information, see “Proportion of importance given to status information” on page 85.

CBR configuration example

To configure CBR with WTE for HTTP follow these steps:

1. Start up the command line interface: issue the **cbrcontrol** command.
2. The **cbrcontrol** prompt will be given. Issue the following commands.
(*cluster(c),port(p),rule(r),server(s)*)
 - cluster add c
 - port add c:p
 - server add c:p:s
 - rule add c:p:r type content pattern url=*
 - rule useserver c:p:r s
3. Set up a browser to use the WTE machine as a proxy server.
4. Load 'http://c/' into your browser where 'c' is the cluster you configured above.
 - Server 's' is invoked
 - The following Web page is displayed http://s/

To configure CBR proxy for IMAP or POP3 follow these steps:

1. Start the CBR proxy: issue the **cbrserver** command
2. Start up the command line interface: issue the **cbrcontrol** command.
3. The **cbrcontrol** prompt will be given. Issue the following commands.
(*cluster(c),port(p),server(s)*)

- `cluster add c`
- `port add c:p protocol [pop3 | imap]`
- `server add c:p:s`

Chapter 8. Advanced Dispatcher and CBR Functions

This chapter explains how to configure the load balancing parameters of Dispatcher and CBR and how to set up Dispatcher and CBR for advanced functions.

Table 8. Advanced configuration tasks for the Dispatcher and CBR functions

Task	Description	Related information
Optionally, change load-balancing settings	<p>You can change the following load-balancing settings:</p> <ul style="list-style-type: none">• Proportion of importance given to status information <p>The default ratio is 50-50-0-0. If you use the default, information from advisors and from ISS is not used.</p> <ul style="list-style-type: none">• Weights• Manager fixed weights• Manager intervals• Advisor intervals• Advisor timeouts• Sensitivity threshold• Smoothing index	"Optimizing the load balancing provided by Dispatcher and CBR" on page 84
Configure high availability or mutual high availability	Set up a second Dispatcher machine to provide a backup.	"High availability" on page 89
Configure wide area Dispatcher support	Set up a remote Dispatcher to load balance across a wide area network.	"Configure wide area Dispatcher support" on page 94
Create custom advisors	Write your own advisors for reporting on specific statuses of your servers.	"Create custom (customizable) advisors" on page 102
Use Server Monitor Agent	For Red Hat Linux, SMA provides server load information to the Dispatcher.	"Server Monitor Agent (SMA)" on page 106
Configure rules-based load balancing	Define conditions under which a subset of your servers will be used.	"Configure rules-based load balancing" on page 108
Use explicit linking	Avoid bypassing the Dispatcher in your links.	"Using explicit linking" on page 113
Use a private network	Configure the Dispatcher to load balance servers on a private network.	"Using a private network configuration" on page 113

Table 8. Advanced configuration tasks for the Dispatcher and CBR functions (continued)

Task	Description	Related information
Use wildcard cluster to combine common server configurations	Addresses that are not explicitly configured will use the wildcard cluster as a way to load balance traffic.	“Use wildcard cluster to combine server configurations” on page 114
Use wildcard cluster to load balance firewalls	All traffic will be load balanced to firewalls.	“Use wildcard cluster to load balance firewalls” on page 115
Use wildcard cluster with WTE for transparent proxy	Allows Dispatcher to be used to enable a transparent proxy.	“Use wildcard cluster with WTE for transparent proxy” on page 116
Use wildcard port to direct unconfigured port traffic	Handles traffic that is not configured for any specific port.	“Use wildcard port to direct unconfigured port traffic” on page 116
Use sticky affinity feature to configure a cluster’s port to be sticky	Allows client requests to be directed to the same server.	“How classical ND affinity works” on page 116
Use Server Directed Affinity API	Provides an API which allows an external agent to influence the Dispatcher affinity behavior	“Server Directed Affinity API to control client-server affinity” on page 117
Use cross port affinity to expand the sticky (affinity) feature across ports	Allows client requests received from different ports to be directed to the same server.	“Cross port affinity” on page 118
Use affinity address mask to designate a common IP subnet address	Allows clients requests received from the same subnet to be directed to the same server.	“Affinity address mask” on page 119
Use rule affinity override to provide a mechanism for a server to override the port sticky feature	Allows a server to override the stickytime setting on its port.	“Rule affinity override” on page 119
Use Cookie Affinity to load balance servers for CBR	Allows a session to maintain affinity for a particular server.	“Cookie affinity” on page 120
Use binary logging to analyze server statistics	Allows server information to be stored in and retrieved from binary files.	“Using binary logging to analyze server statistics” on page 121

Optimizing the load balancing provided by Dispatcher and CBR

The manager function of Dispatcher and CBR performs load balancing based on the following settings:

- “Proportion of importance given to status information” on page 85
- “Weights” on page 86

- “Manager intervals” on page 86
- “Advisor intervals” on page 87
- “Advisor timeouts” on page 87
- “Sensitivity threshold” on page 88
- “Smoothing index” on page 88

You can change these settings to optimize load balancing for your network.

Proportion of importance given to status information

The manager may use the following external factors in its weighting decisions:

- The number of active connections on each load balanced server machine (as tracked by the executor)
- The number of new connections on each load balanced server machine (as tracked by the executor)
- Input from the advisors
- Input from the system monitoring tools, such as ISS, SMA (for Red Hat Linux only), or WLM

Along with the current weight for each server and some other information required for its calculations, the manager gets the first two values from the executor. These values are based on information that is generated and stored internally in the executor.

You can change the relative proportion of importance to the manager of the four values. Think of the proportions as percentages; the sum of the relative proportions must equal 100%. The default ratio is 50/50/0/0, which ignores the advisor and system information. You may need to try different proportions to find the combination that gives the best performance.

The number of active connections is dependent upon the number of clients as well as the length of time necessary to use the services that are being provided by the load balanced server machines. If the client connections are quick (such as small Web pages served using HTTP GET), then the number of active connections will be fairly low. If the client connections are slower (such as a database query), then the number of active connections will be higher.

You should avoid setting the first two values too low. You will disable Dispatcher’s load balancing and smoothing unless you have the first two values set to at least 20 each.

Weights

Weights are set by the manager function based upon internal counters in the executor, feedback from the advisors, and feedback from a system-monitoring program, such as ISS. If you want to set weights manually, do not run the manager.

Weights are applied to all servers on a port. For any particular port, the requests will be distributed between servers based on their weights relative to each other. For example, if one server is set to a weight of 10, and the other to 5, the server set to 10 should get twice as many requests as the server set to 5.

To specify the maximum weight boundary that any server can have, enter the **ndcontrol port set maxweight** command. This command affects how much difference there can be between the number of requests each server will get. If you set the maximum weight to 1, then all the servers can have a weight of 1, 0 if quiesced, or -1 if marked down. As you increase this number, the difference in how servers can be weighted is increased. At a maximum weight of 2, one server could get twice as many requests as another. At a maximum weight of 10, one server could get 10 times as many requests as another. The default maximum weight is 20.

If an advisor finds that a server has gone down, it tells the manager, which sets the weight for the server to zero. As a result, the executor will not send any additional connections to that server as long as that weight remains zero. If there were any active connections to that server before the weight changed, they will be left to complete normally.

Manager fixed weights

Without the manager, advisors cannot be run and cannot detect if a server is down. If you choose to run the advisors, but do *not* want the manager to update the weight you have set for a particular server, use the **fixedweight** option on the **ndcontrol server** command. For example:

```
ndcontrol server set cluster:port:server fixedweight yes
```

Once **fixedweight** is set to yes, you can use the **ndcontrol server set weight** command to set the weight to the value you desire. The server weight value remains fixed while the manager is running until you issue another **ndcontrol server** command with **fixedweight** set to no. For more information, see “**ndcontrol server** — configure servers” on page 228.

Manager intervals

To optimize overall performance, the manager is restricted in how often it can interact with the executor. You can make changes to this interval by entering the **ndcontrol manager interval** and **ndcontrol manager refresh** commands.

The manager interval specifies how often the manager will update the server weights that the executor uses in routing connections. If the manager interval is too low, it can mean poor performance as a result of the manager constantly interrupting the executor. If the manager interval is too high, it can mean that the executor's request routing will not be based on accurate, up-to-date information.

For example, to set the manager interval to 1 second, enter the following command:

```
ndcontrol manager interval 1
```

The manager refresh cycle specifies how often the manager will ask the executor for status information. The refresh cycle is based on the interval time.

For example, to set the manager refresh cycle to 3, enter the following command:

```
ndcontrol manager refresh 3
```

This will cause the manager to wait for 3 intervals before asking the executor for status.

Advisor intervals

Note: The advisor defaults should work efficiently for the great majority of possible scenarios. Be careful when entering values other than the defaults.

The advisor interval sets how often an advisor asks for status from the servers on the port it is monitoring and then reports the results to the manager. If the advisor interval is too low, it can mean poor performance as a result of the advisor constantly interrupting the servers. If the advisor interval is too high, it can mean that the manager's decisions about weighting will not be based on accurate, up-to-date information.

For example, to set the interval to 3 seconds for the HTTP advisor for port 80, enter the following command:

```
ndcontrol advisor interval http 80 3
```

It does not make sense to specify an advisor interval that is smaller than the manager interval.

Advisor timeouts

To make sure that out-of-date information is not used by the manager in its load-balancing decisions, the manager will not use information from the advisor whose time stamp is older than the time set in the advisor timeout. The advisor timeout should be larger than the advisor polling interval. If the

timeout is smaller, the manager will ignore reports that logically should be used. By default, advisor reports do not time out.

Sensitivity threshold

The product provides other methods for you to optimize load balancing for your servers. To work at top speed, updates to the weights for the servers are only made if the weights have changed significantly. Constantly updating the weights when there is little or no change in the server status would create an unnecessary overhead. When the percentage weight change for the total weight for all servers on a port is greater than the sensitivity threshold, the manager updates the weights used by the executor to distribute connections. Consider, for example, that the total weight changes from 100 to 105. The change is 5%. With the default sensitivity threshold of 5, the manager will not update the weights used by the executor, because the percentage change is not **above** the threshold. If, however, the total weight changes from 100 to 106, the manager will update the weights. To set the manager's sensitivity threshold to a value other than the default (for example, 6), enter the following command:

```
ndcontrol manager sensitivity 6
```

In most cases, you will not need to change this value.

Smoothing index

The manager calculates the server weights dynamically. As a result, an updated weight can be very different from the previous one. Under most circumstances, this will not be a problem. Occasionally, however, it may cause an oscillating effect in the way the requests are load balanced. For example, one server can end up receiving most of the requests due to a high weight. The manager will see that the server has a high number of active connections and that the server is responding slowly. It will then shift the weight over to the free servers and the same effect will occur there too, creating an inefficient use of resources.

To alleviate this problem, the manager uses a smoothing index. The smoothing index limits the amount that a server's weight can change, effectively smoothing the change in the distribution of requests. A higher smoothing index will cause the server weights to change less drastically. A lower index will cause the server weights to change more drastically. The default value for the smoothing index is 1.5. At 1.5, the server weights can be rather dynamic. An index of 4 or 5 will cause the weights to be more stable. For example, to set the smoothing index to 4, enter the following command:

```
ndcontrol manager smoothing 4
```

In most cases, you will not need to change this value.

High availability

The high availability feature is only available for the Dispatcher component.

To improve Dispatcher availability, the Dispatcher high availability function uses the following mechanisms:

- Two Dispatchers with connectivity to the same clients, and the same cluster of servers, as well as connectivity between the Dispatchers. Both Dispatchers must be using the same operating system.
- A “heartbeat” mechanism between the two Dispatchers to detect Dispatcher failure.
- A list of reach targets, addresses that both Dispatcher machines must be able to contact in order to load balance traffic normally. For more information, see “Failure detection capability using heartbeat and reach target” on page 91.
- Synchronization of the Dispatcher information (that is, the connection tables, reachability tables, and other information).
- Logic to elect the active Dispatcher which is in charge of a given cluster of servers, and the standby Dispatcher which continuously gets synchronized for that cluster of servers.
- A mechanism to perform IP takeover, when the logic or an operator decides to switch active and standby.

Note: For an illustration and description of a *mutual high availability* configuration, where two Dispatcher machines sharing two cluster sets provide backup for each other, see “Mutual high availability” on page 49. Mutual high availability is similar to high availability but is based specifically on cluster address rather than on a Dispatcher machine as a whole. Both machines must configure their shared cluster sets the same.

Configure high availability

Complete syntax for **ndcontrol highavailability** is in “Appendix B. Command reference for Dispatcher and CBR” on page 191.

For a more complete discussion of many of the tasks below, see “Setting up the Dispatcher machine” on page 55.

1. Start the server on both Dispatcher server machines.
2. Start the executor on both machines.
3. Ensure that the nonforwarding address (NFA) of each Dispatcher machine is configured, and is a valid IP address for the subnet of the Dispatcher machines.

Windows only: In addition, configure each nonforwarding address using the **ndconfig** command. For example:

```
ndconfig en0 nfa_addr netmask netmask
```

4. Set up the cluster, port, and server information on both machines.

Note: For mutual high availability configuration (Figure 16 on page 49), for example, configure the cluster sets shared between the 2 Dispatchers as follows:

- For Dispatcher 1 issue:

```
ndcontrol cluster set clusterA primaryhost NFAdispatcher1  
ndcontrol cluster set clusterB primaryhost NFAdispatcher2
```

- For Dispatcher 2 issue:

```
ndcontrol cluster set clusterB primaryhost NFAdispatcher2  
ndcontrol cluster set clusterA primaryhost NFAdispatcher1
```

5. Start the manager and advisors on both machines. The reach advisor is started automatically when the manager starts.
6. Create alias script files on each of the 2 Dispatcher machines. See “Using scripts” on page 92.
7. Add the heartbeat information on both machines:

```
ndcontrol highavailability heartbeat add sourceaddress destinationaddress
```

Note: *Sourceaddress* and *destinationaddress* are the IP addresses (either DNSnames or dotted-decimal addresses) of the Dispatcher machines. The values will be reversed on each machine. Example:

```
Primary - highavailability heartbeat add 9.67.111.3  
9.67.186.8  
Backup - highavailability heartbeat add 9.67.186.8  
9.67.111.3
```

For mutual high availability, at least one heartbeat pair must have the NFAs of the pair as the source and destination address.

8. On both machines, configure the list of IP addresses that the Dispatcher must be able to reach in order to ensure full service, using the **reach add** command. Example:

```
ndcontrol highavailability reach add 9.67.125.18
```

Reach targets are recommended but not required. See “Failure detection capability using heartbeat and reach target” on page 91, for more information.

9. Add the backup information to each machine:

- For the **primary** machine:

```
ndcontrol highavailability backup add primary [auto | manual] portnum
```

- For the **backup** machine:

```
ndcontrol highavailability backup add backup [auto | manual] portnum
```

- For mutual high availability each Dispatcher machine has **both** primary and backup roles:

```
ndcontrol highavailability backup add both [auto | manual] portnum
```

Note: Select an unused port on your machines as the *portnum*. Your two machines will communicate over this port.

10. Check the high availability status on each machine:

```
ndcontrol highavailability status
```

The machines should each have the correct role (backup, primary, or both), states, and substates. The primary should be active and synchronized; the backup should be in standby mode and should be synchronized within a short time. The strategies must be the same.

Notes:

1. To configure a single Dispatcher machine to route packets without a backup, do not issue any of the high availability commands at startup.
2. To convert two Dispatcher machines configured for high availability to one machine running alone, stop the executor on one of the machines, then delete the high availability features (the heartbeats, reach, and backup) on the other.
3. In both of the two cases above, you must alias the network interface card with cluster addresses, as required.
4. When two Dispatcher machines are run in high availability configuration and are synchronized, it is recommended that you enter all `ndcontrol` commands on the standby machine first, and then on the active machine.
5. When running two Dispatcher machines in a high availability configuration, unexpected results may occur if you set any of the parameters for the executor, cluster, port, or server (for example, port stickytime) to different values on the two machines.
6. For mutual high availability, consider the case where one of the Dispatchers must actively route packets for its primary cluster as well as take over routing packets for the backup cluster. Ensure this will not exceed your capacity for throughput on this machine.

Failure detection capability using heartbeat and reach target

Besides the basic criteria of failure detection (the loss of connectivity between active and standby Dispatchers, detected through the heartbeat messages), there is another failure detection mechanism named “reachability criteria.” When you configure the Dispatcher you provide a list of hosts that each of the Dispatchers should be able to reach in order to work correctly.

You should choose at least one host for each subnet your Dispatcher machine uses. The hosts could be routers, IP servers or other types of hosts. Host reachability is obtained by the reach advisor, which pings the host. Switchover

takes place either if the heartbeat messages cannot go through, or if the reachability criteria are met better by the standby Dispatcher than by the primary Dispatcher. To make the decision based on all available information, the active Dispatcher regularly sends the standby Dispatcher its reachability capabilities. The standby Dispatcher then compares those capabilities with its own and decides whether to switch.

Recovery Strategy

Two Dispatcher machines are configured: the primary machine, and a second machine called the *backup*. At startup, the primary machine sends all the connection data to the backup machine until that machine is synchronized. The primary machine becomes *active*, that is, it begins load balancing. The backup machine, meanwhile, monitors the status of the primary machine, and is said to be in *standby* state.

If the backup machine at any point detects that the primary machine has failed, it performs a *takeover* of the primary machine's load balancing functions and becomes the active machine. After the primary machine has once again become operational, the machines respond according to how the recovery *strategy* has been configured by the user. There are two kinds of strategy:

Automatic

The primary machine resumes routing packets as soon as it becomes operational again.

Manual

The backup machine continues routing packets even after the primary becomes operational. Manual intervention is required to return the primary machine to active state and reset the backup machine to standby.

The strategy parameter must be set the same for both machines.

The manual recovery strategy allows you to force the routing of packets to a particular machine, using the takeover command. Manual recovery is useful when maintenance is being performed on the other machine. The automatic recovery strategy is designed for normal unattended operation.

For a mutual high availability configuration, there is no per cluster failure. If any problem occurs with one machine, even if it affects just one cluster, then the other machine will take over for both clusters.

Using scripts

For Dispatcher to route packets, each cluster address must be aliased to a network interface device.

- In a stand-alone Dispatcher configuration, each cluster address must be aliased to a network interface card (for example, en0, tr0).
- In a high availability configuration:
 - On the active machine, each cluster address must be aliased to a network interface card (for example, en0, tr0).
 - On the standby machine, each cluster address must be aliased to a loopback device (for example, lo0).
- In any machine in which the executor has been stopped, all aliases should be removed to prevent conflicts with another machine that may be started.

Since the Dispatcher machines will change states when a failure is detected, the commands above must be issued automatically. Dispatcher will execute user-created scripts to do that. The following scripts may be used. These scripts can be found in the **dispatcher/samples** subdirectory and must be moved to the **bin** subdirectory to be run.

Note: For a mutual high availability configuration, each “go” script will be called by the Dispatcher with a parameter identifying the primary Dispatcher address. The script must query this parameter and perform the **ifconfig** commands for those cluster addresses associated with that primary Dispatcher.

goActive

The goActive script is executed when a Dispatcher goes into active state and begins routing packets.

- If you run Dispatcher in a high availability configuration, you must create this script. This script deletes loopback aliases and adds device aliases.
- If you run Dispatcher in a stand-alone configuration, you do not need this script.

goStandby

The goStandby script is executed when a Dispatcher goes into standby state monitoring the health of the active machine, but not routing any packets.

- If you run Dispatcher in a high availability configuration, you must create this script. This script should delete device aliases and add loopback aliases.
- If you run Dispatcher in a stand-alone configuration, you do not need this script.

goInOp

The goInOp script is executed when a Dispatcher executor is stopped and before it is started for the first time.

- If you normally run Dispatcher in a high availability configuration, you may create this script. This script deletes all devices and loopback aliases.
- If you normally run Dispatcher in a standalone configuration, this script is optional. You may create it and have it delete device aliases, or you may choose to delete them manually.

goIdle The goldle script is executed when a Dispatcher goes into idle state and begins routing packets. This occurs when the high availability features have not been added, as in a stand-alone configuration. It also occurs in a high availability configuration before the high availability features have been added or after they have been removed.

- If you normally run Dispatcher in a high availability configuration, you should **not** create this script.
- If you normally run Dispatcher in a stand-alone configuration, this script is optional. You may create it and have it add device aliases, or you may choose to add them manually. If you do not create this script for your stand-alone configuration, you will have to use the `ndcontrol cluster configure` command or manually configure the aliases each time the executor is started.

Configure wide area Dispatcher support

This feature is only available for the Dispatcher component.

If you are not using the Dispatcher's wide area support, a Dispatcher configuration requires that the Dispatcher machine and its servers all be attached to the same LAN segment (see Figure 22). A client's packet comes into the ND machine and is sent to the server, and then from the server directly back to the client.

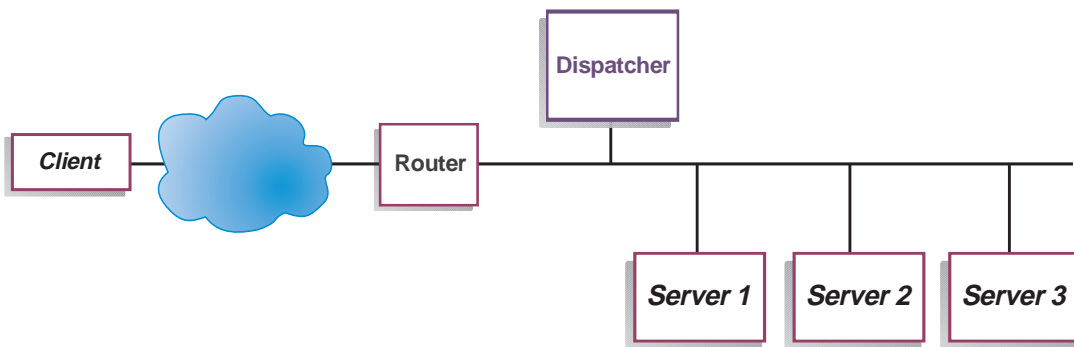


Figure 22. Example of a configuration consisting of a single LAN segment

The Wide Area Dispatcher enhancement adds support for offsite servers, known as *remote servers*. Servers on the same LAN segment as the Dispatcher machine will be referred to as local servers. A remote server consists of a remote Dispatcher machine and its locally attached servers. All the Dispatcher machines must be on the same operating system. A client's packet can now go from the Internet to a Dispatcher machine, from there to a geographically remote Dispatcher machine to one of its locally attached servers, and from the server directly back to the Internet and the client.

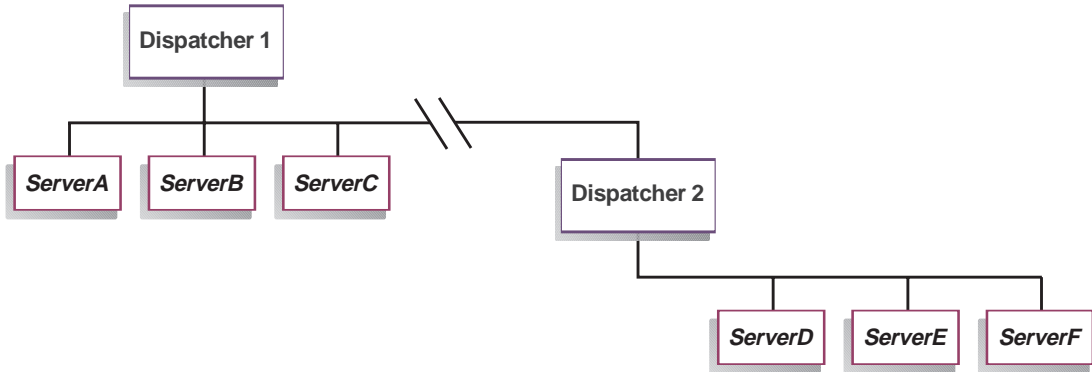


Figure 23. Example of configuration using local and remote servers

This allows one cluster address to support all worldwide client requests while distributing the load to servers around the world.

The Dispatcher machine initially receiving the packet can still have local servers attached to it and it can distribute the load between its local servers and the remote servers. See Figure 23.

Command Syntax

Wide area commands are not complex. To configure wide area support :

1. Select a port number that is unused on all Dispatcher machines. Assign this port number as the "Wide Area Port Number." The Dispatcher machines will use this port to exchange wide area information. To set the port number, issue the following command on all Dispatcher machines:
`ndcontrol executor set wideportnumber <port>`
2. Add the servers. When you add a server to a Dispatcher, you must define whether the server is local or remote (see above). To add a server and define it as local, issue the **ndcontrol server add** command without specifying a router. This is the default. To define the server as remote, you must specify the router through which Dispatcher must send the packet in order to reach the remote server. The server must be another Dispatcher and the server's address must be the nonforwarding address of the

Dispatcher. For example, in Figure 24 on page 97, if you are adding *ND 2* as a remote server under *ND 1*, you must define *router 1* as the router address. General syntax:

```
ndcontrol server add cluster:port:server router <address>
```

3. Configure aliases. On the first Dispatcher machine (where the client request arrives from the Internet), the cluster address must be aliased using **cluster configure**, **ifconfig** or **ndconfig**, as before. On the remote Dispatcher machines, however, the cluster address is **not** aliased to a network interface card.

Using remote advisors with wide area support

On entry-point Dispatchers, advisors will work correctly without any special configuration. On remote Dispatchers, you will need to perform the following configuration steps for each remote cluster address. For a high-availability configuration at the remote ND location, you must perform these steps on both machines.

AIX

- Alias the cluster address to the loopback adapter. For example:
ifconfig lo0 alias 9.67.34.123 netmask 255.255.240.0

Note: Advisors running on both the local and remote Dispatcher machines are necessary.

Red Hat Linux

- Alias the cluster address to the loopback adapter. For example:
ifconfig lo:1 alias 9.67.34.123 netmask 255.255.240.0 up

Note: Advisors running on both the local and remote Dispatcher machines are necessary.

Solaris

- No additional configuration steps are required.

Windows

1. Configure the loopback adapter with the remote cluster address as an alias. For example:
ndconfig lo0 alias 9.67.34.123 netmask 255.255.240.0
2. Delete any entries in the arp table for the remote cluster address.
 - a. To view the contents of the arp table, enter:
arp -a
 - b. To delete an entry if one exists, enter:
arp -d 9.67.34.123

Note: To determine the MAC address of your interface, enter:

- 1) **ping** <your_hostname>
- 2) **arp -a**

and look for the address of your machine.

3. Add a permanent entry to the arp table for the remote cluster address. Associate it with the MAC address of one of your main Token Ring or Ethernet devices. For example:

```
arp -s 9.67.34.123 00-01-5a-42-34-56
```

Configuration example

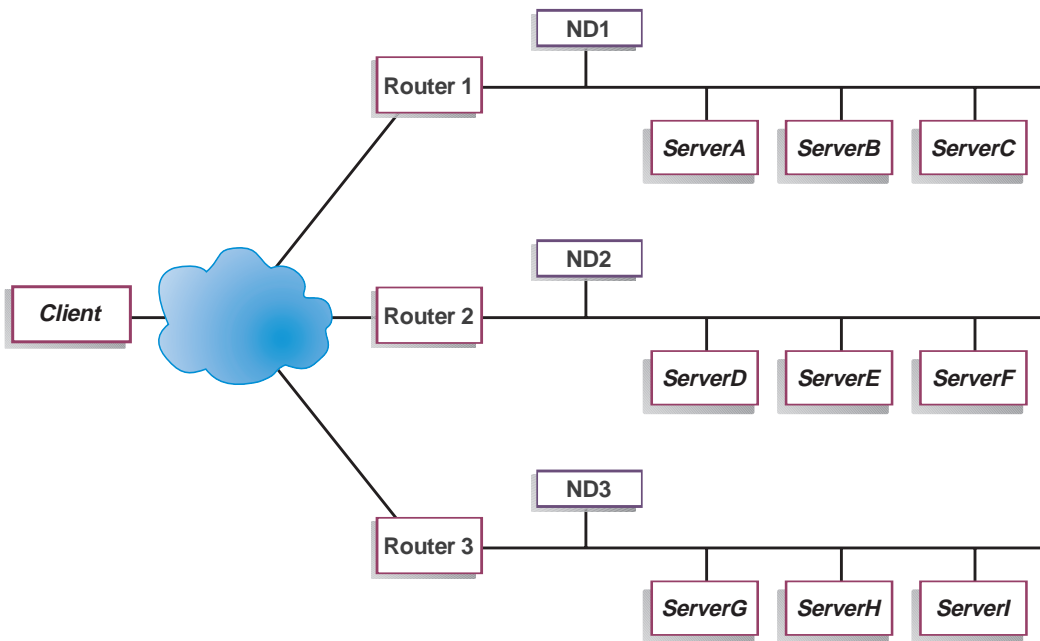


Figure 24. Wide area example configuration

This example applies to the configuration illustrated in Figure 24.

Here is how to configure the Dispatcher machines to support cluster address xebec on port 80. ND1 is defined as the “entry point,” and port 12345 is the wide area port. A token-ring LAN is assumed. Note that ND1 has five servers defined: three local and two remote. Remotes ND2 and ND3 each have three local servers defined.

At the console of the first Dispatcher (ND1), do the following:

1. Set the nonforwarding address of the Dispatcher machine.

- ndcontrol executor set nfa ND1**
- 2. Assign the wide area port number.
ndcontrol executor set wi 12345
- 3. Define the cluster.
ndcontrol cluster add xebec
- 4. Define the port.
ndcontrol port add xebec:80
- 5. Define the servers.
 - a. **ndcontrol server add xebec:80:ServerA**
 - b. **ndcontrol server add xebec:80:ServerB**
 - c. **ndcontrol server add xebec:80:ServerC**
 - d. **ndcontrol server add xebec:80:ND2 router R1**
 - e. **ndcontrol server add xebec:80:ND3 router R1**
- 6. If using Windows, configure the nfa of the Dispatcher LAN adapter.
ndcontrol cluster configure ND1 and also configure xebec as clusteraddr.
- 7. Configure the cluster address.
ndcontrol cluster configure xebec

At the console of the second Dispatcher (ND2):

- 1. Set the nonforwarding address of the Dispatcher machine.
ndcontrol executor set nfa ND2
- 2. Assign the wide area port number.
ndcontrol executor set wi 12345
- 3. Define the cluster.
ndcontrol cluster add xebec
- 4. Define the port.
ndcontrol port add xebec:80
- 5. Define the servers.
 - a. **ndcontrol server add xebec:80:ServerD**
 - b. **ndcontrol server add xebec:80:ServerE**
 - c. **ndcontrol server add xebec:80:ServerF**
- 6. If using Windows, configure the nfa of the Dispatcher LAN adapter.
ndcontrol cluster configure ND2

At the console of the third Dispatcher (ND3):

- 1. Set the nonforwarding address of the Dispatcher machine.
ndcontrol executor set nfa ND3
- 2. Assign the wide area port number.

ndcontrol executor set wi 12345

3. Define the cluster.

ndcontrol cluster add xebec

4. Define the port.

ndcontrol port add xebec:80

5. Define the servers.

a. **ndcontrol server add xebec:80:ServerG**

b. **ndcontrol server add xebec:80:ServerH**

c. **ndcontrol server add xebec:80:ServerI**

6. If using Windows, configure the nfa of the Dispatcher LAN adapter.

ndcontrol cluster configure ND3

Notes

1. On all servers (A-I), alias the cluster address to the loopback.
2. Clusters and ports are added with `ndcontrol` on all participating Dispatcher machines: the entry point Dispatcher and all remotes.
3. See “Using remote advisors with wide area support” on page 96 for help with using remote advisors with wide area support.
4. Wide area support prohibits infinite routing loops. (If a Dispatcher machine receives a packet from another Dispatcher, it will not forward it to a third Dispatcher.) Wide area supports only one level of remotes.
5. Wide area supports UDP and TCP.
6. Wide area works along with high availability: Each Dispatcher may be backed up by an adjacent standby machine (on the same LAN segment).
7. The Manager and Advisors work with wide area, and, if used, should be started on all participating Dispatcher machines.
8. Network Dispatcher supports WAN only on like operating systems.

ISS

You can use the Dispatcher along with ISS to do load balancing across a wide area network. See Figure 28 on page 136.

Advisors

Advisors are agents within the Dispatcher and CBR components. Their purpose is to assess the health and loading of server machines. They do this with a proactive client-like exchange with the servers. Advisors can be considered as lightweight clients of the application servers.

The product provides several protocol-specific advisors for the most popular protocols. It does not make sense to use some of the provided advisors with

the CBR component (such as: Telnet). Since version 2.0, the product has supported the concept of a “custom advisor” that allows users to write their own advisors.

How advisors work

Advisors periodically open a TCP connection with each server and send a request message to the server. The content of the message is specific to the protocol running on the server. For example, the HTTP advisor sends a “HEAD” request to the server.

Advisors then listen for a response from the server. After getting the response, the advisor makes an assessment of the server. To calculate this “load” value, most advisors measure the time for the server to respond, and then use this value (in milliseconds) as the load.

Advisors then report the load value to the manager function, where it appears in the manager report in the “Port” column. The manager then calculates aggregate weight values from all its sources, per its proportions, and sets these weight values into the executor function. The Executor will then use these weights for load balancing new incoming client connections.

If the advisor determines that a server is alive and well, it will report a positive, non-zero load number to the Manager. If the advisor determines that a server is not active, it will return a special load value of negative one (-1). The Manager and the Executor will not forward any further connections to that server.

List of advisors

- The HTTP advisor opens a connection, sends a HEAD request, waits for a response connection, and returns the elapsed time as a load.
- The FTP advisor opens a connection, sends a SYST request, waits for a response, closes the connection, and returns the elapsed time as a load.
- The Telnet advisor opens a connection, waits for an initial message from the server, closes the connection, and returns the elapsed time as a load.
- The NNTP advisor opens a connection, waits for an initial message from the server, sends a quit command, closes the connection, and returns the elapsed time as a load.
- The IMAP advisor opens a connection, waits for an initial message from the server, sends a quit command, closes the connection, and returns the elapsed time as a load.
- The POP3 advisor opens a connection, waits for an initial message from the server, sends a quit command, closes the connection, and returns the elapsed time as a load.

- The SMTP advisor opens a connection, waits for an initial message from the server, sends a quit, closes the connection, and returns the elapsed time as a load.
- The SSL advisor opens a connection, sends a CLIENT HELLO request, waits for a response, closes the connection, and returns the elapsed time as a load.

Note: The SSL advisor has no dependency upon key management or certificates.

- The WTE advisor opens a connection, sends a WTE specific HTTP GET request, and interprets the response as a WTE load.

Note: When using WTE advisor, WTE needs to be running on all servers being load balanced. The machine on which the IBM Network Dispatcher resides does not need to have WTE installed unless it is on the same machine it is load balancing.

- The connect advisor does not exchange any protocol-specific data with the server. It simply measures the time it takes to open and close a TCP connection with the server. This advisor is useful for server applications which use TCP, but with a higher-level protocol for which an IBM-supplied or custom advisor is not available.
- The ping advisor does not open a TCP connection with the servers, but instead reports whether the server responds to a ping. While the ping advisor may be used on any port, it was designed for configurations using the wildcard port, over which multiple protocol traffic may be flowing. It is also useful for configurations using non-TCP protocols with their servers, such as UDP.
- The reach advisor pings its target machines. This advisor was designed for the Dispatcher's high availability components to determine reachability of its "reach targets." Its results flow to high availability component and do not appear in the manager report. The reach advisor is started automatically with the Manager function.
- The WAS (WebSphere Application Server) advisor works in conjunction with the WAS servers. Customizable sample files for the WAS advisor are provided in the install directory. For more information, see "WebSphere Application Server (WAS) advisor" on page 103.
- The WLM (Workload Manager) advisor is designed to work in conjunction with servers on OS/390 mainframes running the MVS Workload Manager (WLM) component. For more information, see "Workload Manager advisor" on page 105.
- Dispatcher provides the ability for a customer to write a custom (customizable) advisor. This enables support for proprietary protocols (on top of TCP) for which IBM has not developed a specific advisor. For more information, see "Create custom (customizable) advisors" on page 102.

Create custom (customizable) advisors

The custom (customizable) advisor is a small piece of Java code, which you provide as a class file, that gets called by the base code. The base code provides all administrative services, such as starting and stopping an instance of the custom advisor, providing status and reports, and recording history information in a log file. It also reports results to the manager component. Periodically the base code will perform an advisor cycle, where it individually evaluates all servers in its configuration. It starts by opening a connection with a server machine. If the socket opens, the base code will call the “getLoad” method (function) in the custom advisor. The custom advisor then performs whatever steps are necessary to evaluate the health of the server. Typically, it will send a user-defined message to the server and then wait for a response. (Access to the open socket is provided to the custom advisor.) The base code then closes the socket with the server and reports the load information to the Manager.

The base code and custom advisor can operate in either normal or replace mode. Choice of the mode of operation is specified in the custom advisor file as a parameter in the constructor method.

In normal mode, the custom advisor exchanges data with the server, and the base advisor code times the exchange and calculates the load value. The base code then reports this load value to the manager. The custom advisor needs only return a zero (on success) or negative one (on error). To specify normal mode, the replace flag in the constructor is set to false.

In replace mode, the base code does not perform any timing measurements. The custom advisor code performs whatever operations are desired for its unique requirements, and then returns an actual load number. The base code will accept the number and report it to the manager. For best results, normalize your load number between 10 and 1000, with 10 representing a fast server, and 1000 representing a slow server. To specify replace mode, the replace flag in the constructor is set to true.

With this feature, you can write your own advisors that will provide the precise information about servers that you need. A sample custom advisor is provided with the Network Dispatcher product. After installing Dispatcher or CBR components, you may find the sample code in `/samples/CustomAdvisors/ADV_sample.java`.

The default install directories are:

- AIX: `/usr/lpp/nd/dispatcher/`, `/usr/lpp/nd/cbr/`
- Red Hat Linux: `/opt/nd/dispatcher/`, `/opt/nd/cbr/`
- Sun: `/opt/nd/dispatcher/`, `/opt/nd/cbr/`

- Windows: c:\Program Files\IBM\nd\dispatcher\,c:\Program Files\IBM\nd\cbr\

WebSphere Application Server (WAS) advisor

Sample custom advisor files specifically for the WebSphere Application Server (WAS) advisor are also provided in the install directory for both the Dispatcher and the CBR components.

- ADV_was.java is the file to be compiled and run on the Network Dispatcher machine
- NDAdvisor.java.servlet (to be renamed NDAdvisor.java) is the file to be compiled and run on the WAS machine.

The WAS advisor sample files reside in the same samples directory as the ADV_sample.java file.

Naming Convention

Your custom advisor file name must be in the form “ADV_myadvisor.java.” It must start with the prefix “ ADV_” in uppercase. All subsequent characters must be in lowercase letters.

As per Java conventions, the name of the class defined within the file must match the name of the file. If you copy the sample code, be sure to change all instances of “ADV_sample” inside the file to your new class name.

Compilation

Custom advisors are written in Java language. You must obtain a Java compiler for your machine, and be able to run its compiler, javac. These files are referenced during compilation:

- the custom advisor file
- the base classes file, lib/ibmnd.jar, found in the Dispatcher or CBR install directory
- the java classes file, lib/classes.zip, found in your java install directory

Your classpath must point to all three of these files during the compile.

For Windows, a compile command might look like this:

```
c:\temp>c:\jdk1.1.8\bin\javac
-classpath .;ibmnd.jar;c:\jdk1.1.8\lib\classes.zip
ADV_fred.java
```

where:

- Your advisor file is named ADV_fred.java
- Your advisor file is stored in the current directory
- You installed java in c:\jdk1.1.8
- The classpath dot “.” points to the current directory

- You have copied `ibmnd.jar` from Dispatcher to the current directory

The output for the compilation is a class file, for example

`ADV_fred.class`

Note: If you wish, custom advisors may be compiled on one operating system and run on another. For example, you may compile your advisor on Windows, copy the class file (in binary) to an AIX machine, and run the custom advisor there.

For AIX, Red Hat Linux, and Sun, the syntax is similar.

Run

To run the custom advisor, you must first copy the class file to the proper subdirectory of the Dispatcher or CBR component:

`samples/CustomAdvisors/ADV_fred.class`

Configure the component, start its manager function, and issue the command to start your custom advisor:

```
ndcontrol advisor start fred 123
```

or

```
cbrcontrol advisor start fred 123
```

where:

- `fred` is the name of your advisor, as in `ADV_fred.java`
- `123` is the port on which your advisor will operate

Required routines

Like all advisors, a custom advisor extends the function of the advisor base, called `ADV_Base`. It is the advisor base that actually performs most of the advisor's functions, such as reporting loads back to the manager for use in the manager's weight algorithm. The advisor base also performs socket connect and close operations and provides send and receive methods for use by the advisor. The advisor itself is used only for sending and receiving data to and from the port on the server being advised. The TCP methods within the advisor base are timed to calculate the load. A flag within the constructor in the `ADV_base` overwrites the existing load with the new load returned from the advisor if desired.

Note: Based on a value set in the constructor, the advisor base supplies the load to the weight algorithm at specified intervals. If the actual advisor has not completed so that it can return a valid load, the advisor base uses the previous load.

These are base class methods:

- A **constructor** routine. The constructor calls the base class constructor (see the sample advisor file)
- An **ADV_AdvisorInitialize** method. This method provides a hook in case additional steps need to be taken after the base class completes its initialization.
- A **getload** routine. The base advisor class performs the open socket; therefore getload needs only to issue the appropriate send and receive requests to complete the advise cycle.

Search order

Dispatcher and CBR first look at the list of native advisors that it provides. If it does not find a given advisor there, Dispatcher and CBR then look at the customer's list of customized advisors.

Naming and path

- The custom advisor class must be located within the subdirectory of `samples\CustomAdvisors` in the Dispatcher and CBR base directories. The defaults for this directory vary by operating system:
 - AIX
`/usr/lpp/nd/dispatcher/samples/, /usr/lpp/nd/cbr/samples`
 - Red Hat Linux
`/opt/nd/dispatcher/samples/, /opt/nd/cbr/samples`
 - Solaris
`/opt/nd/dispatcher/samples/, /opt/nd/cbr/samples`
 - Windows
`C:\Program Files\IBM\nd\dispatcher\samples\, C:\Program Files\IBM\nd\cbr\samples`
- Only lowercase, alphabetic characters are permitted. This eliminates case sensitivity when an operator types in commands on the command line. The advisor name must be prefixed with **ADV_**.

Sample advisor

The program listing for a sample advisor is included in “Sample advisor” on page 258. After installation, this sample advisor can be found in the `samples\CustomAdvisors` directory.

Workload Manager advisor

This feature is only available for the Dispatcher component.

WLM is code that runs on MVS mainframes. It can be queried to ask about the load on the MVS machine.

When MVS Workload Management has been configured on your OS/390 system, Dispatcher can accept capacity information from WLM and use it in the load balancing process. Using the WLM advisor, Dispatcher will periodically open connections through the WLM port on each server in the Dispatcher host table and accept the capacity integers returned. Since these integers represent the amount of capacity that is still available and Dispatcher expects values representing the loads on each machine, the capacity integers are inverted by the advisor and normalized into load values (i.e. a large capacity integer but a small load value both represent a healthier server). The resulting loads are placed into the System column of the manager report.

There are several important differences between the WLM advisor and other Dispatcher advisors:

1. Other advisors open connections to the servers using the same port on which flows normal client traffic. The WLM advisor opens connections to the servers using a port different from normal traffic. The WLM agent on each server machine must be configured to listen on the same port on which the Dispatcher WLM Advisor is started. The default WLM port is 10007.
2. Other advisors only assess those servers defined in the Dispatcher cluster:port:server configuration for which the server's port matches the advisor's port. The WLM advisor advises upon every server in the Dispatcher cluster:port:server configuration. Therefore you must not define any non-WLM servers when using the WLM advisor.
3. Other advisors place their load information into the manager report under its "Port" column. The WLM advisor places its load information into the manager report under its system column.
4. It is possible to use both protocol-specific advisors along with the WLM advisor. The protocol-specific advisors will poll the servers on their normal traffic ports, and the WLM advisor will poll the system load using the WLM port.

ISS and SMA Restriction

Like ISS and SMA (for Red Hat Linux), the WLM agent reports on server systems as a whole, rather than on individual protocol-specific server daemons. ISS, SMA, and WLM place their results into the system column of the manager report. As a consequence, running both the WLM advisor and ISS, or running both the WLM advisor and SMA at the same time is not supported.

Server Monitor Agent (SMA)

This feature is only available for the Dispatcher component.

For **Red Hat Linux** only, the Server Monitor Agent (SMA) replaces the ISS system monitor. SMA provides server load information to the Dispatcher in the form of system-specific metrics, reporting on the health of the servers. The ND manager queries the Server Monitor Agent residing on each of the servers, assigning weights to the load balancing process using the metrics gathered from the agents. The results are placed into the manager report. Unlike ISS, SMA provides no DNS-based load balancing capabilities.

For a configuration example see Figure 8 on page 38.

WLM Restriction

Like the WLM advisor, the SMA agent reports on server systems as a whole, rather than on individual protocol-specific server daemons. Both WLM and SMA place their results into the system column of the manager report. As a consequence, running both the WLM advisor and SMA at the same time is not supported.

Prerequisites

The Server Monitor Agent can only be installed on servers running on the Red Hat Linux platform. Therefore, in order for Dispatcher to load balance correctly with SMA, all the servers in the configuration must have the agent installed and must be running on a Red Hat Linux platform.

How to Use Server Monitor Agent (SMA)

- ND Manager (Dispatcher side)
 - Start `ndserver`.
 - Issue command: **`ndcontrol manager start manager.log port`**
port is the RMI port chosen for all agents to run on.
 - Issue command: **`ndcontrol manager systemscript scriptname`**
scriptname is the name of the script (stored on the server) which should run on each of the servers in the configuration. Two scripts are provided for the customer - `cpuload` and `memload`. The script contains a command which should return a numeric value in the range of 0-100. For more information on `systemscript` command, see “`ndcontrol manager — control the manager`” on page 213.
 - Set the manager proportions so that the fourth proportion, for system, is set to a nonzero value.
 - Add only servers to the configuration which contain a server monitor agent running on the port specified in the manager start command.

Note: Ensure Security —

- On the Network Dispatcher machine, create a key file for the component that is running (using **`ndkeys create`** or **`cbrkeys create`** command). See “Remote Authenticated Administration” on page 161, for more information on `ndkeys`.

- On the server machine, copy the resulting key file to the `/opt/nd/sma/key` directory. Verify that the key file's permissions enable the file to be readable by the root.
- SMA (Server side)
 - Install the SMA package from the IBM Network Dispatcher install. (This package is unique for the Red Hat Linux platform.)
 - Check the **start** script in `/usr/bin/servermonitor` directory to verify that the desired RMI port is being used.
 - Optionally, customers can write their own customized script files which define the command that the SMA will issue on the server machines. Ensure that any custom scripts are executable and located in the `opt/nd/sma/scripts` directory. Custom scripts **must** return a numeric value in the range of 0-100.

The following two scripts are already provided for the customer: **cpuload** (returns the percentage of cpu in use ranging from 0-100) and **memload** (returns the percentage of memory in use ranging from 0-100). These scripts reside in the previously stated directory.

 - Start the agent by issuing the **servermonitor** command.
 - To stop the Server Monitor Agent, issue the **kill** command and pass it the process number of the SMA.

Configure rules-based load balancing

You can use rules-based load balancing to fine tune when and why packets are sent to which servers. The Dispatcher and CBR components review any rules you add from first priority to last priority, stopping on the first rule that it finds to be true, then load balancing the content between any servers associated with the rule. It already balances the load based on destination and port, but using rules expands your ability to distribute connections.

You should use rules-based load balancing with the Dispatcher component when you want to use a subset of your servers for some reason. You must always use rules with the CBR component. You can use the following types of rules:

- Client IP address
- Time of day
- Connections per second for a port
- Active connections total for a port
- Client port
- Always true
- Type of service (TOS)
- Content of a request

We recommend that you make a plan of the logic that you want the rules to follow before you start adding rules to your configuration.

How are rules evaluated?

All rules have a name, type, priority, begin range, and end range, and may have a set of servers. In addition, the content type rule in the CBR component has a matching regular expression pattern associated with it. Rules are evaluated in priority order, with lower priority rules evaluated first. In other words, a rule with a priority of 1 will be evaluated before a rule with a priority of 2. The first rule that is satisfied will be used. Once a rule has been satisfied, no further rules are evaluated.

For a rule to be satisfied, it must meet two conditions:

1. The predicate of the rule must be true. That is, the value it is evaluating must be between the begin and end ranges, or the content must match the regular expression specified in the content rule's pattern. For rules of type "true," the predicate is always satisfied, regardless of the begin and end ranges.
2. If there are servers associated with the rule, at least one of them must be available to forward packets to.

If a rule has no servers associated with it, the rule only needs to meet condition one to be satisfied. In this case, Dispatcher will drop the connection request, and CBR will cause WTE to return an error page. If no rules are satisfied, Dispatcher will select a server from the full set of servers available on the port, and CBR will cause WTE to return an error page.

Using rules based on the client IP address

You may want to use rules based on the client IP address if you want to screen the customers and allocate resources based on where they are coming from.

For example, you have noticed that your network is getting a lot of unpaid and therefore unwanted traffic from clients coming from a specific set of IP addresses. You create a rule using the **ndcontrol rule** command, for example:

```
ndcontrol rule add 9.67.131.153:80:ni type ip beginrange 9.0.0.0 endrange 9.255.255.255
```

This rule would screen out any connection from IBM clients. You would then add to the rule the servers which you wanted accessible to IBMers, or if you do not add any servers to the rule, requests coming from 9.x.x.x addresses would not be served by any of your servers.

Using rules based on the time of day

You may want to use rules based on the time of day for capacity planning reasons. For example, if your Web site gets hit most during the same group of

hours every day, you might want to dedicate five servers to HTTP during full-time, then adding another five during the peak time period.

Another reason you might use a rule based on the time of day is when you want to take some of the servers down for maintenance every night at midnight, so you would set up a rule that excludes those servers during the necessary maintenance period.

Using rules based on the connections per second on a port

Note: The manager must be running for the following to work.

You may want to use rules based on connections per second on a port if you need to share some of your servers with other applications. For example, you set two rules:

1. If connections per second on port 80 > 100 then use these 2 servers
2. If connections per second on port 80 > 2000 then use these 10 servers

Or you might be using Telnet and want to reserve two of your five servers for Telnet, except when the connections per second increase above a certain level. This way, Dispatcher would balance the load across all five servers at peak times.

Using rules based on the active connections total on a port

Note: The manager must be running for the following to work.

You may want to use rules based on active connections total on a port if your servers get overloaded and start throwing packets away. Certain Web servers will continue to accept connections even though they do not have enough threads to respond to the request. As a result, the client requests time out and the customer coming to your Web site is not served. You can use rules based on active connections to balance capacity within a pool of servers.

For example, you know from experience that your servers will stop serving after they have accepted 250 connections. You can create a rule using the **ndcontrol rule** command or the **cbrcontrol rule** command, for example:

```
ndcontrol rule add 130.40.52.153:80:pool2 type active  
beginrange 250 endrange 500
```

or

```
cbrcontrol rule add 130.40.52.153.80:pool2 type active  
beginrange 250 endrange 500
```

You would then add to the rule your current servers plus some additional servers, which will otherwise be used for other processing.

Using rules based on the client port

This rule type is only available in the Dispatcher component.

You may want to use rules based on the client port if your clients are using some kind of software that asks for a specific port from TCP/IP when making requests.

For example, you could create a rule that says that any request with a client port of 10002 will get to use a set of special fast servers because you know that any client request with that port is coming from an elite group of customers.

Using rules that are always true

A rule may be created that is “always true.” Such a rule will always be selected, unless all the servers associated with it are down. For this reason, it should ordinarily be at a lower priority than other rules.

You can even have multiple “always true” rules, each with a set of servers associated with it. The first true rule with an available server is chosen. For example, assume you have six servers. You want two of them to handle your traffic under all circumstances, unless they are both down. If the first two servers are down, you want a second set of servers to handle the traffic. If all four of these servers are down, then you will use the final two servers to handle the traffic. You could set up three “always true” rules. Then the first set of servers will always be chosen as long as at least one is up. If they are both down, one from the second set will be chosen, and so forth.

As another example, you may want an “always true” rule to ensure that if incoming clients do not match any of the rules you have set, they will not be served. You would create a rule using the **ndcontrol rule** command like:

```
ndcontrol rule add 130.40.52.153:80:jamais type true priority 100
```

Then you would not add any servers to the rule, causing the clients packets to be dropped with no response.

Note: You do not need to set a beginrange (b) or endrange (e) when creating an always true rule.

You can define more than one “always true” rule, and thereafter adjust which one gets executed by changing their priority levels.

Using rules based on type of service (TOS)

This rule type is only available in the Dispatcher component.

You may want to use rules based on the content of the “type of service” (TOS) field in the IP header. For example, if a client request comes in with one TOS value that indicates normal service, it can be routed to one set of servers. If a different client request comes in with a different TOS value that indicates a higher priority of service, it can be routed to a different set of servers.

The TOS rule allows you to fully configure each bit in the TOS byte using the **ndcontrol rule** command. For significant bits that you want matched in the TOS byte, use 0 or 1. Otherwise, the value x is used. The following is an example for adding a TOS rule:

```
ndcontrol rule add 9.67.131.153:80:tsr type service tos 0xx1010x
```

Using rules based on the request content

This rule type is only available in the CBR component.

You will want to use content type rules to send requests to sets of servers specifically set up to handle some subset of your site’s traffic. For example, you may want to use one set of servers to handle all *cgi-bin* requests, another set to handle all streaming audio requests, and a third set to handle all other requests. You would add one rule with a pattern that matches the path to your *cgi-bin* directory, another that matches the file type of your streaming audio files, and a third always true rule to handle the rest of the traffic. You would then add the appropriate servers to each of the rules. For further information about configuring content type rules, see “Chapter 7. Configuring the Content Based Routing component” on page 73.

Adding rules to your configuration

You can add rules using the **ndcontrol** or **cbrcontrol rule add** command, by editing the sample configuration file, or with the graphical user interface (GUI). You can add one or more rules to every port you have defined.

It is a two-step process: add the rule, then define which servers to serve to if the rule is true. For example, our system administrator wanted to track how much use the proxy servers were getting from each division on site. She knows which IP addresses are given to each division. She would create the first set of rules based on client IP address to separate each division’s load:

```
ndcontrol rule add 130.40.52.153:80:div1 type ip b 9.1.0.0 e 9.1.255.255
ndcontrol rule add 130.40.52.153:80:div2 type ip b 9.2.0.0 e 9.2.255.255
ndcontrol rule add 130.40.52.153:80:div3 type ip b 9.3.0.0 e 9.3.255.255
```

Next, she would add a different server to each rule, then measure the load on each of the servers in order to bill the division properly to the services they are using. For example:

```
ndcontrol rule useserver 130.40.52.153:80:div1 207.72.33.45
ndcontrol rule useserver 130.40.52.153:80:div2 207.72.33.63
ndcontrol rule useserver 130.40.52.153:80:div3 207.72.33.47
```

Using explicit linking

In general, the load-balancing functions of the Dispatcher work independently of the content of the sites on which the product is used. There is one area, however, where site content can be important, and where decisions made regarding content can have a significant impact upon the Dispatcher's efficiency. This is in the area of link addressing.

If your pages specify links that point to individual servers for your site, you are in effect forcing a client to go to a specific machine, thus by-passing any load balancing function that might otherwise be in effect. For this reason, it is recommended that you always use the address of Dispatcher in any links contained in your pages. Note that the kind of addressing used may not always be apparent, if your site uses automated programming that dynamically creates HTML. To maximize your load-balancing, you should be aware of any explicit addressing and avoid it where possible.

Using a private network configuration

You can set up Dispatcher and the TCP server machines using a private network. This configuration can reduce the contention on the public or external network that can affect performance.

For AIX, this configuration can also take advantage of the fast speeds of the SP High Performance Switch if you are running Dispatcher and the TCP server machines on nodes in an SP Frame.

To create a private network, each machine must have at least two LAN cards, with one of the cards connected to the private network. You must also configure the second LAN card on a different subnet. The Dispatcher machine will then send the client requests to the TCP server machines through the private network.

Windows Only: Execute the following command:

```
ndconfig en1 10.0.0.x netmask 255.255.255.0
```

Where `en1` is the name of the second interface card in the Dispatcher machine, `10.0.0.x` is the network address of the second interface card, and `255.255.255.0` is the netmask of the private network.

The servers added using the **ndcontrol server add** command must be added using the private network addresses; for example, referring to the Apple server example in Figure 25 on page 114, the command should be coded as:

```
ndcontrol server add cluster_address :80:10.0.0.1,
```

not

```
ndcontrol server add cluster_address :80:9.67.131.18
```

If you are using ISS to provide load information to Dispatcher, you must configure ISS to report loads on the private addresses.

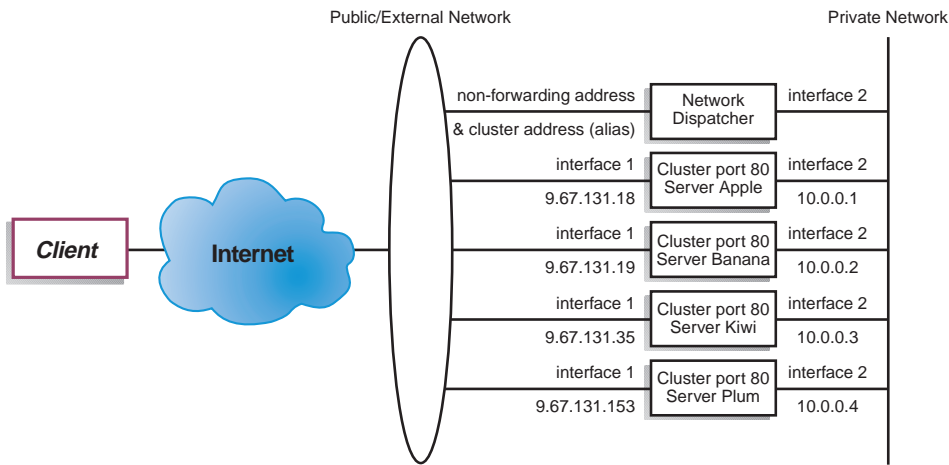


Figure 25. Example of a private network using Dispatcher

Using a private network configuration only applies to the Dispatcher component.

Use wildcard cluster to combine server configurations

The “wildcard” refers to the cluster’s ability to match multiple IP addresses (i.e. acts as a wildcard). Cluster address 0.0.0.0 is used to specify a wildcard cluster.

If you have many cluster addresses to load-balance, and the port/server configurations are identical for all your clusters, you can combine all the clusters into one star configuration.

You must still explicitly configure each cluster address on one of the network adapters of your Dispatcher workstation. You should not add any of the cluster addresses to the Dispatcher configuration using the ndcontrol cluster add command however.

Add only the wildcard cluster (address 0.0.0.0), and configure the ports and servers as required for load balancing. Any traffic to any of the adapter configured addresses will be load balanced using the wildcard cluster configuration.

An advantage of this approach is that traffic to all the cluster addresses is taken into account when determining the best server to go to. If one cluster is getting a lot of traffic, and it has created many active connections on one of the servers, traffic to other cluster addresses will be load balanced using this information.

You can combine the wildcard cluster with actual clusters if you have some cluster addresses with unique port/server configurations, and some with common configurations. The unique configurations must each be assigned to an actual cluster address. All common configurations can be assigned to the wildcard cluster.

Using wildcard cluster to combine server configurations only applies to the Dispatcher component.

Use wildcard cluster to load balance firewalls

Using wildcard cluster to load balance firewalls only applies to the Dispatcher component. Cluster address 0.0.0.0 is used to specify a wildcard cluster.

The wildcard cluster can be used to load balance traffic to addresses that are not explicitly configured on any network adapter of the Dispatcher workstation. In order for this to work, the Dispatcher must at least be able to see all the traffic it is to load balance. The dispatcher workstation will not see traffic to addresses that have not been explicitly configured on one of its network adapters unless it is set up as the default route for some set of traffic.

Once Dispatcher has been configured as a default route, any TCP or UDP traffic through the Dispatcher machine will be load balanced using the wildcard cluster configuration.

One application of this is to load balance firewalls. Since firewalls can process packets for any destination address and any destination port, you need to be able to load balance traffic independent of the destination address and port.

Firewalls are used to handle traffic from non-secure clients to secure servers, and the responses from the secure servers, as well as traffic from clients on the secure side to servers on the non-secure side, and the responses.

You must set up two Dispatcher machines, one to load balance non-secure traffic to the non-secure firewall addresses and one to load balance secure traffic to the secure firewall addresses. Since both of these Dispatchers must use the wildcard cluster and wildcard port with different sets of server addresses, the two Dispatchers must be on two separate workstations.

Use wildcard cluster with WTE for transparent proxy

Using wildcard cluster with WTE for transparent proxy only applies to the Dispatcher component. Cluster address 0.0.0.0 is used to specify a wildcard cluster.

The wildcard cluster function also allows Dispatcher to be used to enable a transparent proxy function for a Web Traffic Express (WTE) server residing on the same box as Dispatcher. This is an AIX feature only, as there must be communication from the dispatcher component to the TCP component of the operating system.

To enable this feature, you must start WTE listening for client requests on port 80. You then configure a wildcard cluster. In the wildcard cluster, you configure port 80. In port 80, you configure the NFA of the Dispatcher machine as the only server. Now any client traffic to any address on port 80 will be delivered to the WTE server running on the Dispatcher workstation. The client request will then be proxied as usual, and the response will be sent back from WTE to the client. In this mode, the Dispatcher component is not performing any load balancing.

Use wildcard port to direct unconfigured port traffic

The wildcard port can be used to handle traffic that is not for any explicitly configured port. One use of this is for load balancing firewalls. A second use is to ensure that traffic to an unconfigured port is handled appropriately. By defining a wildcard port with no servers, you will guarantee that any request to a port that has not been configured will be discarded rather than delivered back to the operating system. Port number 0 (zero) is used to specify a wildcard port, for example:

```
ndcontrol port add cluster:0
```

How classical ND affinity works

You enable the affinity feature when you configure a cluster's port to be sticky. Configuring a cluster's port to be sticky allows subsequent client requests to be directed to the same server. This is done by setting "port stickytime" to some number of seconds. The feature is disabled by setting stickytime to zero.

Interaction with cross port affinity: If you are enabling cross port affinity, stickytime values of the shared ports must be the same (nonzero) value. See "Cross port affinity" on page 118, for more information.

Behavior when disable affinity

With the feature disabled, whenever a new TCP connection is received from a client, Dispatcher picks the right server at that moment in time and forwards the packets to it. If a subsequent connection comes in from the same client, Dispatcher treats it as an unrelated new connection, and again picks the right server at that moment in time.

Behavior when enable affinity

With the feature enabled, if a subsequent request is received from the same client, the request is directed to the same server.

Over time, the client will finish sending transactions, and the affinity record will go away. Hence the meaning of the sticky "time." Each affinity record lives for the "stickytime" in seconds. When subsequent connections are received within the stickytime, the affinity record is still valid and the request will go to the same server. If a subsequent connection is not received within stickytime, the record is purged; a connection that is received after that time will have a new server selected for it.

Server Directed Affinity API to control client-server affinity

The Server Directed Affinity API only applies to the Dispatcher component.

The SDA feature provides an API that allows an external agent to influence the Dispatcher affinity behavior.

SDA Features

Your application may have indicated that their server systems have the knowledge to direct client requests to particular server machines better than Dispatcher can. Rather than having a client be "directed" to the same server as selected by Dispatcher's load balancing selection, you may want the client to be "directed" to the server of your choosing. The SDA feature provides this API. You can now write your own software to implement an SDA agent, which communicates with a listener in Dispatcher. It can then manipulate the Dispatcher affinity tables to:

- query the contents
- insert new records
- remove records

Records inserted in an affinity table by an SDA agent remain in the table indefinitely. They do not timeout. They are removed only when the SDA agent removes them or if a Dispatcher advisor detects that the server is dead.

Dispatcher's SDA components

Dispatcher implements a new socket listener to accept and handle requests from an SDA agent. When an SDA agent opens a connection with Dispatcher, the listener will accept it and leave the connection open. Multiple requests and responses can flow over this persistent connection. The socket will close when the SDA agent closes it or if Dispatcher detects an unrecoverable error. Inside Dispatcher, the listener takes each request from the SDA agent, communicates with the appropriate affinity table in the Dispatcher executor kernel, and prepares a response for the SDA agent.

For more information, refer to the following files after the Dispatcher has been installed. The files appear in the Dispatcher's install directory tree.

The files are:

- the API: `samples/SDA/SDA_API.htm`
- the sample code for an SDA agent: `samples/SDA/SDA_SampleAgent.java`.

Cross port affinity

Cross port affinity only applies to the Dispatcher component.

Cross port affinity is the sticky feature that has been expanded to cover multiple ports. For example, if a client request is first received on one port and the next request is received on another port, cross port affinity allows the dispatcher to send the client request to the same server. In order to use this feature, the ports must:

- share the same cluster address
- share the same servers
- have the same (nonzero) **stickytime** value
- have the same **stickymask** value

More than one port can link to the same **crossport**. When subsequent connections come in from the same client on the same port or a shared port, the same server will be accessed. The following is an example of configuring multiple ports with a cross port affinity to port 10:

```
ndcontrol port set cluster:20 crossport 10
ndcontrol port set cluster:30 crossport 10
ndcontrol port set cluster:40 crossport 10
```

After cross port affinity has been established, you have the flexibility to modify the stickytime value for the port. However, it is recommended that you change the stickytime values for all shared ports to the same value, otherwise unexpected results may occur.

To remove the cross port affinity, set the crossport value back to its own port number. See “ndcontrol port — configure ports” on page 219, for detailed information on command syntax for the **crossport** option.

Affinity address mask

Affinity address mask only applies to the Dispatcher component.

Affinity address mask is a sticky feature enhancement to group clients based upon common subnet addresses. Specifying **stickymask** on the **ndcontrol port** command allows you to mask the common high-order bits of the 32-bit IP address. If this feature is enabled, when a client request first makes a connection to the port, all subsequent requests from clients with the same subnet address (represented by that part of the address which is being masked) will be directed to the same server.

For example, if you want all incoming client requests with the same network Class A address to be directed to the same server, you set the stickymask value to 8 (bits) for the port. To group client requests with the same network Class B address, set the stickymask value to 16 (bits). To group client requests with the same network Class C address, set the stickymask value to 24 (bits).

For best results, set the stickymask value when first starting the Network Dispatcher. If you change the stickymask value dynamically, results will be unpredictable.

Interaction with cross port affinity: If you are enabling cross port affinity, stickymask values of the shared ports must be the same. See “Cross port affinity” on page 118, for more information.

To enable affinity address mask, issue an **ndcontrol port** command similar to the following:

```
ndcontrol port set cluster:port stickymask 8
```

Possible stickymask values are 8, 16, 24 and 32. A value of 8 specifies the first 8 high-order bits of the IP address (network Class A address) will be masked. A value of 16 specifies the first 16 high-order bits of the IP address (network Class B address) will be masked. A value of 24 specifies the first 24 high-order bits of the IP address (network Class C address) will be masked. If you specify a value of 32, you are masking the entire IP address which effectively disables the affinity address mask feature. The default value of stickymask is 32.

See “**ndcontrol port — configure ports**” on page 219, for detailed information on command syntax for stickymask (affinity address mask feature).

Rule affinity override

With rule affinity override, you can override the stickiness of a port for a specific server. For example, you are using a rule to limit the amount of connections to each application server, and you have an overflow server with an always true rule that says “please try again later” for that application. The

port has a stickytime value of 25 minutes, so you don't want the client to be sticky to that server. With rule affinity override, you can change the overflow server to override the affinity normally associated with that port. The next time the client requests the cluster, it is load balanced to the best available application server, not the overflow server.

See “ndcontrol server — configure servers” on page 228, for detailed information on command syntax for the rule affinity override, using the server **sticky** option.

Cookie affinity

The cookie affinity feature applies only to CBR with WTE (for HTTP), which supports load balancing based on rules. It provides a new way to make clients “sticky” to a particular server. This function is enabled by setting the stickytime of a rule to a positive number, and setting the affinity to “cookie.” This can be done when the rule is added, or using the rule set command. See “ndcontrol rule — configure rules” on page 223, for detailed information on command syntax.

Once a rule has been enabled for cookie affinity, new client requests will be load-balanced using standard CBR algorithms, while succeeding requests from the same client will be sent to the initially chosen server. The chosen server is stored as a cookie in the response to the client. As long as the client's future requests contains the cookie, and each request arrives within the stickytime interval, the client will maintain affinity with the initial server.

Cookie affinity is used to ensure that a client continues to be load balanced to the same server for some period of time. This is accomplished by sending a cookie to be stored by the clients browser. The cookie contains the cluster:port:rule that was used to make the decision, the server that was load balanced to, and a timeout timestamp for when the affinity is no longer valid. Whenever a rule fires that has cookie affinity turned on, the cookies sent by the client are examined. If a cookie is found that contains the identifier for the cluster:port:rule that fired, then the server load balanced to, and the expires timestamp are extracted from the cookie. If the server is still in the set used by the rule, and its weight is greater than zero, and the expires timestamp is greater than now, then the server in the cookie is chosen to load balance to. If any of the preceding three conditions are not met, a server is chosen using the normal algorithm. Once a server has been chosen (using either of the two methods) a new cookie is constructed, with the five pieces of information. (IBM CBR cluster:port:rule=timestamp), the timestamp will be the time that affinity expires. The “cluster:port:rule, and the server_chosen” are encoded so that no information about the CBR configuration is revealed. An “expires” parameter is also inserted in the cookie. This parameter is in a format the

browser can understand, and causes the cookie to become invalid two hours after the expires timestamp. This is so the client's cookie database isn't cluttered up.

This new cookie is then inserted in the headers that go back to the client, and if the client's browser is configured to accept cookies, it will send back subsequent requests.

How to enable cookie affinity

To enable cookie affinity for a particular rule, use the "rule set" command:

```
rule set cluster:port:rule stickytime 60
rule set cluster:port:rule affinity cookie
```

Stickiness is set per rule. Two different rules can send the same client to two different servers. Within one rule the client will be sent to one server, for another rule the client could be sent to a different server. This is a requirement, and for CBR, the servers for rules will be different, so a server using a rule will probably not even exist in another rule. This is desirable because you can make your rule for handling cgi or servlet requests (for example) sticky. Your rule for normal http requests will use normal load balancing.

Why use cookie affinity

Making a rule sticky would normally be used for CGI or servlets that store client state on the server. The state is identified by a cookie ID (these are server cookies). Client state is only on the selected server, so the client needs the cookie from that server to maintain that state between requests.

Normal client IP affinity

Normal client IP affinity for CBR is set by rule. To enable normal affinity use:

```
rule set cluster:port:rule stickytime 300
rule set cluster:port:rule affinity clientip
```

The default for rule affinity is client IP, so if you have never set it to "cookie" you do not need to set it to "clientip."

Using binary logging to analyze server statistics

The binary logging feature allows server information to be stored in binary files. These files can then be processed to analyze the server information that has been gathered over time.

The following information is stored in the binary log for each server defined in the configuration.

- cluster address
- port number

- server address
- server weight
- server total connections
- server active connections
- server port load
- server system load

Some of this information is retrieved from the executor as part of the manager cycle. Therefore the manager must be running in order for the information to be logged to the binary logs.

Use **ndcontrol log** command set to configure binary logging.

- log start
- log stop
- log set interval <second>
- log set retention <hours>
- log status

The start option starts logging server information to binary logs in the logs directory. One log is created at the start of every hour with the date and time as the name of the file.

The stop option stops logging server information to the binary logs. The log service is stopped by default.

The set interval option controls how often information is written to the logs. The manager will send server information to the log server every manager interval. The information will be written to the logs only if the specified log interval seconds have elapsed since the last record was written to the log. By default, the log interval is set to 60 seconds. There is some interaction between the settings of the manger interval and the log interval. Since the log server will be provided with information no faster than manager interval seconds setting the log interval less than the manager interval effectively sets it to the same as the manager interval. This logging technique allows you to capture server information at any granularity. You can capture all changes to server information that are seen by the manager for calculating server weights. However, this amount of information is probably not required to analyze server usage and trends. Logging server information every 60 seconds gives you snapshots of server information over time. Setting the log interval very low can generate huge amounts of data.

The set retention option controls how long log files are kept. Log files older than the retention hours specified will be deleted by the log server. This will

only occur if the log server is being called by the manager, so stopping the manager will cause old log files not to be deleted.

The status option returns the current settings of the log service. These settings are whether the service is started, what the interval is, and what the retention hours are.

A sample Java program and command file have been provided in the `nd/dispatcher/samples/BinaryLog` and `nd/cbr/samples/BinaryLog` directories. This sample shows how to retrieve all the information from the log files and print it to the screen. It can be customized to do any type of analysis you want with the data. An example using the supplied script and program for the dispatcher would be:

```
ndlogreport 1999/05/01 8:00 1999/05/01 17:00
```

to get a report of Dispatcher server information from 8:00 AM to 5:00 PM on May 1, 1999.

Chapter 9. Planning for the Interactive Session Support component

This chapter describes what the network planner should consider before installing and configuring the Interactive Session Support (ISS) component. This chapter includes the following sections:

- “Hardware and software requirements”
- “Planning Considerations” on page 127
- “Using ISS and Dispatcher in a two-tiered configuration” on page 136
- “Configuring highly available Dispatchers in a two-tier configuration” on page 137
- “Ping Triangulation Considerations” on page 139
- “Using Affinity (StickyTime) with ISSNameServer” on page 140

The ISS component is not available for the Red Hat Linux platform. It is replaced by another system monitor - Server Monitor Agent (SMA). For information on using SMA for Red Hat Linux, see “Server Monitor Agent (SMA)” on page 106.

Hardware and software requirements

Requirements for AIX

- Any IBM RS/6000-based machine
- IBM AIX Version 4.3.2 or higher
- 50 MB of available disk space for installation

Note: Additional disk space will be needed for logs.

- The following Network Interface Cards (NICs) are supported:
 - 16 Mb Token ring
 - 10 Mb Ethernet
 - 100 Mb Ethernet
 - Fiber distributed data interface (FDDI)
- IBM Java development kit (JDK) version 1.1.8 or higher (but not 1.2.x versions)
- Netscape Navigator 4.07 (or higher) or Netscape Communicator 4.61 (or higher) for viewing online Help

Requirements for Solaris

- Any SPARC workstation supported by Solaris version 2.6 and Solaris 7 (32 bit mode)
- 50 MB of available disk space for installation

Note: Additional disk space will be needed for logs.

- The following Network Interface Cards (NICs) are supported:
 - 10 Mb Ethernet
 - 100 Mb Ethernet
- IBM Java runtime environment (JRE) version 1.1.8 or higher (but not 1.2.x versions)
- Sun Microsystems HotJava Browser 1.0.1 (or higher) for viewing online Help

Requirements for Windows 2000 and Windows NT

- Any Intel x86 PC supported by Microsoft Windows 2000; or Microsoft Windows NT, version 4.0 with Service Pack 5 (or higher)
- Windows 2000 Professional, Server, or Advanced Server; or Windows NT Workstation or Server
Interactive Session Support can run on Windows 2000 or NT, Professional/Workstation or Server
- Windows 2000 Server or Windows NT Server is required for machines running ISS NameServer Observers
- 50 MB of available disk space for installation

Note: Additional disk space will be needed for logs.

- The following Network Interface Cards (NICs) are supported:
 - 16 Mb Token ring
 - 10 Mb Ethernet
 - 100 Mb Ethernet
- IBM Java runtime environment (JRE) version 1.1.8 or higher (but not 1.2.x versions)
- Ensure the default browser is either Netscape Navigator 4.07 (or higher), Netscape Communicator 4.61 (or higher), Internet Explorer 4.0 (or higher). The default browser is used for viewing online Help.

Note: If Network Dispatcher is unable to locate the JRE that is installed, then define an environment variable IBMND_JRE_LOCATION which should be set to the fully qualified path to the javai.dll file. An example of the syntax could be:

```
IBMND_JRE_LOCATION="c:\Program Files\ibm\JRE\1.1.8\bin\javai.dll"
```

Planning Considerations

ISS creates an environment in which to function. In this environment, a controlling node:

- communicates with the nodes providing the service
- processes data collected from agents, which is forwarded to observers
- load balances requests from its conclusions

This environment is defined in a configuration file, which you will need to write or modify. For more information, see “Chapter 10. Configuring the Interactive Session Support component” on page 143. The implementation of this environment will determine how effectively requests are load balanced. This section discusses the following planning considerations for the ISS component:

- “Setting up ISS cells”
- “Observers” on page 128
- “Selection methods” on page 130
- “ISS Metrics” on page 132

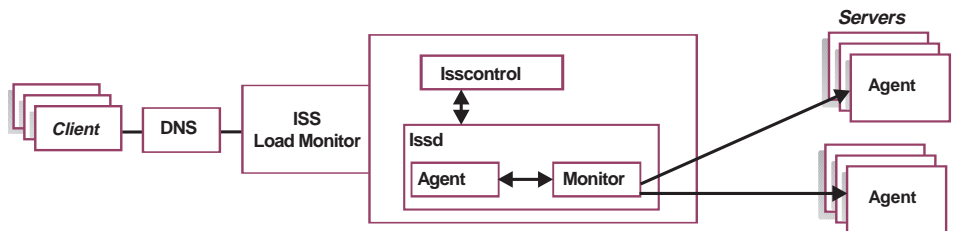


Figure 26. Example of the ISS component

Setting up ISS cells

Each host or machine in your site that will participate in ISS is considered a **node**. Any IP address or DNS name associated with a **node** that will be used must also be defined using the **node** parameters: InternalNet or Interface.

A **cell** is a group of nodes administered as a single logical unit. One of the nodes is elected the leader, or **monitor**, in accordance with its priority level within the cell. All other nodes are then considered to be **agents**. The nodes in a cell must be close enough together, geographically and on the network that the monitor can quickly and reliably communicate with each of them. In general, you will have one cell for each geographical site on your network. There should be a cell name to identify each cell created.

If you plan to use ping triangulation (see “Ping Triangulation Considerations” on page 139) between global sites, you will need to set up multiple cells, one at each site. Each site configuration must contain the local and global cell information.

The load on a node can be determined by measuring various resources such as a CPU usage, disk availability, or process count. Since the same type of resource may be used for several services/applications, we define resources by the **ResourceType** parameter. A ResourceType is further defined using the Metric, MetricNormalization, MetricLimits, and Policy keywords.

In general, a group of servers that are measured by the same **ResourceTypes** are defined as a **service**. A node or interface can be a member of a single service or many services. If you are familiar with the earlier versions of ISS, you can think of a service as similar to a Pool, which was defined as a group of machines performing the same function. Service parameters include NodeList and ResourceList, which tell the service which node(s) and resource(s) to use respectively.

When you set up an individual service for an ISS cell, all the nodes in that service should be equivalent. It is recommended that you create a common data space and common name space. This means that regardless of the server a user is connected to, the user will be able to log in using a consistent user name and password, and will be able to access the appropriate data. You can accomplish this by using a distributed file system such as AFS, DFS, or NFS, or by using a shared disk architecture.

After configuring nodes, services, resourcetypes, and their parameters, you should configure observers. This will determine the implementation of ISS. The concept of the observer is described below.

- In DNS-update and DNS-replace mode: ISS provides a route for incoming clients to get to a node that hosts a particular service.
- As part of the calculations, ISS checks that the node is alive and not too heavily loaded.
- The ISS administrator defines certain resources that must be available before a node can provide a service. These resources are user-defined and the accuracy of the ISS load-balancing function depends on careful choice of the combination of metrics for each service.

Observers

The **observer** type that is defined in the configuration specifies the mode of implementation of ISS. An observer is a load balancer or any other network process that uses information about the status of nodes in the cell. ISS supports three types of observers (implementation mode) that can subscribe to reports on a cell and/or perform load balancing for that cell.

Three implementation modes of ISS are:

- DNS-Update
- DNS-Replace
- DNS-Ignore

An observer (implementation mode) is configured using one of three keywords:

- **Nameserver:**

This corresponds to **DNS-Update** mode. If you have a small local DNS area, then you could potentially use ISS in "DNS-update" mode, in which it will update the main site DNS nameserver. These updates are frequent (possibly as often as every heartbeat) so this configuration is only recommended in a DNS environment that wholly or mainly dedicated to the ISS servers. ISS works in conjunction with the nameserver to map DNS names of ISS services to IP addresses of the servers considered to be the "best" candidate. ISS assists the nameserver by:

- Monitoring the load on each server in the cell.
- Ensuring that the server used for any particular request is the one with the lightest load. Load balancing decisions are based on how you have configured ISS.

- **ISSNameserver:**

This corresponds to **DNS-Replace** mode. ISS functions as a nameserver (it performs nameserver functions) for a limited subset of the network, but does not contain all the functions of a standard DNS. You need to be able to configure a separate subnet for ISS to use with DNS-replace. Entries should be added to the primary site DNS nameserver to redirect requests for services balanced by ISS to the ISS nameservers. ISS then works transparently as part of the normal DNS protocols.

- **Dispatcher:**

This corresponds to **DNS-Ignore** mode. ISS works in conjunction with the Dispatcher to perform the load-balancing function. ISS does not make any load balancing decisions itself. The ISS monitor collects server load information from the ISS agents running on the individual servers and forwards it to the Dispatcher. The Dispatcher uses this load information, along with other sources of information, to perform load balancing.

Each observer must have a **servicelist** specified, which defines to which service the observer subscribes. Every cycle the services are updated and provide the updated results to the subscribed observers. With these updated results, the observer either:

- performs load balancing by updating the DNS name server
- performs load balancing by performing name resolution

- provides the results to Dispatcher for it to load balance

See “Chapter 10. Configuring the Interactive Session Support component” on page 143 for detailed configuration information.

In an advanced configuration, such as a two-tier configuration, **DNS-Ignore** mode can be implemented with **DNS-Update** mode or **DNS-Replace** mode in. For more information, see “Load Balancing Dispatcher with ISS” on page 51.

Selection methods

Another important factor in ISS load balancing is the **SelectionMethod** keyword. Each service maintains a ranking of the best nodes to perform the configured service. You can change the calculation that determines the best node by using the different selection methods. The **SelectionMethod** keyword must be one of the following values:

- **RoundRobin**

Round-robin selection performs no load balancing. Load figures are ignored, although a server will not be recommended if it appears to be down.

- **Best**

For each update period the monitor calculates the highest-ranking server to use during that period. The ranking is re-calculated once every update period. This is the default.

- **Statistical RoundRobin**

For each update period the monitor calculates the load on each server and assigns a proportion of tokens to that server. Each incoming request is assigned a token. If the number of tokens runs out before the next update, then the token levels are reset based on the last available figures.

For both the **Best** and **Statistical Round-Robin** selection methods, you can use two types of metric: Internal and External.

Each service may have a different selection method without affecting any other services. A node may belong to more than one service and each of those services may have different selection methods without affecting any other. Refer to Table 9 on page 131 to determine which configuration method best suits your needs.

Note: For all selection methods, ISS detects if a server fails and does not schedule requests for that server.

Table 9. ISS configuration methods

Select this method:	If these circumstances apply:
Round-Robin	<p>You do not want to measure the actual loads on the servers in the cell, and simply cycling through them will provide a good distribution of load for the types of activity that will be performed on the servers.</p> <p>For more information, see “Round-robin configuration”.</p>
Best	<p>You want to direct requests to the server with the least load.</p> <p>For more information, see “Best Configuration”.</p>
Statistical Round-Robin	<p>You receive extremely frequent requests and you want to balance them in a ratio that is proportional to the server loads</p> <p>For more information, see “Statistical Round-robin configuration”.</p>

Round-robin configuration: With a round-robin configuration, ISS balances the load across a group of servers by assigning requests to the servers in a round-robin fashion. ISS selects each server for a specific length of time, which you specify using the `HeartbeatInterval` and `HeartBeatsPerUpdate` keywords. During this time interval, it will recommend that server. It will not recommend a server that has been identified as having failed. This method ignores the load on each of the servers.

There are advantages and disadvantages to using the round-robin method. The main advantage is that using round-robin avoids the overhead (on the servers) associated with other measurement types. The main disadvantage of round-robin measurement is that ISS does not notice if a particular server is getting very busy. For example, ISS could continue to use a busy server as the server to which new users connect. Round-robin scheduling works well provided the client load has reasonably uniform access frequency and duration.

Best Configuration: Using the “Best” selection method, ISS routes requests during a specific heartbeat interval to the server with the lowest load at the start of that interval. It will not recommend a server that has been identified as having failed. This selection method works well for any duration of the connections, provided the frequency of new connections is low relative to the heartbeat interval.

Statistical Round-robin configuration: ISS gathers load statistics for all the servers and uses them to build up a profile of the most and least heavily loaded servers. Requests within a heartbeat interval are shared between the servers in proportion to their loading. This selection method works well where there is a high frequency of short-lived connections.

It is not appropriate to define any metrics for use with the **RoundRobin** selection method, unless you are using ISS in a two-tier scenario and the loading information is required to be passed on to Dispatcher.

For both **Best** and **Statistical Round-Robin**, you must have a metric specified.

ISS Metrics

To define a ResourceType, ISS provides two methods for collecting load information: internal and external metrics. Internal metrics are supplied by ISS, whereas external metrics are calculated by executing a defined command string on the server that returns a single number result to ISS.

Table 10. ISS Metric types

Select this Metric:	If these circumstances apply:
Internal	ISS supplies a few pre-defined metrics that can be used to measure the load on a server.
External	You may define any command string as a metric to measure server load. This metric should return a single number as the result.

- **Internal Metrics**

There are two Internal metrics that ISS supplies: **CPUload** and **FreeMem**. CPUload measures the percentage of the CPU that is being utilized. CPUload requires the Policy to be Min. FreeMem measures the amount of free physical memory as a percentage. FreeMem requires the Policy to be Max. See "External Metrics" below for an explanation of the **Policy** keyword. Both of these metrics can reasonably be expected to return a value between 0 and 100. This should be defined with the **MetricNormalization** keyword.

Note: FreeMem is not available on the Solaris version of ISS.

- **External Metrics**

With an external metric, you can use any load measurement method that you choose. To create an external metric, you can either write a command string directly into the configuration files or store a command or series of commands in an executable file. Typical files would be a shell script on Unix, or a batch file on Windows. The command or series of commands must return a single numeric result. If you are using multiple operating systems there must be an equivalent command for all the environments used.

For example, you would write the executable filename (including path) in the configuration file as follows:

```
ResourceType UnixScript
Metric External "\etc\script1"
```

or

```
ResourceType NTScript  
Metric External "C:\nd\yourBin\yourScript.bat"
```

The **Policy** parameter in the ISS configuration file determines how the result is interpreted. When you want higher numbers to represent lower loading, set the Policy to **Max**, which indicates that a higher value returned is desirable.

When you want lower numbers to represent lower loading, set the Policy to **Min**, which indicates that a lower value is desirable. The default value is Min. See the description of the Policy keyword in “Configuration file keywords for ISS” on page 238 for more details.

The following example illustrates an external metric that counts the number of processes running on a system. Standard UNIX commands are used to create this metric:

```
ps -ef | wc -l
```

If there were 50 processes running on your system, this external metric would generate a numeric value of 50. The complete coding in the configuration would be something like:

```
ResourceType CPUprocess  
Metric External ps -ef | wc -l  
MetricNormalization 0 100  
MetricLimits 90 95  
Policy Min
```

See “How a configuration file is created” on page 146 for details on MetricLimit.

Another popular external metric measures the percentage of idle CPU (the inverse of CPULoad). Standard UNIX commands are used to create this metric:

```
vmstat 3 2 | tauk -l awk '{print $16}'
```

Note: This example is for AIX. On Solaris, use: `vmstat 3 2 | tail —1 awk '{print $22}'`.

This returns the percentage of idle CPU time during a 3 second sampling period. This value should be maximized when you want to identify the least loaded server. If the CPU is 85% idle, this external metric would generate a numeric value of 85. The complete coding in the configuration would be something like:

```

ResourceType IdleCPU
Metric External vmstat 3 2 | tail -1 | awk '{print $16}'
MetricNormalization 0 100
MetricLimits 60 20
Policy Max

```

Windows Note: Windows does not provide the same commands as UNIX for creating custom metrics. However, tools providing similar results can be obtained from sources on the Internet. Windows users can also use the custom configuration option by writing their own programs, as long as those programs write only a number to standard output.

Multiple ResourceTypes, hence the associated metric, can be listed in the ResourceList of a service. The results returned by the metrics are combined using the metric weight. This allows multiple factors to be considered in the load balancing decision.

Interfaces: an explanation

Imagine three machines, **alien**, **snowy**, and **sunny**: each has several network adapters and multiple IP addresses. Some IP addresses are aliases on a single network adapter, and some are unique to a particular adapter. For simplicity, assume that each machine has only one DNS name, mapping to one of its IP addresses.

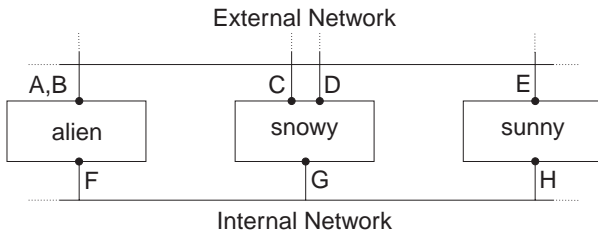


Figure 27. A diagram of an ISS interface

Summary of the network configuration:

alien.foo.com	10.1.1.1	A
	10.1.1.2	B
	10.1.1.3	F
snowy.foo.com	10.1.1.4	C
	10.1.1.5	D
	10.1.1.6	G
sunny.foo.com	10.1.1.7	E
	10.1.1.8	H

Using the three nodes, we can supply several different services, and we may wish to recommend different IP addresses externally depending on which service we are advertising.

For example:

- HTTPService, www.foo.com, will be provided by alien, snowy, and sunny at the addresses 10.1.1.1, 10.1.1.4, and 10.1.1.7
- FTPService, ftp.foo.com, will be provided by alien, snowy, and sunny at the address 10.1.1.2, 10.1.1.5, and 10.1.1.7

We also want to be able to use the extra adapters on each machine for internal communications between the machines (for instances, for ISS to send internal packets on) and so we have:

- Internal communication using 10.1.1.3, 10.1.1.6, and 10.1.1.8

Using ISS with the above scenario, you will need the following lines in your config file:

```
# ISS Config file for Cell ISSCell

Cell          ISSCell          Local

# Nodes

Node          alien.foo.com    1
InternalNet   10.1.1.3
Interface     10.1.1.2

Node          snowy.foo.com   2
InternalNet   10.1.1.6
Interface     10.1.1.5

Node          sunny.foo.com   3
Interface     10.1.1.8

# Information about metrics would be included here

Service       HTTPService    www.foo.com
NodeList      alien          10.1.1.4    sunny

Service       FTPService     ftp.foo.com
NodeList      10.1.1.2      10.1.1.5   10.1.1.7
```

Note: ISS will be able to perform normal DNS resolution to map the hostname [e.g. alien.foo.com] to one of the IP address [e.g. 10.1.1.1]. You only need to specify an interface if you wish the same machine to use multiple IP addresses.

The InternalNet address will be used by ISS for its internal communications transparently: you do not need to configure anything else to make this work.

When defining an interface, (using the **Interface** keyword, **ISSControl add Interface**, or the GUI), you may identify either the IP address, or the

hostname. If you supply the hostname, then ISS will look up the IP address which matches it. If you supply the IP address, then there will be no further resolution.

Using ISS and Dispatcher in a two-tiered configuration

As shown in Figure 28, IBM Network Dispatcher supports the concept of a two-tiered architecture. For a complete sample configuration, see “A Two Tier Sample configuration file for ISS” on page 267. Using this configuration, two components of IBM Network Dispatcher are used together to maximize load balancing for your server network. This approach is especially useful if you need to provide load balancing for servers that are located in different geographical areas. ISS is supplied with information from each of the local Dispatchers and uses an intelligent round-robin or user-specified metric to determine which Dispatcher IP address is returned to the client. The combination of the ISS and Dispatcher functions maximizes the potential load for your worldwide network.

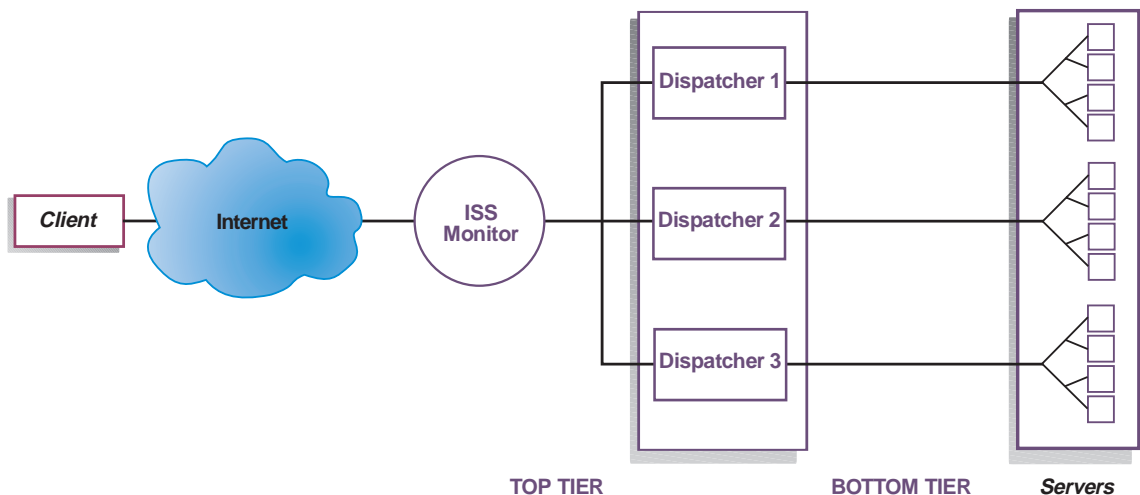


Figure 28. Example of a two-tiered configuration using IBM Network Dispatcher

In this example:

- The server site appears to clients and end users as a single site.
- The server site can be made up of several groups of clustered servers, each associated with a Dispatcher. (The Dispatcher may be shared by more than one group of clustered servers.) The groups of servers can be geographically separated.

- The ISS function handles the geographical load balancing to the top tier of Dispatchers, and the Dispatcher function handles local load balancing to the bottom tier of servers. The Dispatcher machines are configured as Dispatcher observers.
- ISS, through name resolution, provides the client with the IP address of the least-busy Dispatcher. Then, when the client goes to that address, the Dispatcher forwards the client's request to the server the Dispatcher selects.
- The Dispatcher selects a server based on weights it calculates internally. The settings for these weights can also be changed by the user. For a description of these settings, see "Optimizing the load balancing provided by Dispatcher and CBR" on page 84.

Optionally, the ISS agent can run on the servers to provide system load information to the Dispatchers.

Configuring highly available Dispatchers in a two-tier configuration

Configuring ISS to work with highly available Dispatchers should be similar to the sample shown in "A Two Tier Sample configuration file for ISS" on page 267 with just the following modifications:

- Instead of defining the cluster with the Node keyword, the cluster should be defined with the Interface keyword on the "Active" Dispatcher node.
- The NodeList of the Service should then define a single instance of the cluster address.
- Using the Dispatcher high availability goActive script, isscontrol commands should: delete the cluster from one Dispatcher Node, add the cluster to the other Dispatcher Node, then add the cluster back to the Service's NodeList.

There are some restrictions in getting this configuration to work.

- Each cluster must only be configured for one node's interface.
- A "Startup" config file for ISS must exist without any cluster definitions.
- All participating machines should have started ISS before starting the Dispatcher's high availability feature.
- The goActive scripts on each Dispatcher will delete/add the cluster appropriately (see below).
- Each Service may contain only one cluster per Dispatcher (one interface per Service).

For example, the ISS configuration file would initially contain the following node and service definitions:

```
Node ISSMonitor 01
Node dispatcherA 02
Node dispatcherB 03
```

```
Service Top_Tier www.PickActiveDispatcher.com
ResourceList checkND
SelectionMethod BEST
NodeList
```

The Dispatcher's goActive script would also contain the following isscontrol commands (example for Node dispatcherA):

```
/usr/bin/isscontrol delete interface clusterAddr from node dispatcherB
/usr/bin/isscontrol add interface clusterAddr to node dispatcherA
/usr/bin/isscontrol add interface clusterAddr to service Top_Tier
```

Since dispatcherB doesn't have an interface in the initial configuration file, an error message will be written to the ISS log, but ISS will continue to run. Once the active Dispatcher starts, these commands will be issued. The configuration file should contain the following:

```
Node ISSMonitor 01
Node dispatcherA 02
Interface clusterAddr
Node dispatcherB 03
Service Top_Tier www.PickActiveDispatcher.com
ResourceList checkND
SelectionMethod BEST
NodeList clusterAddr
```

In order for ISS to recommend only the active Dispatcher and not the standby, a Dispatcher resource is needed to "fail" the standby Dispatcher. This resource should call an external metric that determines the state of the Dispatcher. If it is active, a "good" value is returned. If it is standby, a "failed" value is returned. For the previous example, the resource could be defined as follows:

```
ResourceType checkND
Metric external /usr/lpp/nd/dispatcher/NdState
MetricNormalization 0 10
MetricLimits 5 8
Policy Min
```

One implementation of the NdState script would search the machine's list of interfaces using the **netstat -ni** command (for Unix based platforms) or using the **ipconfig** command (for Windows). A return value of 1 results if clusterAddr is aliased on a NIC (tr0 or en0). A return value of 9 results if clusterAddr is aliased on the loopback adapter. Due to the MetricLimits configured, ISS would fail the node which returned a value of 9 and recommend only the node that returned a value of 1.

Note: Undesirable results will occur if the cluster address is defined with multiple Interface keywords or if defined multiple times in the Service's NodeList. The order in which ISS and Dispatcher are started is very important. All ISS nodes will only be updated with the configuration file changes once initialization has completed.

Since ISS is started without any nodes in the Service's (Top_Tier) NodeList, the metric is not running. To ensure that the metric is running, an additional Service is needed which includes the dispatchers' hostnames, like the following:

```
Service runMetrics placeholder
ResourceList      checkND
SelectionMethod   Best
NodeList          dispatcherA
                  dispatcherB
```

None of the dispatcher observers should include this Service (runMetrics) in any of the ServiceList definitions.

Ping Triangulation Considerations

Ping triangulation is a process by which client requests can be routed to the cell which is geographically closest. Ping triangulation technology uses a process of "echo location " performed on a remote Internet client, allowing you to perform load balancing across multiple synchronized sites. Upon receiving a request, ISS sends an Internet ICMP echo (ping) packet to that host, and times the echo response. It also asks the other sites to perform such a measurement, and inform it of their results.

From this information, the most appropriate site is determined, and future requests may be redirected. Note that ISS will not forward requests to another cell if the local cell is unable to supply the desired service. In ISS, ping triangulation is active only if you have one or more global cells defined and a PingCache defined of size 1kb or greater. Each local cell's configuration file must include the information about the other cells, their nodes, the services to be used, and the ISSNameServer that is being used. Between all the cells, exactly one ISSNameServer should be configured. The node information is used to maintain a connection with the remote cell. The service information is used to defined which services are enabled for ping triangulation. Enabling ping triangulation for a service will allow requests to be redirected according to which cell is located relatively close to the client's location. If the ISSNameServer used in the configuration is in the other cell, it also must be defined in the global cell's definition. An example global cell definition with the ISSNameServer could be:

```
Cell      Hursley      Global
Node      spnode1.hursley.ibm.com 01
Node      spnode2.hursley.ibm.com 02
Node      spnode3.hursley.ibm.com 03
NotMonitor
Service    WWWService      www.ourservice.com
SelectionMethod   Best
NodeList          spnode2.hursley.ibm.com
                  spnode3.hursley.ibm.com
ISSNameserver     spnode1.hursley.ibm.com 53
```

Note: In order for an ISSNameServer to function in a ping triangulation configuration, the ISSNameServer must be located on the monitor. This allows the ISSNameServer to reference the PingCache on that machine.

To configure ping triangulation, you define, in the configuration of each cell, each of the other cells. For example, if you have two cells in the configuration for cell 1, you define cell 1 as a local cell and cell 2 as a global cell. Similarly, cell 2 defines itself as local and cell 1 as global. For a complete sample configuration, see “Ping Triangulation Sample configuration file for ISS” on page 270.

Note: Maintenance and synchronization of the configuration files between cells must be handled manually by the ISS administrator(s). ISSControl and GUI updates will only be sent to the local cell members.

Using Affinity (StickyTime) with ISSNameServer

StickyTime allows an ISSNameServer to control the client-server affinity.

Note: The ISSNameServer affinity is not based on or an application of the TTL (Time To Live) function configured in a name server. This affinity feature creates its own affinity records and functions independently of any TTL settings.

When this affinity feature is *disabled*, the ISSNameServer chooses the right server according to the configuration and provides that server address to the requesting client. The subsequent request from that client will be received as a new connection and the right server address at that moment will be supplied to the client.

When this feature is *enabled*, the ISSNameServer chooses the right server according to the configuration and provides that server address to the requesting client. An entry is made in the ISS affinity table for that client IP. When the same client makes subsequent requests, the ISSNameServer queries the ISS affinity table for that client IP entry. If the entry has not expired according to the StickyTime configured, then the same address that was previously returned to the client is sent again.

Garbage collection is run to clean out all the timed-out affinity table entries. The frequency of garbage collection is configured by the approximate number of requests that must be made before it should occur. A small garbage collection interval will cause it to occur more frequently.

Note: For this feature to work properly, the ISSNameServer must be located on the active ISS monitor. This allows the ISSNameServer to access the ISS affinity table.

To configure **StickyTime** and **StickyGarbageCollectionInterval**, use the `isscontrol` command line to set the values. To *enable* the affinity feature, configure an `ISSNameServer` by setting a `StickyTime` that is greater than zero. The feature is *disabled* by setting `StickyTime` to zero (the default). For more information on these `isscontrol` keywords, see “`isscontrol` — control iss” on page 237.

Chapter 10. Configuring the Interactive Session Support component

This chapter explains how to configure the ISS component of IBM Network Dispatcher. Before following the steps in this chapter, see “Chapter 9. Planning for the Interactive Session Support component” on page 125.

Overview of configuration tasks

Table 11. Configuration tasks for the ISS function

Task	Description	Related information
Define each cell.	A cell requires a name and various attributes.	“Defining a Cell” on page 146
For each cell, define the nodes in the cell.	Nodes are the machines in the cell, identified by id number and IP address or DNS name.	“Defining a Node” on page 147
For each cell, specify one or more resource types.	A resource type is a type of object on which a metric measures load. For example, CPU.	“Defining a Resource” on page 148
For each cell, specify one or more services.	A service corresponds to a <i>group</i> of servers measured by the same resource(s). Each service is identified by an id and a DNS name.	“Defining a Service” on page 149
For each cell, specify the observers that will subscribe to services in that cell.	An observer may be: <ul style="list-style-type: none">• A Network Dispatcher• A regular nameserver• An ISS nameserver	“Defining an Observer” on page 150

Methods of configuration

You will need to create an ISS configuration file to define your configuration. A minimal configuration default file **iss.cfg** has been provided, or you can use an example file. To create the configuration file, see “Modify iss.cfg file or an example file” on page 144.

There are two basic methods of configuring the ISS component:

- Use isscontrol commands, see “ISScontrol commands” on page 144
- Use the ISS GUI, see “ISS GUI” on page 144

Modify **iss.cfg** file or an example file

Edit the default configuration file **iss.cfg** provided, or modify an example configuration file. Examples of configuration files are in “Dispatcher Sample configuration file for ISS” on page 263 and in the `<nd_installation_path>/iss/samples` directory.

ISScontrol commands

Prior to issuing ISS commands, you must modify an ISS configuration file and then start the ISS daemon. Edit the default configuration file **iss.cfg**. Or, for a sample configuration, see “Dispatcher Sample configuration file for ISS” on page 263. To start the ISS daemon, enter **iss** followed by the appropriate parameters. Parameters for the **iss** command are listed in “start iss” on page 236.

Note: For Windows — When issuing the **iss** command from the command prompt to start the ISS daemon, the **-i** (interactive) option must always be included.

Use **isscontrol** commands once the ISS daemon has been started with a configuration file containing the **CELL** keyword and a list of nodes. The **isscontrol** command line can be used to configure the ISS function. These commands allow you to modify a configuration file that defines the resources, services, and observers for a give cell. For more information, see “Appendix C. Command reference for ISS” on page 235 for **isscontrol** commands.

ISS GUI

For an example of the GUI, see Figure 2 on page 6.

Prior to starting the GUI, you must modify an ISS configuration file and then start the ISS daemon.

Modify an ISS configuration file: You can edit the minimal configuration default file **iss.cfg**, or you can edit a sample configuration file found in the `<nd_installation_path>/iss/samples` directory.

Start the ISS daemon: For AIX, Solaris, and Windows, enter **iss** followed by the appropriate parameters. Parameters for the **iss** command are listed in “start iss” on page 236. The **-g** option is required for the administration of the GUI.

Note: For Windows — When issuing the **iss** command from the command prompt to start the ISS daemon, the **-i** (interactive) option must always be included.

Alternatively for Windows, you can start the ISS daemon using the following steps:

1. Click **Settings**
2. Click **Control Panel**, then double-click **Services**.
3. Select **IBM Interactive Session Support**.
4. In the Startup Parameters window, specify the arguments for `iss`.
In the Startup Parameters window, if you want to code a backslash (\), you must specify a double backslash (\\). For example, if the full path name of your configuration file is `d:\ibmiss\iss_config.cfg`, then you would type: `d:\\ibmiss\\iss_config.cfg`.
5. Click **Start**.

Note: The `-c` option for the `iss` command will enable you to use a file other than the default `iss.cfg` configuration file. The default paths are:

- **AIX or Solaris:** - `/etc/iss.cfg`
- **Windows:** - `\Program Files\IBM\nd\iss\iss.cfg`

In order to configure the ISS component from the GUI, you must first select **Interactive Session Support** in the tree structure. This component does not appear on the Red Hat Linux platform. For Red Hat Linux, ISS is replaced by Server Monitor Agent (SMA). (For information on SMA, see “Server Monitor Agent (SMA)” on page 106.

The GUI can be used to do anything that you would do with the `isscontrol` command. For example, to define a cell using the command line, you would enter `isscontrol add cell <name>` command. To define a cell from the GUI, you would click button number two of the mouse on **Host**, then in the pop-up menu, click on **add cell**. Enter the cell name in the pop-up window, then click **OK**.

The **File** menu located at the top of the GUI will allow you to save your current host connections to a file or restore connections in existing files across all Network Dispatcher components.

You can access **Help** by clicking the question mark icon in the upper right hand corner of the Network Dispatcher window.

- **Field Help** — describes each field, default values
- **How do I** — lists tasks that can be done from that screen
- **Contents** — a table of contents of all the Help information
- **Index** — an alphabetical index of the help topics

For more information about using the GUI, see “General Instructions for using the GUI” on page 5.

Updating the ISS configuration file

ISS uses a configuration file to define the cells, resources, services, and observers. A minimal configuration file **iss.cfg** has been provided. The name of any configuration file can be supplied as a command line argument when ISS is run. Sample configuration files are also provided with ISS. See “Sample ISS configuration files” on page 263.

The ISS configuration file is an ASCII file consisting of a series of definitions. Each definition must be at the start of a line, and must start with a keyword that may be followed by further keywords and then a value. Blank lines are ignored and lines that begin with semicolons (;) or hash-symbols (#) are treated as comments. All keywords are case-insensitive. The case of parameters is preserved, since they may contain custom metrics or host names. Each keyword definition in the configuration file refers to a particular object being configured: a cell, node, service, resourcetype, or observer. Any entry for a particular type of object refers to the most recent instance of that type in the file. For example, a definition for the keyword **Portnumber**—which always refers to a given **Cell**—may appear anywhere in the configuration file. It always refers to the most recently defined cell.

A copy of a sample file can be found in “Dispatcher Sample configuration file for ISS” on page 263. It resides in the <nd_installation_path>/iss/samples directory. You can edit it to specify your configuration. With this basic configuration, items in the **isscontrol** commands or the ISS GUI can be used to add/set/delete the configuration. You may have a variety of configuration files for different workloads and other requirements.

The keywords that relate to a specific cell can occur in any order within the cell definition. A complete list of keywords can be found in “Configuration file keywords for ISS” on page 238.

How a configuration file is created

If you plan to use the ISS GUI to develop your configuration, you need a basic configuration file containing the **Cell** keyword and at least the **Node** keyword and their respective definitions.

You may or may not need to create a wholly new configuration. The steps to do so are included below to help you understand how the configuration file works.

Defining a Cell

- Put in a cell for each geographical cell you want to manage. This line begins with the keyword **Cell** and includes a name you assign and a designation of **Global** or **Local**. A global cell is a separate, remote cell.

Adding and deleting sub cell entities for **Global** cells is limited to the following:

- add/delete nodes
- add/delete services
- add/delete issnameserver

All other changes to a global cell must be made locally, where that cell is the local cell. A local cell is the cell that the node will be a member of. There must be one and only one local cell defined in the configuration file.

Example:

```
Cell                Hursley                Local
```

Optionally, the following cell attributes may be specified. If you prefer to change the default values, see the table below.

Table 12. Cell Attributes

Attribute	Default value
PortNumber	7139
LogLevel	INFO
LogSize	unlimited
HeartbeatInterval	10
HeartbeatsPerUpdate	2
HeartbeatsToNetFail	4
HeartbeatsToNodeFail	6
PingCache	0
AuthKey	0

Defining a Node

Specify each of the nodes (host machines) in the cell, using the keyword **Node**, the id number and the IP address or DNS name.

```
Node      Monitor1.hursley.ibm.com  01
```

Note: The id number is used in two ways:

1. As an identifier. As such, it should be a unique, non-zero number.
2. As a priority. Lower values indicate nodes closer in line to be selected as the monitor.

Optionally, the following node attribute may be specified:

- InternalNet
- Interface

- NodeWeight
- NotMonitor
- NotISSAgent
- AuthKey

For more information about these keywords, see “Configuration file keywords for ISS” on page 238.

Defining a Resource

Specify each resource for the cell with the keyword **ResourceType** and a name to identify the resource.

```
ResourceType          CPU
```

For each **ResourceType** define the following attributes:

1. **Metric**

This specifies how to measure load for this resource, either with an **internal** or **external** metric. See “ISS Metrics” on page 132 for more details.

2. **Policy**

This specifies how the Metric will be interrupted by ISS. When you want higher numbers to represent higher loading, set the Policy to **Max**, which indicates that a higher value returned is desirable. When you want lower numbers to represent lower loading, set the Policy to **Min**, which indicates that a lower value returned is desirable. The default value is Min.

3. **MetricNormalization**

This is a range between the lower and upper limits of measurement, indicated as whole numbers. The units referred to depend upon the metric you are using. For example, if you were measuring CPU usage, the units would be percentage points. The range would therefore be 0 to 100. Any metrics coming into the monitor outside this range would be corrected by being clipped to fit within these limits.

4. **MetricLimits**

You can set these limits to prevent a server from failing totally by having too many requests assigned to it. ISS measures the loads on servers periodically. If the loads are within predefined limits, it judges them okay and continues on. If the loads are outside the limits, ISS removes the server from the ranking for that service. (If the server is handling more than one service, it may still be included in the ranking for the other service or services.) There are two levels defined here:

Fail The level beyond which the metric should not go. At **Fail**, ISS removes the server from ranking. In the CPU example, this would be a percentage of utilization approaching but not reaching 100 percent. A reasonable value might be 95 percent when using a Min policy.

Recover

This is the level down to which the metric should be taken before the server is returned to ranking. If Policy is set to Min, the **Recover** level will be somewhat below the **Fail** level. In our example, it might be 90 percent. Specifying a significant difference between these two levels prevents the phenomenon wherein resources come into service, fail, then come back after minimal recovery only to quickly fail again.

5. MetricWeight

This specifies the priority of the metric and the default is 1. See “Configuration file keywords for ISS” on page 238 for more details.

For example, the following coding specifies a metric for finding the CPU load.

Metric	Internal	CPUload
Policy	Min	
MetricNormalization	0	100
MetricLimits	90	95
MetricWeight	1.0	
ResourceType	CPU	

Defining a Service

Specify each service in the cell by the keyword **Service**, a name to identify the service (such as *Service WWWService www.issdev.hursely.ibm.com 129.40.161.74 21*), and a DNS name to which the client will make requests and to which ISS will respond. Services are dependent on a resource(s) and the node(s) that have the resource(s). The following service attributes must be defined for the service to be useful:

- **ResourceList**

specifies the ResourceType name(s) from which this service will collect information.

- **NodeList**

specifies the node(s) on which the resource(s) are available.

- **SelectionMethod**

specifies which method ISS should use to determine which node to recommend. For more information, see “Selection methods” on page 130.

Optionally, two more attributes can be defined on the **Service** keyword line. Those are the Dispatcher cluster address and the Dispatcher port. These two attributes must be defined when a Dispatcher Observer is configured. Although these attributes may identify a specific cluster and port, ISS does not report cluster or port specific information to the Dispatcher. A Dispatcher’s ServiceList should only have one service for each set of servers. If there are two or more services for the same set of servers, the Dispatcher will only use the loads from the last service ISS reports on.

- **Dispatcher Cluster Address**

the address on which Dispatcher is expecting client requests.

- **DispatcherPort**

a port defined in the dispatcher configuration for this cluster address.

```
Service myServiceName placeholder dispatch.ral.ibm.com 80
```

Note: The **DNSname** attribute of the **Service** keyword is not used in this scenario; however, the configuration file must have a placeholder string such as *"placeholder."*

Other optional service attributes are:

- **Overflow**

a node to default to if all the other nodes in the **NodeList** are not available.

- **Alarm**

a command, which will vary according to the operating system, to execute if the **Overflow** node is resorted to.

Defining an Observer

Specify each observer in the cell by one of the following keywords:

ISSNameserver

followed by the DNS name for this node and the port number on which the DNS protocol communicates, usually port 53.

Nameserver

followed by the DNS name for this node and the port number on which the DNS protocol communicates, usually port 53.

Dispatcher

followed by the Dispatcher's NFA and the port number on which the Dispatcher is listening for ISS updates, usually 10004.

For each of the observers specified, the observer attribute, **ServiceList**, must specify to which services the observer subscribes. When a **NameServer** observer is defined, the **NamedData** and **NamedRev** keywords must be specified. For more information, see "Configuration file keywords for ISS" on page 238.

Migrating from an earlier version

If you are modifying an older configuration file (from Version 1.1 or earlier), below are some important points to keep in mind:

1. For every **Metric** in an older configuration file, choose the kind of object on which that metric measures load. Create a **ResourceType** for that kind of object. For example, if you used `vmstat 1 2 | tail -1 | awk '{print $16}'`

to find the system idle time, you might now create a ResourceType CPU, with an external metric of `vmstat 1 2 | tail -1 | awk '{print $16}'`. Incidentally, you can now obtain a percentage measurement of CPU usage using the internal metric “CPULoad.”

- For every machine that was previously specified in the servers list of any pool, add a **Node** entry in the new file, including a unique ID number.
- For every **Pool** in the old file, create a **Service** in the new file. Similarly, the previous list of **Servers** has become the new **Nodelist**, and where previously a **Metric** had been defined, now a **ResourceList** is provided instead.
- Where previously **ISS_MODE** had specified whether a DNS was to be updated, replaced, or ignored, now you create a list of observers, which can subscribe to several load-balancing reports simultaneously. An observer can be a **Nameserver**, an **ISSNameserver**, or a **Dispatcher**.

The configuration file must be protected from alteration under any circumstances. If authentication keys are used, it must also be protected from reading by anyone other than the administrator. It should therefore be kept in a secure area of the node’s file system to prevent unauthorized access.

If you are migrating from v1.2 or later, your old configuration file should be fully compatible. There are some new keywords which you may decide to make use of, see “Configuration file keywords for ISS” on page 238.

Setting up the TCP/IP name server

Use the questions in Table 13 to help you set up the TCP/IP name server used by ISS.

Table 13. Guidelines for setting up the name server used by ISS.

Question	Related Information
Do you want to add your ISS service names to an existing domain name server?	<p>If no, go to the next question.</p> <p>If yes, you can either:</p> <ul style="list-style-type: none"> • Add the ISS service DNS names to a domain served by your existing name server, or • Set up a new subdomain for the ISS service names. <p>If you do not want to set up a new subdomain, simply add the ISS service names to your name server data file. For instructions on setting up a new subdomain, go to “Setting up a subdomain for your ISS service DNS names” on page 152.</p> <p>Note: For Windows, NameServer requires the whole NameServer to be stopped and restarted each time an update is made (there is no “HUP” equivalent), so we do not support the use of ISS with this particular NameServer.</p>

Table 13. Guidelines for setting up the name server used by ISS (continued).

Question	Related Information
How do you want to set up your ISS name server?	<p>You can set up the ISS name server in either of the following ways:</p> <ul style="list-style-type: none"> • Add the ISS service names as a subdomain of the new ISS name server, and modify the client machines so that they go first to the new name server. “Setting up a subdomain for your ISS service DNS names” describes how to set up a subdomain for the ISS service names and how to set up a machine to act as a name server for that subdomain. “Configuring the ISS client machines” on page 156 explains how to modify the client workstations so that they use the new name server. • Add the servers as a subdomain of the new ISS service names, and have your existing name server pass along requests for the new subdomain. With this approach, you do not have to modify the client configuration files. The site name server can be configured to pass along anything for the new ISS subdomain. This allows the client workstations to remain unchanged, directing all requests for name resolution to the site name server. “Setting up a subdomain for your ISS service DNS names” describes how to set up a subdomain for the ISS service names and how to set up a machine to act as a name server for that subdomain. Note: This approach is best suited for large sites. If your site has a large number of users, it may not be appropriate to add the ISS service names to the site name server because ISS causes the name server to reload its database quite frequently. For large named data files, the overhead of doing this could be prohibitive. In such sites, the ISS service names should be added to a separate subdomain, administered by a separate name server machine.
Do you want to set up a backup for your ISS DNS name server?	If yes, go to “Do I need an ISS backup?” on page 157.

Setting up a subdomain for your ISS service DNS names

The DNS name server files presented in this section are designed to work in conjunction with the following ISS configuration file:

```
Cell          RTP_site local
#             ...
#             ...
```



```

Node          issmon1.site.corp.com  01
Node          server1.site.corp.com  10
Node          server2.site.corp.com  11
Node          server3.site.corp.com  12
#
ResourceType  CPULoad_1
Metric        Internal CPULoad
Policy        Min
#
Service       service1 pool1.rtp.site.corp.com
NodeList      server1.site.corp.com
              server2.site.corp.com
              server3.site.corp.com
ResourceList  CPULoad_1
SelectionMethod Best
#
NameServer    issmon1.site.corp.com
ServiceList   service1
NamedData     /etc/named.data
NamedRev      /etc/named.rev

```

The service *service1* has the DNS name pool1.rtp.site.corp.com associated with it. Also associated with service1 are the nodes server1, server2, and server3. Since the type of observer is **NameServer**, ISS will work in conjunction with the DNS name server running on node issmon1 to resolve the DNS name pool1.rtp.site.corp.com to the IP address of either server1, server2, or server3. Note that the name of the DNS NameServer must be listed in the named.data file (see “Editing the Data File” on page 154).

Since the SelectionMethod is **Best**, ISS will attempt to map pool1 to the “best” of the three servers using the ISS internal metric CPULoad. The DNS name server running on the node issmon1 has authority over the subdomain rtp.site.corp.com. Note that the server nodes associated with service1 are not designated as being part of the subdomain rtp.site.corp.com. This may seem to contradict a statement made earlier in Table 13 on page 151 that the server nodes must be part of this subdomain. However, that requirement can be relaxed. It is not necessary to change the server machines in order for ISS to provide load balancing for those machines.

The three files that determine the behavior of the DNS name server are the boot file, the data file and the reverse data file. The following sections describe each file and include editing guidelines.

Editing the Boot File

AIX and Solaris: The boot file is normally called named.boot. If you are starting named daemon from the command line, you can specify the name of this file as a command line parameter. If you do not supply the name, then the named daemon uses the /etc/named.boot file. As root, copy the ISS sample boot file to the appropriate directory and modify it to match your

system configuration. Make sure to make a backup copy of your existing named.boot file before performing this operation.

On AIX or Solaris systems, the following lines must be in the boot file:

```
primary    rtp.site.corp.com      /etc/named.data
primary    in-addr.arpa          /etc/named.rev
```

Editing the Data File

This file is normally called the named.data file and is identified in the boot file. You should copy the sample data file to the chosen directory (for example, /etc for AIX and Solaris) and modify it to match your system configuration.

For the ISS configuration in the example, the associated named.data file contains resource records similar to the following:

```
;
; The SOA (Start of Authority) record
;
rtp.site.corp.com. IN SOA issmon1.site.corp.com.
root.issmon1.site.corp.com. (
                        1000      ; Serial
                        3600      ; Refresh
                        300       ; Retry
                        3600000    ; Expire
                        86400     ; Minimum
;
; The NS (Name Server) records ;
rtp.site.corp.com.      IN NS   issmon1.site.corp.com.
site.corp.com.          IN NS   siteserv.site.corp.com.
;
; The A (address) records ;
loopback.site.corp.com. IN A    127.0.0.1
localhost.site.corp.com. IN A    127.0.0.1
pool1.rtp.site.corp.com. IN A    9.117.17.23
issmon1.site.corp.com.  IN A    9.117.17.31
siteserv.site.corp.com. IN A    9.117.32.254
server1.site.corp.com.  IN A    9.117.17.21
server2.site.corp.com.  IN A    9.117.17.22
server3.site.corp.com.  IN A    9.117.17.23
```

This file specifies that the DNS name server running on the node issmon1.site.corp.com is authoritative for the subdomain rtp.site.corp.com and contains mostly host name to IP address mappings (forward resolution). For completeness, the A records of server1, server2, and server3 are included here although this is really redundant. Any name that the DNS name server running on issmon1 cannot resolve and belongs to the parent domain site.corp.com is passed on to the DNS name server running on siteserv. ISS will periodically modify the IP address of the resource record pool1.rtp.site.corp.com. IN A 9.117.17.23 to that of the “best” of the three servers (server1, server2, and server3). After this modification, ISS asks

the DNS name server running on issmon1 to re-read its configuration files. In this way, load balancing is achieved since the name pool1.rtp.site.corp.com is always mapped to the “best” server. Note that \$INCLUDE statements in named.data file are not supported by ISS.

Editing the Reverse Data File

This file is normally called the named.rev file and is identified in the boot file. You should copy the sample reverse data file to the chosen directory (for example, /etc for AIX and Solaris) and modify it to match your system configuration.

For the ISS configuration in the example, the associated named.rev file contains resource records similar to the following:

```

;
;
; The SOA (Start of Authority) record
;
@      IN      SOA      issmon1.site.corp.com.
root.issmon1.site.corp.com. (
                                1000      ; Serial
                                3600      ; Refresh
                                300       ; Retry
                                3600000   ; Expire
                                86400    ) ; Minimum
;
; Name Servers
;
17.117.9      IN      NS      issmon1.site.corp.com.
117.9         IN      NS      siteserv.site.corp.com.
;
; Pointer records. Reverse address mappings
;
1.0.0.127     IN      PTR      localhost.site.corp.com.
1.0.0.127     IN      PTR      loopback.site.corp.com.
23.17.117.9   IN      PTR      pool1.rtp.site.corp.com.
254.32.117.9  IN      PTR      siteserv.site.corp.com.
31.17.117.9   IN      PTR      issmon1.site.corp.com.

```

ISS will periodically modify the “reverse IP address” of the record 23.17.117.9 IN PTR pool1.rtp.site.corp.com to that of the “best” of the three servers (server1, server2, and server3). This is followed by a request to the DNS name server running on issmon1 to refresh itself with new configuration data.

Starting the DNS name server

Once you have finished editing the boot file, reverse data file, and data file, you can start the name server:

- **AIX:** As root user, enter either of the following commands:

- `/etc/named -b named.boot_file`
Example: `/etc/named -b named.boot`
- `startsrc -s named -a “-b named.boot_file”`
Example: `startsrc -s named -a “-b named.boot”`
- **Solaris:** As root user, enter the following command:
`/usr/sbin/in.named -b /full_path_directory/named.boot_file`

Example: `/usr/sbin/in.named -b /opt/ibmiss/named.boot`

Configuring the ISS client machines

To resolve an ISS DNS service name such as `pool1.rtp.site.corp.com`, it is normally not necessary to make changes to the client machines. The site DNS name server running on `siteserv.site.corp.com` can be configured to pass along resolution requests for the subdomain `rtp.site.corp.com` to the DNS name server running on `issmon1.site.corp.com`. However, if you want to modify the client machine so that the DNS name server running on `issmon1` is the primary name server, it is possible to do so. In the following, we will call the name server running on `issmon1` the ISS name server.

To configure the client machines:

- **AIX:** Follow the instructions presented in this section. These instructions illustrate how to set up the TCP/IP configuration on the client machines so that the ISS name server is the main name server and the site name server is the backup name server. You can do this using SMIT. Using SMIT, select these choices:
 - Communications Applications and Services
 - TCP/IP
 - Further Configuration
 - Name Resolution
 - Domain name server (`/etc/resolv.conf`)
 - Add a name server

When you are finished, ensure that the `/etc/resolv.conf` file has the name server entries in the following order:

1. The ISS name server machine should be first.
 2. The site name server machine should be second.
- **Solaris:** Edit the `/etc/resolv.conf` file so that it has the name server entries in the following order:
 1. The ISS name server machine should be first.
 2. The site name server machine should be second.
 - **Windows NT:** Modify the DNS Service search order using the following steps:

1. Click **Control Panel**, then double-click **Network**.
2. Click **Protocols**, then **TCP/IP Protocol and Properties**.
3. Edit the DNS Service search order in the DNS folder:
 - a. The ISS name server machine should be first.
 - b. The site name server machine should be second.
- **Windows 2000:** Modify the DNS Service search order using the following steps:
 1. Click **Control Panel**, then **Network** and **Dial-up Connections**.
 2. Right click on the active (enabled) connection and select **Properties**.
 3. Select **Internet Protocol (TCP/IP)**, then **Properties**.
 4. Click **Advanced**, then select the **DNS** tab.
 5. Edit the DNS Service search order:
 - a. The ISS name server machine should be first.
 - b. The site name server machine should be second.

To test the operation of the ISS name server, make a name resolution request from a client machine.

- **AIX and Windows:** Enter **ping pool1.rtp.site.corp.com**.
- **Solaris:** Enter **/usr/sbin/ping -s pool1.rtp.site.corp.com**.

This returns the dotted-decimal Internet address for **pool1** that is specified in the named.data file. Note that if the `/etc/resolv.conf` file of the client machine contains the statement

```
search rtp.site.corp.com site.corp.com corp.com
```

then `ping pool1.rtp.site.corp.com` can be abbreviated to `ping pool1`.

Do I need an ISS backup?

ISS handles high availability internally: all nodes (except those labelled "NotMonitor" "NotISSAgent") can take on a monitor role, and thus there will always be an ISS presence unless all of those nodes are unavailable. Therefore it is not necessary to configure an ISS backup. In DNS-replace mode, you may choose to configure more than one ISS NameServer in order to increase the DNS availability. In DNS-update mode, you may choose to list more than an NameServer to increase availability.

Starting ISS

When configuration is completed, go to "Using the Interactive Session Support component" on page 167 for instructions on how to start ISS.

Chapter 11. Operating and managing IBM Network Dispatcher

This chapter explains how to use the components of IBM Network Dispatcher and includes the following sections:

- “Using the Dispatcher component”
- “Using the Content Based Routing component” on page 171
- “Using the Interactive Session Support component” on page 167
- “Remote Authenticated Administration” on page 161
- “Using Simple Network Management Protocol with the Dispatcher” on page 162

Using the Dispatcher component

This section explains how to use the Dispatcher component.

Starting Dispatcher

To start the Dispatcher component, you can either enter commands from the command line, use the GUI, or put the commands into a configuration file, for example `mysample.cmd`. For a description of the required commands, go to “Setting up the Dispatcher machine” on page 55.

Stopping the Dispatcher

Stop all functions of the Dispatcher by:

- Entering **ndcontrol executor stop** (AIX, Red Hat Linux, and Solaris)
- Rebooting your Dispatcher machine (Windows)

Using FIN count to control garbage collection

A client sends a FIN packet after it has sent all its packets so that the server will know that the transaction is finished. When Dispatcher receives the FIN packet, it marks the transaction from active state to FIN state. When a transaction is marked FIN, the memory reserved for the connection can be cleared by the garbage collector built into the executor.

You can use the FIN timeout and count to set how often the executor will perform garbage collection and how much it will perform. The executor periodically checks the list of connections it has allocated. When the number of connections in the FIN state is greater than or equal to the FIN count, the executor will attempt to free the memory used to hold this connection information. You can change the FIN count by entering the **ndcontrol executor set fincount** command.

The garbage collector frees the memory for any connection that is in the FIN state and is older than the number of seconds specified in the FIN timeout. You can change the FIN timeout by entering the **ndcontrol executor set fintimeout** command.

The FIN count also affects how often “stale” connections are removed. A connection is considered stale when there has been no activity on that connection for the number of seconds specified in the **ndcontrol port set staletimeout** command. If you have little memory on your Dispatcher machine, you should set the FIN count lower. If you have a busy Web site, you should set the FIN count higher.

Using Dispatcher logs

Dispatcher posts entries to a server log, a manager log, a log for each advisor you use, and a subagent log. You can set the logging level to define the expansiveness of the messages written to the log. At level 0, only errors are logged. At level 1 (the default), Dispatcher also logs headers and records of events that happen only once (for example, a message about an advisor starting to be written to the manager log). Level 2 includes ongoing information, and so on, with level 5 including every message produced to aid in debugging a problem if necessary.

You can also set the maximum size of a log. When you set a maximum size for the log file, the file will wrap; when the file reaches the specified size, the subsequent entries will be written at the top of the file, overwriting the previous log entries. You cannot set the log size to a value that is smaller than the current one. Log entries are timestamped so you can tell the order in which they were written.

The higher you set the log level, the more carefully you should choose the log size. At level 0, it is probably safe to leave the log size to the default of unlimited. However, when logging at level 3 and above, you should limit the size without making it too small to be useful.

Changing the log file paths

By default, the logs generated by the Dispatcher will be stored in the logs directory of the Dispatcher installation. This path can be changed by setting the *nd_logdir* variable in the *ndserver* script.

AIX, Red Hat Linux, and Solaris: The *ndserver* script is found in */usr/bin/ndserver*. In this script, the variable *nd_logdir* is set to the default directory. You can modify this variable to specify your log directory. Example:

```
ND_LOGDIR=/path/to/my/logs/
```


Windows: The `ndserver.cmd` script is found in the Windows system directory, typically `C:\WINNT\SYSTEM32`. In `ndserver.cmd`, the variable `nd_logdir` is set to the default directory. You can modify this variable to specify your log directory. Example:

```
set ND_LOGDIR=c:\path\to\my\logs\
```

For all operating systems, make sure that there are no spaces on either side of the equal sign and that the path ends in a slash ("/" or "\" as appropriate).

Binary logging

The binary logging feature of Dispatcher uses the same log directory as the other log files. See "Using binary logging to analyze server statistics" on page 121.

Reporting GUI

Various charts can be displayed based on information seen by the executor and relayed to the manager:

- Connections per second per cluster
- Connections per second per port (multiple ports could be shown on the same graph)
- Connections per second per server (multiple servers could be shown on the same graph)
- Relative weighting values per server on a particular port
- Average connection duration per server on a particular port

Remote Authenticated Administration

Network Dispatcher provides an option to run its configuration programs on a machine other than the one running the Network Dispatcher servers.

Communication between the configuration programs (`ndcontrol`, `cbrcontrol`, `isscontrol`, `ndwizard`, `ndadmin`) is performed using Java Remote Method Invocation (RMI) calls. The command to connect to a Network Dispatcher machine for remote administration is **`ndcontrol host:remote_host`**. If the RMI call comes from a machine other than the local machine, a public key/private key authentication sequence must occur before the configuration command will be accepted.

Communication between the control programs running on the same machine as the component servers are not authenticated.

Use these commands to generate these public and private keys to be used for remote authentication:

ndkeys [create | delete]

cbrkeys [create | delete]

isskeys [create | delete]

These commands run only on the same machine as the Network Dispatcher.

The create option causes a public key to be created in the components key directory and a private key to be created in the components administration keys directory. The private key is created with the address of the server and the RMI port it is listening on as the file name. These private keys must then be transported to the remote clients and placed in the appropriate components administration keys directory.

For example if the **ndserver** process has been started on a machine with hostname address 10.0.0.25 and is listening on RMI port 10099, the **ndkeys create** command generates two files:

The public key /usr/lpp/nd/dispatcher/key/authorization.key
The private key /usr/lpp/nd/admin/keys/dispatcher/10.0.0.25-10099.key

The administration fileset has been installed on another machine. The 10.0.0.25–10099.key file must be placed in **/usr/lpp/nd/admin/keys/dispatcher/** on the remote client machine.

The remote client will now be authorized to configure **ndserver** on 10.0.0.25.

This same key must be used on all remote clients that you want to authorize to configure **ndserver** on 10.0.0.25.

If you were to run the **ndkeys create** command again, a new set of public/private keys would be generated. This would mean that all remote clients who tried to connect using the previous keys would not be authorized. The new key would have to be placed in the correct directory on those clients you want to reauthorize.

The **ndkeys delete** command deletes the public and private keys on the server machine. If these keys are deleted, no remote clients will be authorized to connect to the servers.

Using Simple Network Management Protocol with the Dispatcher

Note: For Red Hat Linux, SNMP is not supported on IBM Network Dispatcher.

A network management system is a program that runs continuously and is used to monitor, reflect status of, and control a network. Simple Network Management Protocol (SNMP), a popular protocol for communicating with devices in a network, is the current network management standard. The network devices typically have an SNMP *agent* and one or more subagents. The SNMP agent talks to the *network management station* or responds to command line SNMP requests. The SNMP *subagent* retrieves and updates data and gives that data to the SNMP agent to communicate back to the requester.

Dispatcher provides an SNMP *Management Information Base* (ibmNetDispatcherMIB) and an SNMP subagent. This allows you to use any network management system, such as — Tivoli NetView, Tivoli Distributed Monitoring, or HP OpenView — to monitor the Dispatcher's health, throughput, and activity. The MIB data describes the Dispatcher being managed and reflects current Dispatcher status. The MIB will be installed in the /nd/admin/MIB/ subdirectory.

The network management system uses SNMP GET commands to look at MIB values on other machines. It then can notify you if specified threshold values are exceeded. You can then affect Dispatcher performance, by modifying configuration data for Dispatcher, to proactively tune or fix Dispatcher problems before they become Dispatcher or Web server outages.

SNMP commands and protocol

The system usually provides an SNMP agent for each network management station. The user sends a GET command to the SNMP agent. In turn, this SNMP agent sends a GET command to retrieve the specified MIB variable values from a subagent responsible for those MIB variables.

Dispatcher provides a subagent that updates and retrieves MIB data. The subagent responds with the appropriate MIB data when the SNMP agent sends a GET command. The SNMP agent communicates the data to the network management station. The network management station can notify you if specified threshold values are exceeded.

The Dispatcher SNMP support includes an SNMP subagent that uses Distributed Protocol Interface (DPI) capability. DPI is an interface between an SNMP agent and its subagents.

Enabling SNMP on AIX and Solaris

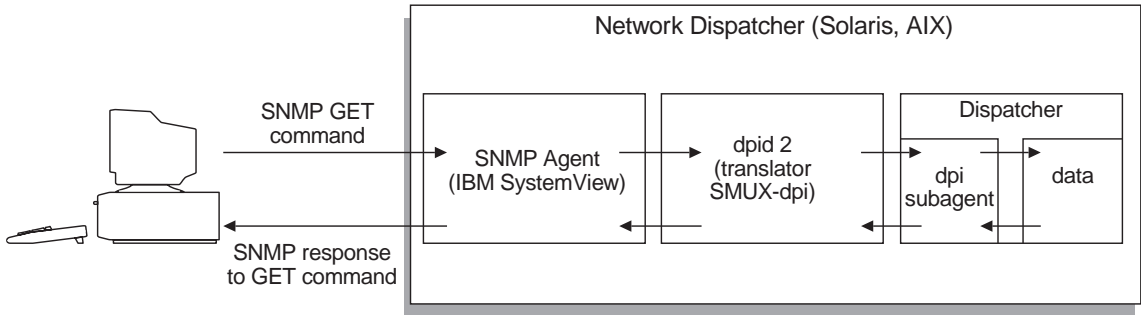


Figure 29. SNMP commands for AIX and Solaris

AIX provides an SNMP agent that uses SNMP Multiplexer protocol (SMUX). You must also use an SNMP agent on Solaris that is SMUX-enabled. DPID2 is an additional executable that works as a translator between DPI and SMUX.

For Solaris, DPID2 is provided in the `/opt/nd/dispatcher/samples/dpid2` directory.

The DPI agent must run as a root user. Before you execute the DPID2 daemon, update the `/etc/snmpd.peers` file and the `/etc/snmpd.conf` file as follows:

- In the `/etc/snmp/conf.peers` file, add the following entry for dpid:

```
"dpid2"          1.3.6.1.4.1.2.3.1.2.2.1.1.2
"dpid_password"
```
- In the `/etc/snmp/conf.file`, add the following entry for dpid:

```
smux          1.3.6.1.4.1.2.3.1.2.2.1.1.2
dpid_password      #dpid
```

Refresh `snmpd` so that it will reread the `/etc/snmpd.conf` file:

```
refresh -s snmpd
```

Start the DPID SMUX peer:

```
dpid2
```

The daemons must be started in the following order:

1. SNMP agent
2. DPI translator
3. Dispatcher subagent

Enabling SNMP on Windows

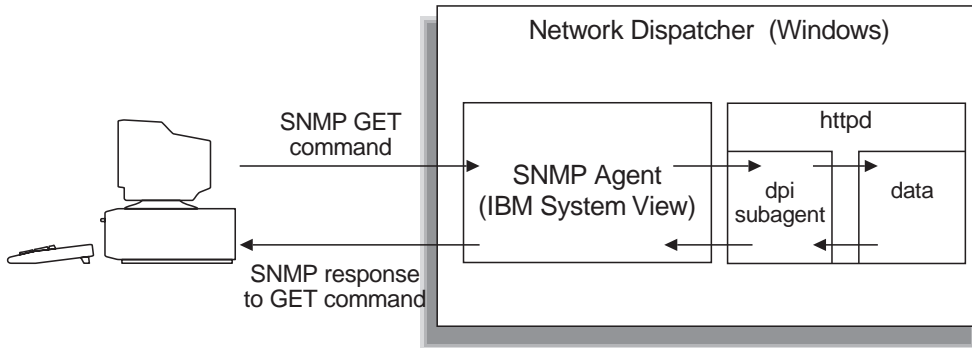


Figure 30. SNMP commands for Windows

To get a DPI-capable SNMP agent for Windows, install the IBM SystemView Agent toolkit from <http://www.support.tivoli.com/sva/shasdk.html>.

Before you start the SystemView SNMPD process, you must disable the Microsoft Windows SNMP support. The SystemView snmpd supports DPI subagents and Microsoft-compliant agents.

To disable the Windows NT SNMP support:

1. From your Windows NT desktop, click the **My Computer** icon.
2. Click the **Control Panel** icon.
3. Click the **Services** icon.
4. Select **SNMP**.
5. Click **HW Profiles**.
6. Click **Disable**.

To disable the Windows 2000 SNMP support:

1. Click Start->Programs->Administrative Tools->Services.
2. Right click **SNMP**, then select **Properties**.
3. Change 'Startup type:' to 'Disabled'

Note: If you do not disable the Microsoft Windows SNMP support, the Dispatcher SNMP subagent will not be able to connect to the snmpd agent.

To configure the SystemView SNMP agent, follow the instructions in "Providing a security password for SNMP" on page 166.

Providing a security password for SNMP

You can create community names (passwords). The default SNMP community name is `public`. In UNIX systems, this is set up in a file named `/etc/snmpd.conf`.

For Windows, before creating passwords for server communities, configure the SystemView SNMP agent.

1. From your desktop, click the **SystemView Agent** icon.
2. Click on **snmpcfg**.
3. In the SNMP Configuration dialog box, add the community name. For testing purposes, enter **public** as the community name. Make sure that `svstart.batch` has been started.

This step allows any host in any network to access the SNMP MIB variables. After you have verified that these values work, you can change them according to your requirements.

4. Start `snmpd.conf`.
5. Check the `sva.batch` file and ensure that the **-dpi** option is specified before starting `snmpd`.

Use the **ndcontrol subagent start** command to define the password used between the Dispatcher DPI subagent and the SNMP agent. The default is `public`. If you change this value, you must also add the new community name to the SystemView Agent `snmpcfg`.

Traps

SNMP communicates by sending and receiving *traps*, messages sent by managed devices to report exception conditions or the occurrence of significant events, such as a threshold having been reached.

The subagent uses two traps:

- `indHighAvailStatus`
- `indSrvrGoneDown`

The **indHighAvailStatus** trap announces that the value of the high-availability status state variable (`hasState`) has changed. The possible values of `hasState` are:

- idle** This machine is load balancing and is not trying to establish contact with its partner Dispatcher.
- listen** High availability has just started and the Dispatcher is listening for its partner.
- active** This machine is load balancing.

-standby

This machine is monitoring the active machine.

-preempt

This machine is in a transitory state during the switch from primary to backup.

-elect The Dispatcher is negotiating with its partner regarding who will be the primary or backup.

-no_exec

The executor is not running

The **indSrvrGoneDown** trap announces that the weight for the server specified by the csAddr, psNum, ssAddr portion of the Object Identifier has gone to zero. The last known number of active connections for the server is sent in the trap. This trap indicates that, as far as the Dispatcher can determine, the specified server has gone down.

Due to a limitation in the SMUX API, the enterprise identifier reported in traps from the ibmNetDispatcher subagent may be the enterprise identifier of dpid2, instead of the enterprise identifier of ibmNetDispatcher, 1.3.6.1.4.1.2.6.144. However, the SNMP management utilities will be able to determine the source of the trap because the data will contain an object identifier from within the ibmNetDispatcher MIB.

Turning the SNMP support on and off from the ndcontrol command

The **ndcontrol subagent start** command turns the SNMP support on. The **ndcontrol subagent stop** command turns the SNMP support off.

For more information about the ndcontrol command, see “ndcontrol subagent — configure SNMP subagent” on page 233.

Using the Interactive Session Support component

Operating and managing Interactive Session Support

This section explains how to use the Interactive Session Support (ISS) component and includes the following information:

- “Starting ISS” on page 168
- “Reconfiguring ISS while it is running” on page 170
- “Establishing an interactive session” on page 170
- “Ending an interactive session” on page 170
- “Stopping ISS” on page 170

Starting ISS

ISS operates as either monitor or agent depending on the parameters entered. Use the `iss` command to start the ISS daemon.

The `iss` command can be used with the following parameters:

-c *config_file*

Use the specified file as the configuration file. Each node should have the same configuration file. The default paths are:

- **AIX and Solaris:** `/etc/iss.cfg`
- **Windows:** `\Program Files\IBM\nd\iss\iss.cfg`

-g<portnum>

Enables the administration GUI either on default port 12099 or the <portnum> specified.

-h

Displays a help message.

-i

Run `iss` interactively.

-l *log_file*

Use the specified file as the log file instead of `/etc/iss.log`. If you are using a shared file system, each log file should be given a unique name. The default paths are:

- **AIX and Solaris:** `/var/log/iss.log`
- **Windows:** `\Program Files\IBM\nd\iss\iss.log`

-p <portnum>

Use the specified port number for control traffic.

AIX and Solaris: You can run `iss` in the foreground or the background. To start `iss`, as root, enter:

bin/iss

followed by any of the parameters listed above.

Notes:

1. **AIX and Solaris:** If you want to follow what the daemon is doing, type `tail -f <logfile>`
2. If you are using a TCP/IP name server as an Observer, ensure that the name server is running and is correctly resolving names from the client machines before you start ISS. If the name server is not running, ISS

generates an error message and continues. If that occurs, check the name server configuration, start the name server, and try again.

3. **Windows:** Since **iss** runs as a Windows service, messages generated by **iss** are directed to a file. The default file is `c:\Program Files\IBM\nd\iss\iss.log`. To specify a different file, use the `-l` parameter described above.

Windows: **iss** runs as a Windows service. To start **iss**, as administrator:

1. For Windows NT: Click **Control Panel**, then double-click **Services**.
For Windows 2000: Click Start->Programs->Administrative Tools->Services.
2. Select **IBM Interactive Session Support**.
3. In the Startup Parameters window, specify the arguments for **iss**.
In the Startup Parameters window, if you want to code a backslash (\), you must specify a double backslash (\\). For example, if the full path name of your configuration file is `d:\ibmiss\iss_config.cfg`, then you would type: `d:\\ibmiss\\iss_config.cfg`.
4. Click **Start**.

When started, ISS displays an opening message. If it does not find any errors in the ISS configuration file, it begins to load balance TCP/IP connections transparently across the cells defined in the configuration file. If there are problems with the cell definitions in the configuration file, a message is generated and ISS exits. If that occurs, modify the configuration file and try again. Refer to the sample configuration file for guidance.

ISS is designed to run unattended in normal operation, although sites under heavy load will inevitably need administrators to restart failed services and nodes. In normal operation, each node may be started and stopped independently. (See "Controlling ISS" on page 171.) When a node starts, it parses the configuration file and creates entries for each object it will interact with, including its peer nodes. From that list, it works out the preferred order of monitor machines from the node numbers and starts attempting to contact them all at once. The highest-priority node that responds will be chosen as the monitor. If a given higher-priority node does not respond within the timeout period, it is considered out of service and a node with a lower priority is chosen as the monitor.

The monitor has the power to change the configuration of each node and to start and stop it from sampling metrics. While agents are sending metric information to the monitor, they expect regular acknowledgments. If an acknowledgment is not received within the timeout period, the agents consider the monitor out of service and another monitor is selected.

Reconfiguring ISS while it is running

To reconfigure ISS while it is running, use the **isscontrol** command or the ISS GUI. See “Appendix C. Command reference for ISS” on page 235.

Establishing an interactive session

An interactive session can be established with any standard TCP/IP application such as Telnet, RSH, FTP, or rlogin. When the application expects you to provide the machine name or Internet address of the remote host to which you want to connect, enter instead the name of the service to which you want to connect, as in the following example:

```
telnet dept_pool
```

dept_pool is the DNS name of the ISS-defined service. By using the service name, the user is automatically connected to a least-loaded server providing that service, using the leveling metric defined in the configuration file.

Applications that specify TCP/IP machine names or Internet addresses for client server connections should specify the service name in their TCP/IP configuration files.

Depending on the way your site name server is set up and the relationship between the TCP/IP domains in which the client and server machines are located, you may need to specify the service name more fully. For example, *dept_pool.sp* or even *dept_pool.sp.site.corp.com*. This may be necessary if your client is served by AFS.

Ending an interactive session

You can end interactive sessions in the same manner as you would in the absence of ISS. For example, to end an FTP session, enter **quit** at the FTP prompt.

Stopping ISS

AIX and Solaris: As root, enter:

```
isscontrol shutdown <nodename>
```

When ISS is not running, access requests directed to the service names will be resolved to the machine address that was current for each service when ISS was terminated.

Windows: To stop ISS, as administrator:

1. For Windows NT: Click **Control Panel**, then double-click **Services**.
For Windows 2000: Click Start->Programs->Administrative Tools->Services.
2. Select **IBM Interactive Session Support**, then click **Stop**.

Controlling ISS

ISS is controlled through the **isscontrol** command. The complete syntax of this command is described in “Appendix C. Command reference for ISS” on page 235. Some example uses are listed here.

Example 1

```
isscontrol add node dart.hursley.ibm.com 1
```

The above command adds the machine named dart to the cell, and specifies that it is the first machine to be considered as the monitor.

Example 2

```
isscontrol delete node dart.hursley.ibm.com
```

The above command deletes the machine dart from the cell.

Example 3:

```
isscontrol add node wingnut 3  
isscontrol add node wingnut service CorpHTTP
```

The above commands perform the following sequence:

1. Add node wingnut to the local cell. It is third in line to be the monitor.
2. Add node wingnut to the servers list for the CorpHTTP service.

Using the Content Based Routing component

This section explains how to use the CBR component of Network Dispatcher.

Starting and Stopping CBR

CBR proxy (for IMAP or POP3 protocol)

CBR proxies request for IMAP or POP3 mail requests. To start CBR proxy, issue **cbrserver** command at the command prompt. After issuing cbrserver, CBR proxy can be configured as described in “CBR configuration example” on page 80. To stop CBR, issue **cbrserver stop** command at the command prompt.

CBR with WTE (for HTTP protocol)

CBR is a subprocess of WTE. Before you start WTE, CBR must be configured as described in “CBR configuration example” on page 80. Once configured correctly, CBR is started and stopped by starting and stopping the WTE application.

Using CBR logs

The logs used by CBR are similar to those used in Dispatcher. For more description, see “Using Dispatcher logs” on page 160. One difference is how you change the directory where the logs are stored. This directory is set in the WTE configuration file. One of the four entries is ServerInit. This is the entry containing many comma delimited arguments. The second to last argument is the log directory. To change the log directory, change this argument and restart WTE.

Chapter 12. Troubleshooting

This chapter helps you detect and resolve problems associated with IBM Network Dispatcher. Find the symptom you are experiencing in “Troubleshooting tables”. Find “Using ISS trace and debug facilities” on page 184.

Troubleshooting tables

These are the troubleshooting tables for Dispatcher, ISS, and CBR.

Table 14. Dispatcher troubleshooting table

Symptom	Possible Cause	Go to...
Dispatcher not running correctly	Conflicting port numbers	“Checking Dispatcher port numbers” on page 176
Configured a collocated server and it will not respond to load balanced requests	Wrong or conflicting address	“Problem: Dispatcher and server will not respond” on page 178
Connections from client machines not being served or connections timing out	<ul style="list-style-type: none">• Wrong routing configuration• NIC not aliased to the cluster address• Server does not have loopback device aliased to the cluster address• Extra route not deleted• Port not defined for each cluster• Servers are down or set to a weight of zero	“Problem: Dispatcher requests are not being balanced” on page 178
High availability not working	Client machines are not being served or are timing out	“Problem: Dispatcher high-availability function is not working” on page 178
Unable to add heartbeat (Windows)	Source address is not configured on an adapter	“Problem: Unable to add heartbeat (Windows)” on page 178
Server not serving requests (Window)	An extra route has been created in the routing table.	“Problem: Extra routes (Windows only)” on page 179

Table 14. Dispatcher troubleshooting table (continued)

Symptom	Possible Cause	Go to...
Advisors not working correctly with wide area	Advisors are not running on remote machines	"Problem: Advisors not working correctly" on page 179
SNMPD will not start or will not continue to run (Windows)	The community name passed in the SNMP commands does not agree with the community name with which the subagent was started.	"Problem: SNMPD does not run correctly (Windows only)" on page 179
Dispatcher, Microsoft IIS, and SSL are not working or will not continue	Unable to send encrypted data across protocols.	"Problem: Dispatcher, Microsoft IIS, and SSL do not work (Windows only)" on page 179
Connection to remote machine refused	Older version of the keys is still being used.	"Problem: Dispatcher connection to a remote machine" on page 180
The ndcontrol or ndadmin command fails with 'Server not responding' or 'unable to access RMI server' message	Commands fail due to socksified stack.	"Problem: ndcontrol or ndadmin command fails" on page 180
Network Dispatcher machine locks when attempting to collocate the server	Linux kernel patch for aliasing the loopback device prevents collocation	"Problem: Network Dispatcher machine locks up when attempting to collocate" on page 180
"Cannot Find the File..." error message, when running Netscape as default browser to view online help (Windows)	Incorrect setting for HTML file association	"Problem: "Cannot find the file..." error message when trying to view online Help" on page 180

Table 15. CBR Troubleshooting table

Symptom	Possible Cause	Go to...
CBR not running correctly	Conflicting port numbers	"Checking CBR port numbers" on page 177
Server is not responding (CBR w/WTE)	CBR server did not get started within WTE.	"Problem: Server not responding (CBR w/WTE)" on page 181
URL rule doesn't work (CBR w/WTE)	Syntactical or configuration error	"Problem: Syntactical or configuration error" on page 181

Table 15. CBR Troubleshooting table (continued)

The cbrserver command returns a "java.rmi.RMISecurityException: security.fd.read" exception (CBR proxy)	The system's limit on file descriptors is too small for the number of requests that cbrserver is trying to service.	"Problem: The cbrserver command is stopped" on page 182
Receive proxy error when trying to add a port (CBR proxy)	The cluster address was not configured on a NIC before the proxy was started.	"Problem: Receive CBR proxy error when trying to add a port" on page 182

Table 16. ISS troubleshooting table

Symptom	Possible Cause	Go to...
ISS not running correctly	Conflicting port number	"Checking ISS port numbers" on page 176
Starting ISS from the Services panel produces an error message (Windows)	<ul style="list-style-type: none"> ISS is unable to find the configuration file The configuration file contains an error 	"Problem: Starting ISS from the Services panel produces an error" on page 182
"Error communicating with server" results when attempting to connect to Host using ISS GUI	ISS is not running	"Problem: Unable to get ISS GUI to "Connect to Host"" on page 182
All sessions assigned to one machine	<ul style="list-style-type: none"> One machine has consistently light load Problem with configuration or file permissions 	"Problem: All interactive sessions for ISS are assigned to one machine" on page 183
ISS Trace Message: "Cannot rename 'file 1' to 'file 2'. Permission denied."	ISS may not be running.	"Problem: ISS trace shows "Cannot rename 'file 1' to 'file 2'. Permission denied."" on page 183
ISS not recognizing isscontrol or GUI commands	ISS may not be running on localhost.	"Problem: ISS is not recognising the isscontrol or GUI commands" on page 183
ISS Log file reports: "Config: Could not find local node in the config file."	There are more than two addresses on the machine and the localhost is not used in the ISS configuration.	"Problem: Log file reports "Config: Could not find local node in the config file"" on page 184

Table 16. ISS troubleshooting table (continued)

Symptom	Possible Cause	Go to...
ISS sends loads, but Dispatcher reports wrong values	<ul style="list-style-type: none"> • More than one ISS process running on ISS monitor node. • Server address ISS reporting on doesn't match server address defined in Dispatcher. • Load calculation not within MetricNormalization or MetricLimits range. 	"Problem: ISS is sending loads to Dispatcher, but Dispatcher is reporting the wrong values" on page 184

Checking Dispatcher port numbers

If you are experiencing problems running the Dispatcher, it may be that one of your applications is using a port number that the Dispatcher normally uses. Be aware that the Dispatcher server uses the following port numbers:

- 10099 to receive commands from ndcontrol
- 10004 to receive metric reports from ISS
- 10005 to receive information from an SDA agent

If another application is using one of the Dispatcher port numbers, you can change the Dispatcher's port number by doing the following:

- To change the port used to receive commands, only edit the ndserver script. Change the ND_RMIPORT variable at the top of the ndserver script to the port Dispatcher should use to communicate. To change the port used to receive SDA information, change the ND_AFFINITY_PORT variable in the ndserver script to the port Dispatcher should use to accept SDA information.
- To change the port used to receive metric reports from ISS, provide the METRIC_PORT argument when the manager is started. See the description of the START command syntax at "ndcontrol manager — control the manager" on page 213.

Checking ISS port numbers

If you are experiencing problems running the ISS component, it may be that one of your applications is using a port number that ISS normally uses. Be aware that ISS uses the following port numbers:

- 7139 to allow the monitor and agents to communicate
- 10004 for metric reports from ISS to Dispatcher

- 12099 to allow GUI communications

To change the port number that the monitor and agents communicate on use the -p option when starting ISS.

Checking CBR port numbers

If you are experiencing problems running the CBR with WTE, it may be that one of your applications is using a port number that CBR normally uses. Be aware that CBR uses the following port numbers:

- 11099 to receive commands from cbrcontrol
- 10003 to receive metric reports from ISS

If another application is using one of the CBR port numbers, you can change the CBR's port number by doing the following: Edit the changes made to the WTE configuration file. Specifically, the port numbers are listed in the ServerInit entry. Near the end of the parameter string, otherwise known as INIT_STRING, the two numbers are the communication port and the SDA port. Change them to be the desired port numbers. If changes need to be made to the metric port number, then the manager can be restarted with a specific manager start command which allows you to specify the metric port used.

If you are experiencing problems running the CBR proxy using cbrserver, it may be that one of your applications is using a port number that CBR normally uses. Be aware that CBR uses the following port numbers:

- 13099 to receive commands from cbrcontrol
- 10003 to receive metric reports from ISS

If another application is using one of the CBR port numbers, you can change the CBR's port number by doing the following:

- To change the port used to receive commands from cbrcontrol, edit the cbrserver script. Change the ND_RMIPORT variable in the cbrserver script to the port CBR should use to communicate. For Windows, cbrserver script is in the C:\winnt\system32 subdirectory. For other platforms, cbrserver script file is in the /usr/bin/ subdirectory.
- To change the port used to receive metric reports from ISS, then the manager can be restarted with a specific manager start command which allows you to specify the metric port used.

Solving common problems—Dispatcher

Problem: Dispatcher will not run

This problem can occur when another application is using one of the ports used by the Dispatcher. For more information, go to “Checking Dispatcher port numbers” on page 176.

Problem: Dispatcher and server will not respond

This problem occurs when another address is being used other than the address specified. When collocating the Dispatcher and server, be sure that the server address used in the configuration is the NFA address or is configured as collocated.

Problem: Dispatcher requests are not being balanced

This problem has symptoms such as connections from client machines not being served or connections timing out. Check the following to diagnose this problem:

1. Have you configured the nonforwarding address, clusters, ports, and servers for routing? Check the configuration file.
2. Is the network interface card aliased to the cluster address? Use `netstat -ni` to check.
3. Does the loopback device on each server have the alias set to the cluster address? Use `netstat -ni` to check.
4. Is the extra route deleted? Use `netstat -nr` to check.
5. Use the **ndcontrol cluster status** command to check the information for each cluster you have defined. Make sure you have a port defined for each cluster.
6. Use the **ndcontrol server report::** command to make sure that your servers are neither down nor set to a weight of zero.

Problem: Dispatcher high-availability function is not working

This problem appears when a Dispatcher high-availability environment is configured and connections from the client machines are not being served or are timing out. Check the following to correct or diagnose the problem:

- Make sure you have created the `goActive`, `goStandby`, and `goInOp` scripts, and place them in the `bin` directory where Dispatcher is installed.
- For **AIX**, **Red Hat Linux**, and **Solaris**, make sure the `goActive`, `goStandby`, and `goInOp` scripts have execute permission set.
- For **Windows**, be sure to configure the nonforwarding address.

Problem: Unable to add heartbeat (Windows)

This Windows error occurs when the source address is not configured on an adapter. Check the following to correct or diagnose the problem.

- For **Windows** be sure to configure the nonforwarding address using either the token-ring or ethernet interface and issuing either of the following commands:

```
ndconfig tr0 <ip address> netmask <netmask> or
ndcontrol cluster configure
```

Problem: Extra routes (Windows only)

After setting up server machines, you may find that you have inadvertently created one or more extra routes. If not removed, these extra routes will prevent the Dispatcher from operating. To check for and delete them, see “Setting up server machines for load balancing” on page 60.

Problem: Advisors not working correctly

If you are using wide area support, and your advisors do not seem to be working correctly, make sure that they are started on both the local and the remote Dispatchers. See “Using remote advisors with wide area support” on page 96.

Problem: SNMPD does not run correctly (Windows only)

When using SNMP subagents, if the SystemView Agent SNMP daemon does not start and stay up, be sure that you have configured your SNMP community using the `snmpcfg` program. To access SNMP data from the Dispatcher subagent, the community name passed in the SNMP commands must agree with the community name with which the subagent was started.

Problem: Dispatcher, Microsoft IIS, and SSL do not work (Windows only)

When using Dispatcher, Microsoft IIS, and SSL, if they do not work together, there may be a problem with enabling SSL security. For more information about generating a key pair, acquiring a certificate, installing a certificate with a key pair, and configuring a directory to require SSL, see the *Microsoft Information and Peer Web Services Information and Planning Guide*, which comes with Windows. For Windows NT: Click start->programs->microsoft peer web services (common)->product documentation. For Windows NT and Windows 2000, the local URL for the document, which is viewed by a web browser, is: **file:///C:/WINNT/system32/inetsrv/iisadmin/htmldocs/inetdocs.htm**.

For Windows NT, Click on Chapter 5: “Securing Data Transmissions with Secure Sockets Layer (SSL).” Scroll down to the section entitled “Installing a certificate with a Key Pair” and there is a subsection called *Installing a certificate*.

In the *Installing a certificate* section, add the following two steps after step five:

- On the **Key Manager** screen, the message:
You must now choose a server connection for this key to become fully activated on the target machine

is displayed, and click **OK**.

- On the **Server Connection** screen, click on **Default IP** and click **OK**.

Problem: Dispatcher connection to a remote machine

Dispatcher uses keys to allow you to connect to a remote machine and configure it. The keys specify an RMI port for the connection. It is possible to change the RMI port for security reasons or conflicts. When you change the RMI ports, the filename of the key is different. If you have more than one key in your keys directory for the same remote machine, and they specify different RMI ports, the command line will only try the first one it finds. If it is the incorrect one, the connection will be refused. The connection will not occur unless you delete the incorrect key.

Problem: ndcontrol or ndadmin command fails

The ndcontrol (or cbrcontrol) command returns: "Error: Server not responding." Or, the ndadmin command returns: "Error: unable to access RMI server." These errors can result when your machine has a socksified stack. To correct this problem, edit the socks.cnf file to contain the following lines:

```
EXCLUDE-MODULE java
EXCLUDE-MODULE jre
EXCLUDE-MODULE jrew
EXCLUDE-MODULE javaw
```

Problem: Network Dispatcher machine locks up when attempting to collocate

For Red Hat Linux, this problem occurs if the server machine has the Linux kernel patch installed for aliasing the loopback device. When collocating the Network Dispatcher on the same server machine that it is load balancing, the patch for aliasing the loopback (if installed) must be removed. The patch prevents the Red Hat Linux machine from being collocated. Red Hat Linux supports collocation but only in the absence of high availability.

Problem: "Cannot find the file..." error message when trying to view online Help

For Windows when using Netscape as your default browser, the error message which results with this problem is: Cannot find the file '<filename>.html' (or one of its components). Make sure the path and filename are correct and that all required libraries are available." The problem is due to an incorrect setting for HTML file association.

The solution is the following:

- For Windows NT:
 1. Click **My Computer**, select **View**, select **Options**, and click **File Types** tab
 2. Select "Netscape Hypertext Document"
 3. Click **Edit** button, select **open**, click **Edit** button

4. Enter *NSShell* in the **Application:** field (not the Application Used to Perform Action: field), and click **OK**
- For Windows 2000:
 1. Click **My Computer**, click **Tools**, select **Folder Options**, and click **File Types** tab
 2. Select “Netscape Hypertext Document”
 3. Click **Advanced** button, select **open**, click **Edit** button
 4. Enter *NSShell* in the **Application:** field (not the Application Used to Perform Action: field), and click **OK**

Solving common problems—CBR

Problem: CBR will not run

This problem can occur when another application is using one of the ports used by CBR. For more information, go to “Checking CBR port numbers” on page 177.

Problem: Server not responding (CBR w/WTE)

For CBR w/WTE only, if the server does not respond, the problem could be that CBR did not get started within WTE. Determine whether WTE is correctly configured on your machine. Make the following checks to resolve this problem:

- Verify the appropriate Library Path has been set according to the operating system. See “Step 2. Start WTE (for HTTP only)” on page 78, for details.
- Verify the ServerInit entry in the WTE configuration file is accurate and is located all on one line. See “Step 1. Configure WTE to use CBR (for HTTP only)” on page 76, for details.
- Verify the ClassPath is set correctly according to where Java is installed.
- If all the above are configured correctly, verify that WTE is started in the same shell environment, or command shell, in which these variables exist.
- Check WTE and CBR log files for errors: `<WTE_Install_Dir>/logs` and `<CBR_Install_Dir>/logs`.

Problem: Syntactical or configuration error

For CBR w/WTE, if the URL rule does not work, this can be a result of either a syntactical or configuration error. For this problem check the following:

- Verify the rule is configured correctly. See “Appendix D. Rule types for CBR” on page 249, for details.
- Issue a **cbrcontrol rule report** for this rule, and check the ‘Times Fired’ column to see if it has incremented according to the number of requests made. If it has incremented correctly, recheck the server configuration.
- If the rule is not being fired, add an ‘always true’ rule. Issue a **cbrcontrol rule report** on the ‘always true’ rule to verify that it is getting fired.

Problem: The cbrserver command is stopped

On a UNIX platform, this problem occurs when cbrserver is used to load balance a large number of IMAP/POP3 client requests and the system's limit on file descriptors is too small for the number of requests that cbrserver is trying to service. The cbrserver produces the following exception and then is stopped:

```
java.rmi.RMIException: security.fd.read
```

The protocol specific proxy log file reports:

```
SocketException=java.net.SocketException: Socket closed
```

The solution is to modify the **nofiles** (AIX, Red Hat Linux) or the **open files** (Solaris) limit in the shell where cbrserver is started. Increase the nofiles limit to a reasonable number larger than the current nofiles limit. Use `ulimit -a` to display the current nofiles limit, and use `ulimit -n value` to increase the value.

Problem: Receive CBR proxy error when trying to add a port

For CBR proxy (cbrserver) only, the **cbrcontrol port add** command produces the following error message: "The proxy on cluster <cluster>, port <port> did not start." The solution is to configure the cluster address on a NIC before the proxy can be started.

Solving common problems—ISS

Problem: ISS will not run

This problem can occur when another application is using one of the ports used by ISS. For more information, go to "Checking ISS port numbers" on page 176.

Problem: Starting ISS from the Services panel produces an error

For Windows only, the error message for this problem is the following: "Could not start IBM Interactive Session Support service on \\hostname Error 2140: An internal Windows error occurred." After receiving this error message, check the iss.log file for error messages and debug the ISS configuration file according to the errors in the log file. If iss.log file is not created, then ISS was unable to find the configuration file. For ISS to find the configuration file, use the **-c** option in the 'Startup Parameters' to specify the full path of the configuration file. For information on setting options for the 'Startup Parameters', see "ISS GUI" on page 144. For information on 'start iss' command options see, "Appendix C. Command reference for ISS" on page 235

Problem: Unable to get ISS GUI to "Connect to Host"

The error message that results is "Error communicating with server." The solution to this problem is to verify that the ISS daemon is running by looking at the log file or process list. To use the ISS GUI, the iss command must be

issued with the -g option. For Windows, the iss command should be issued with both the -g (GUI) and the -i (interactive) options if started from the command prompt.

Problem: All interactive sessions for ISS are assigned to one machine

This problem may occur due to one of the following reasons:

- One machine has a consistently light load relative to other machines in the cell. This may occur because other machines in the cell are supporting some other activity that is not under the control of ISS.
- There may be a problem with ISS configuration or file permissions, which is preventing ISS from operating satisfactorily.

If this condition does occur, try measuring the loads on the servers manually, and verify that ISS is failing to pick the best one.

Check that the load on the machines is not skewed to the extent that the metric being used to load-balance across the cell would result in the same machine being selected every time. For Unix-based operating systems, you can do this by looking at the amount of idle time each CPU is experiencing (using **vmstat** or **sar**) or by looking at the list of active processes (using **ps**). For Windows platforms, from the **Task Manager**, select **Performance** tab to view CPU usage and select **Processes** tab to view list of active processes. You can test a CUSTOM metric by manually running it on the server.

Also, turn on **LogLevel Trace** to determine what ISS is doing. See “Using ISS trace and debug facilities” on page 184 for details.

Problem: ISS trace shows “Cannot rename 'file 1' to 'file 2'. Permission denied.”

Ensure that ISS is being run by the administrator (for Windows) or the root user (for AIX or Solaris). For AIX, also check the permissions on the files named in the error message.

Problem: ISS is not recognising the isscontrol or GUI commands

Isscontrol commands, which can also be issued from the GUI, are always sent to the localhost on which they are being executed. Be sure that ISS is running on this node. If an InternalNet is defined and it is not the localhost, then ISS will not accept the isscontrol commands or GUI updates. There is a parameter that can be added to the isscontrol or ndadmin script to force it to send the command to the InternalNet address defined.

For isscontrol, edit the PARM2 line in the isscontrol file to the following:
PARM2='-DISSCONTROL_DEBUG=FALSE -DISSCONTROL_HOST=address'
where address is the defined InternalNet address.

For the GUI, edit ndadmin and add the following to the END_ACCESS parameter: -DISSCONTROL_HOST=*address* where *address* is the defined InternalNet address.

Problem: Log file reports “Config: Could not find local node in the config file”

ISS is unable to find the localhost defined in the configuration file. The localhost address must be defined by either the Node or InternalNet address. If there are more than two addresses on the machine and the localhost is not one that is being used in the ISS configuration, then define the localhost as the Node and use an Interface to define the additional desired address.

Problem: ISS is sending loads to Dispatcher, but Dispatcher is reporting the wrong values

There could be several things wrong in this scenario. The following are steps for solving the problem:

- Try increasing the Dispatcher manager loglevel to 5 for 30 to 60 seconds, then change it back to loglevel 1. Look in the Manager.log for a “MetricTable” entry. This table reports the address and load that Dispatcher is receiving from ISS. If the values that Dispatcher reports in the log are not the same as the values that ISS reports it is sending, then check to see if more than one ISS process is running on the ISS monitor node.
- If Dispatcher is reporting the right loads, according to ISS, but those loads are not what the manager report is displaying, then verify that the server addresses that ISS is reporting on match the server addresses defined in Dispatcher. ISS and Dispatcher have no way of communicating that the addresses defined in the ISS configuration are the same machine (just different ip) as the ones defined in Dispatcher.
- If Dispatcher is reporting the failed loads from ISS, then check the ISS log on the monitor node to see what loads it is trying to report to the Dispatcher. If the ISS monitor is receiving a failed load from the servers, look for a message (at debug loglevel) in the servers ISS log that states “Sending FAILED loadInfo packet, avail”. This message indicates that the load calculated for this server is not within the MetricNormaliztion or MetricLimits range.

Using ISS trace and debug facilities

If the ISS component does not appear to be working as you would expect, then try enabling the **Trace** or **Debug** facilities available with the **LogLevel** keyword. It is recommended that you first try to resolve problems using lower values. After examining the output produced by this parameter, if you cannot determine the cause of the problem, consider using loglevel=4. This parameter generates significantly more output than the trace parameter.

The syntax for the **LogLevel** keyword is:

LogLevel 0-4

where:

- 0 Provides no debugging data and higher numbers provide increasing amounts of detail

Note: LogLevel 0 is invalid in the configuration file.

List of some common debug statements

1999/11/14 20:51:05 Monitor: All nodes for service MyService have failed

- Look for 'failed' [lowercase]. All assigned nodes for a service are down. If this is the first time it failed, then the Alarm command would be triggered (if configured).

1999/11/14 20:51:05 monitor.ibm.com: Received recommendation of <none> for service MyCell/MyService

- Look for 'none'. This is related to the previous message. If all nodes are down then ISS will recommend 'none' as the response to queries.

1999/11/14 20:51:06 Monitor: Load updated for 9.37.64.144/MyResource: -1.000000

- Look for *resourcename*:. This simply shows what an agent is sending to the monitor for a particular resource. The resource might be used in different services. It doesn't guarantee anything except what the agent is sending. Look at the lines surrounding these messages to determine what service it applies to.

1999/11/14 20:52:35 Monitor: Marking server 9.37.64.144 as FAILED

- Look for 'FAILED' [uppercase]. This message appears when ISS marks a server down due to missed communication between agent and monitor. This relates to the HeartbeatsToNodeFail timeout (i.e. since last LoadInfo packet from agent). If the above says 'UNREACHABLE', then the agent has missed enough ICMP heartbeats and passed the HeartbeatsToNetFail timeout.

1999/11/14 20:52:35 dispatch: 2541-327 Invalid service configuration

- Invalid service configuration indicates there are no nodes associated with a service. This message will appear until a Node is assigned to the service. If ISS returns 0.0.0.0, use isscontrol to add nodes to the service.

1999/11/14 20:51:50 monitor.ibm.com: Received recommendation of 9.37.64.144 for service MyCell/MyService

- To check what ISS is recommending, look for 'recommendation'. This only applies for ISSNameServer services. If it's updating Dispatcher, it doesn't matter what ISS recommends since Dispatcher makes the choice.

Appendix A. How to read a syntax diagram

The syntax diagram shows you how to specify a command so that the operating system can correctly interpret what you type. Read the syntax diagram from left to right and from top to bottom, following the horizontal line (the main path).

Symbols and punctuation

The following symbols are used in syntax diagrams:

Symbol	Description
▶▶	Marks the beginning of the command syntax.
◀◀	Marks the end of the command syntax.

You must include all punctuation such as colons, quotation marks, and minus signs that are shown in the syntax diagram.

Parameters

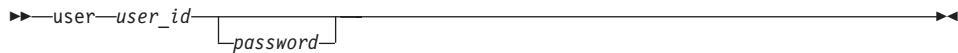
The following types of parameters are used in syntax diagrams.

Parameter	Description
Required	Required parameters are displayed on the main path.
Optional	Optional parameters are displayed below the main path.

Parameters are classified as keywords or variables. Keywords are displayed in lowercase letters and can be entered in lowercase. For example, a command name is a keyword. Variables are italicized and represent names or values you supply.

Syntax examples

In the following example, the user command is a keyword. The required variable is *user_id*, and the optional variable is *password*. Replace the variables with your own values.

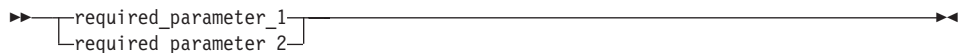


Required keywords: required keywords and variables appear on the main path line.

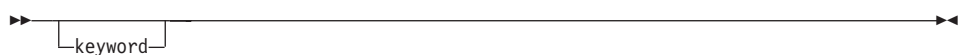


You must code required keywords and values.

Choose one required item from a stack: If there is more than one mutually exclusive required keyword or variable to choose from, they are stacked vertically in alphanumeric order.



Optional values: Optional keywords and variables appear below the main path line.

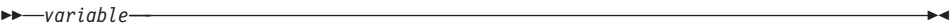


You can choose not to code optional keywords and variables.

Choose one optional keyword from a stack: If there is more than one mutually exclusive optional keyword or variable to choose from, they are stacked vertically in alphanumeric order below the main path line.



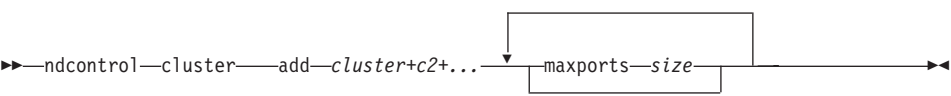
Variables: A word in all italics is a *variable*. Where you see a variable in the syntax, you must replace it with one of its allowable names or values, as defined in the text.



Nonalphanumeric characters: If a diagram shows a character that is not alphanumeric (such as colons, quotes, or minus signs), you must code the character as part of the syntax. In this example, you must code *cluster:port*.



Repeat If a diagram shows an arrow it means you have a choice to repeat that operand or get out of the text.



Appendix B. Command reference for Dispatcher and CBR

This appendix describes how to use the following Dispatcher `ndcontrol` commands and the `cbrcontrol` commands for CBR. CBR uses a subset of these commands.

- “Configuration differences between CBR and Dispatcher”
- “`ndcontrol` advisor — control the advisor” on page 193
- “`ndcontrol` cluster — configure clusters” on page 197
- “`ndcontrol` executor — control the executor” on page 200
- “`ndcontrol` file — manage configuration files” on page 204
- “`ndcontrol` help — display or print help for this command” on page 206
- “`ndcontrol` highavailability — control high availability” on page 207
- “`ndcontrol` host — configure a remote machine” on page 211
- “`ndcontrol` log — control the binary log file” on page 212
- “`ndcontrol` manager — control the manager” on page 213
- “`ndcontrol` port — configure ports” on page 219
- “`ndcontrol` rule — configure rules” on page 223
- “`ndcontrol` server — configure servers” on page 228
- “`ndcontrol` set — configure server log” on page 231
- “`ndcontrol` status — display whether the manager and advisors are running” on page 232
- “`ndcontrol` subagent — configure SNMP subagent” on page 233

You can enter a minimized version of the `ndcontrol` command parameters. You only need to enter the unique letters of the parameters. For example, to get help on the file save command, you can enter **`ndcontrol he f`** instead of **`ndcontrol help file`**.

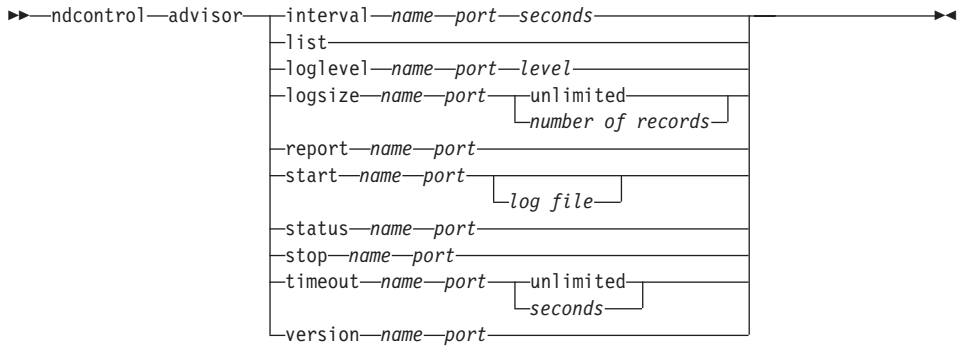
Configuration differences between CBR and Dispatcher

If you are running CBR with WTE or CBR proxy (using `cbrserver`), you can configure via the command line or the GUI. The CBR configuration interface is for the most part a subset of the interface of Dispatcher. Use the **`cbrcontrol`** command instead of `ndcontrol` to configure the CBR component. Some of the commands that are omitted in CBR are listed below.

1. High Availability
2. Subagent
3. executor report

- start
 - stop
 - set nfa <value>
 - set fincount <value>
 - set fintimeout <value>
 - set porttype <value>
4. cluster report {c}
 - set {c} porttype
 5. port add {c:p} porttype
 6. port set {c:p} porttype
 7. add {c:p:r} type port
 8. server add {c:p:s} router
 9. server set {c:p:s} router

ndcontrol advisor — control the advisor



interval

Set how often the advisor will query the servers for information.

name

The name of the advisor. Possible values include **http**, **ftp**, **ssl**, **smtp**, **imap**, **pop3**, **nntp**, and **telnet**. Names of customized advisors are of the format **xxxx**, where **ADV_xxxx** is the name of the class that implements the custom advisor.

port

The number of the port that the advisor is monitoring.

seconds

A positive number representing the number of seconds between requests to the servers about their current status. The default is 7.

list

Show list of advisors that are currently providing information to the manager.

loglevel

Set the logging level for an advisor log.

level

The number of the level (0 to 5). The higher the number, the more information that is written to the advisor log. The default is 1.

logsize

Set the maximum size of an advisor log. When you set a maximum size for the log file, the file will wrap; when the file reaches the specified size, the subsequent entries will be written from the top of the file, overwriting the previous log entries. Log size cannot be set smaller than the current size of the log. Log entries are time-stamped so you can tell the order in which they were written. The higher you set the log level, the more

carefully you should choose the log size, because you can quickly run out of space when logging at the higher levels.

number of records

The maximum size in bytes for the advisor log file. You can specify either a positive number greater than zero, or the word **unlimited**. The log file may not reach the exact maximum size before overwriting because the log entries themselves vary in size. The default value is unlimited.

report

Display a report on the state of the advisor.

start

Start the advisor. There are advisors for each protocol. The default ports are as follows:

Advisor Name	Protocol	Port
ftp	FTP	21
telnet	Telnet	23
smtp	SMTP	25
http	HTTP	80
imap	IMAP	143
pop3	POP3	110
nntp	NNTP	119
ssl	SSL	443
WLM	private	10,007
WTE	HTTP	80
PING	PING	0

log file

File name to which the management data is logged. Each record in the log will be time—stamped.

The default file is *advisorname_port.log*, for example, **http_80.log**. To change the directory where the log files will be kept, see “Changing the log file paths” on page 160.

Set the manager proportions to ensure that the advisor information is used.

status

Display the current status of all the values in an advisor that can be set globally and their defaults.

stop

Stop the advisor.

timeout

Set the number of seconds for which the manager will consider information from the advisor as valid. If the manager finds that the advisor information is older than this timeout period, the manager will not use that information in determining weights for the servers on the port the advisor is monitoring. An exception to this timeout is when the advisor has informed the manager that a specific server is down. The manager will use that information about the server even after the advisor information has timed out.

seconds

A positive number representing the number of seconds or the word **unlimited**. The default value is unlimited.

version

Display the current version of the advisor.

Examples

- To set the interval for the FTP (port 21 in this case) advisor to 6 seconds:
`ndcontrol advisor interval ftp 21 6`
- To display the list of advisors currently providing information to the manager:
`ndcontrol advisor list`

This command produces output similar to:

	ADVISOR		PORT		TIMEOUT	
	http		80		unlimited	
	ftp		21		unlimited	

- To change the log level of the advisor log to 0 for better performance:
`ndcontrol advisor loglevel http 80 0`
- To change the advisor log size to 5000 bytes:
`ndcontrol advisor logsize ftp 21 5000`
- To display a report on the state of the advisor:
`ndcontrol advisor report ftp 21`

This command produces output similar to:

	ADVISOR:	Ftp				
	PORT:	21				

	CLUSTER		SERVER		LOAD	

	9.67.131.18		9.67.129.230		8	
	9.67.131.18		9.67.131.215		-1	

- To start the advisor with the ftpadv.log file:
ndcontrol advisor start ftp 21 ftpadv.log
- To display the current status of values associated with the SSL advisor:
ndcontrol advisor status ssl 443

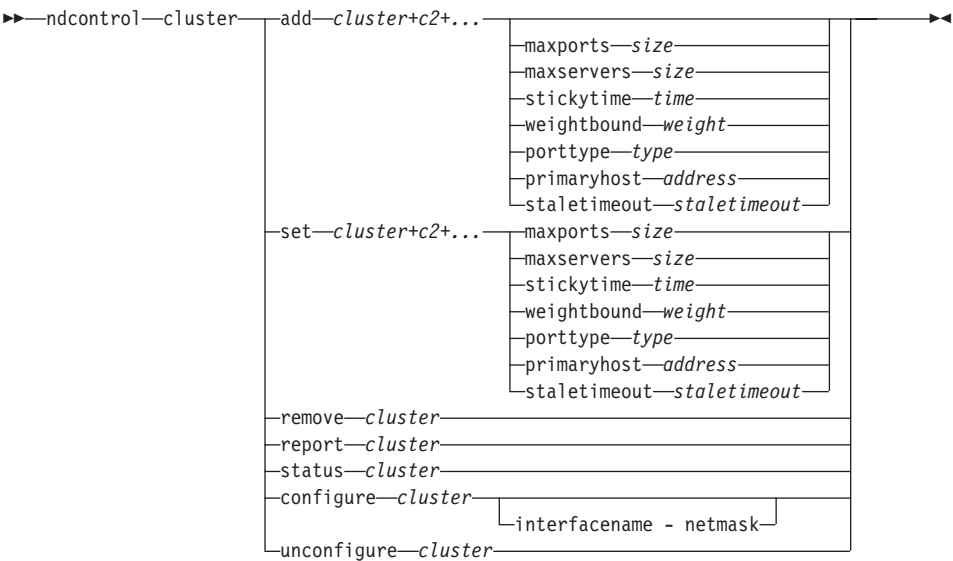
This command produces output similar to the following:

Advisor Status:

```
-----
Interval (seconds) ..... 7
Advisor Log File Name ..... Ssl_443.log
Logging Level ..... 1
Log Size ..... Unlimited
```

- To stop the http advisor at port 80:
ndcontrol advisor stop http 80
- To set the timeout value for advisor information to 5 seconds:
ndcontrol advisor timeout ftp 21 5
- To find out the current version number of the ssl advisor:
ndcontrol advisor version ssl 443

ndcontrol cluster — configure clusters



add

Add this cluster. You must define at least one cluster.

cluster

The address of the cluster as either a symbolic name or in dotted-decimal format. A cluster address value of 0.0.0.0 can be used to specify a wildcard cluster.

Note: Additional clusters are separated by a plus sign (+).

maxports

The maximum number of ports. The default value of maxports is 8.

size

The number of ports allowed.

maxservers

The default maximum number of servers per ports. This may be overridden for individual ports using **port maxservers**. The default value of maxservers is 32.

size

The number of servers allowed on a port.

stickytime

The default stickytime for ports to be created. This may be overridden for individual ports using **port stickytime**. The default value of stickytime is 0.

time

The value of stickytime.

weightbound

The default port weight bound. This may be overridden for individual ports using **port weightbound**. The default value of weightbound is 20.

weight

The value of weightbound.

porttype

The default port type. This may be overridden for individual ports using **port porttype**.

Note: Porttype does not apply to CBR.

type

Possible values are **tcp**, **udp**, and **both**.

primaryhost

The NFA address of this Dispatcher machine or the NFA address of the backup Dispatcher machine. In a mutual high availability configuration, a cluster is associated with either the primary or the backup machine.

If you change the primaryhost of a cluster once the primary and backups are already started and running mutual high availability, you also must force the new primary host to takeover. And, you need to update the scripts and manually unconfigure and configure the cluster correctly. See “Mutual high availability” on page 49 for more information.

address

The address value of the primaryhost. The default is the NFA address of this machine.

staletimeout

The number of seconds during which there can be no activity on a connection before that connection is removed. The default for FTP is 900; the default for Telnet is 32,000,000. The default for all other protocols is 30. This may be overridden for individual ports using **port staletimeout**.

set

Set the properties of the cluster.

remove

Remove this cluster.

report

Show the internal fields of the cluster.

Note: Report does not apply to CBR.

status

Show current status of a specific cluster.

configure

Configures a cluster alias to the network interface card.

Note: Configure does not apply to CBR.

interfacename netmask

It is required if it is a different alias from what Dispatcher first finds.

unconfigure

Deletes the cluster alias from the network interface card.

Note: Unconfigure does not apply to CBR.

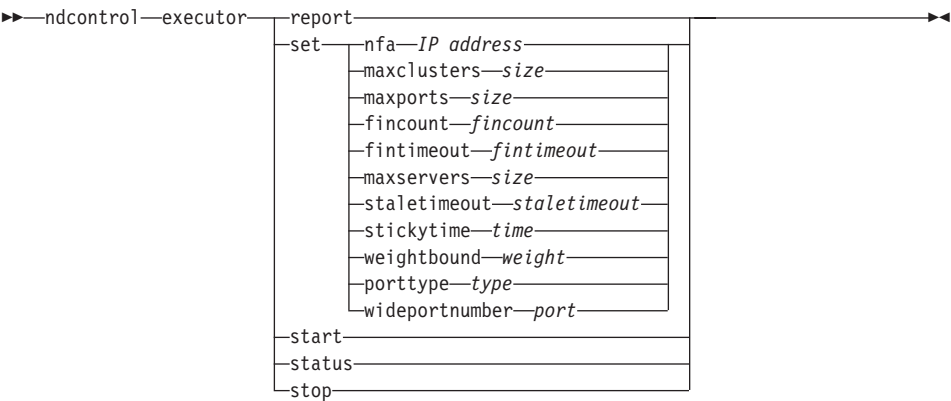
Examples

- To add cluster address 130.40.52.153:
ndcontrol cluster add 130.40.52.153
- To remove cluster address 130.40.52.153:
ndcontrol cluster remove 130.40.52.153
- To add a wildcard cluster:
ndcontrol cluster add 0.0.0.0
- For a mutual high availability configuration, set cluster address 9.6.54.12 with the NFA of the backup machine (9.65.70.19) as the primary host:
ndcontrol cluster set 9.6.54.12 primaryhost 9.65.70.19
- To show the status for cluster address 9.67.131.167:
ndcontrol cluster status 9.67.131.167

This command produces output similar to:

```
Cluster Status:
-----
Address ..... 9.67.131.167
Number of target ports ..... 3
Max Number of Ports ..... 8
Default Max Number of Servers ..... 32
Default Port Sticky Time ..... 0
Default Port Weight Bound ..... 20
Default Port Type ..... tcp/udp
Default stale time ..... 30
Primary Host Address ..... 9.67.131.167
```

ndcontrol executor — control the executor



report

Show the internal fields of the executor.

Note: Report does not apply to CBR.

set

Set the fields of the executor.

nfa

Set the nonforwarding address. Any packet sent to this address will not be forwarded by the Dispatcher machine.

Note: NFA does not apply to CBR.

IP address

The internet protocol address as either a symbolic name or in dotted decimal format.

maxclusters

The maximum number of clusters that can be configured. The default value of `maxclusters` is 100.

size

The maximum number of clusters that can be configured.

maxports

The default value of `maxports` for clusters to be created. This may be overridden by the **cluster set** or **cluster add** command. The default value of `maxports` is 8.

size

The number of ports.

fincount

The number of connections that must be in FIN state before garbage collection of connections will be initiated. The default value of fincount is 4000.

fincount

The fincount value.

Note: Fincount does not apply to CBR.

fintimeout

The number of seconds to keep a connection in memory after the connection has been put in the FIN state. The default fintimeout value is 60.

fintimeout

The fintimeout value.

Note: Fintimeout does not apply to CBR.

maxservers

The default maximum number of servers per port. This may be overridden by the **cluster** or **port** command. The default value of maxservers is 32.

size

The number of servers.

staletimeout

The number of seconds during which there can be no activity on a connection before that connection is removed. The default for FTP is 900; the default for Telnet is 32,000,000. The default for all other ports is 30. This may be overridden by the **cluster** or **port** command.

staletimeout

The staletimeout value.

stickytime

The default port sticky time value for all future ports. It may be overridden by the **cluster** or **port** command. The default stickytime value is 0.

time

The stickytime value.

weightbound

The default port weightbound value for all future ports. It may be overridden by the **cluster** or **port** command. The default weightbound value is 20.

weight

The weightbound value.

porttype

The default port porttype value for all future ports. It may be overridden by the **cluster** or **port** command.

Note: Porttype does not apply to CBR.

type

Possible values are **tcp**, **udp**, and **both**.

wideportnumber

An unused TCP port on each Dispatcher machine. The *wideportnumber* must be the same for all the Dispatcher machines. The default value of wideportnumber is 0, indicating that wide area support is not in use.

Note: Wideportnumber does not apply to CBR.

port

The value of **wideportnumber**.

start

Start the executor.

Note: Start does not apply to CBR.

status

Show the interval fields of the executor.

stop (AIX, Red Hat Linux, and Solaris only)

Stop the executor.

Note: Stop does not apply to CBR.

Examples

- To display the internal counters for Dispatcher:

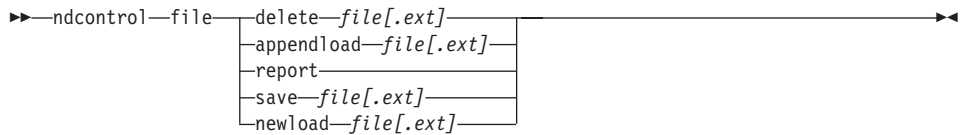
```
ndcontrol executor status
```

```
Executor Status:
```

```
-----  
Nonforwarding Address ..... 9.67.131.151  
Cluster Max Ports ..... 8  
Fin Count ..... 4,000  
Fin Time Out ..... 60  
Port Max Servers ..... 32  
Port Stale Time Out ..... 30  
Port Sticky Time ..... 0  
Port Weight Bound ..... 20  
Port Type ..... tcp/udp  
Max Number of Clusters ..... 100  
Wide area network port number .. 2,001
```

- To set the nonforwarding address to 130.40.52.167:
`ndcontrol executor set nfa 130.40.52.167`
- To set the maximum number of clusters:
`ndcontrol executor set maxclusters 4096`
- To start the executor:
`ndcontrol executor start`
- To stop the executor (**AIX, Red Hat Linux, and Solaris only**):
`ndcontrol executor stop`

ndcontrol file — manage configuration files



delete

Delete the file.

file[.ext]

A configuration file.

The file extension (*.ext*) can be anything you like and can be omitted.

appendload

Append a configuration file to the current configuration and load into the Dispatcher.

report

Report on the available file or files.

save

Save the configuration from the Dispatcher to the file.

Note: Files are saved into and loaded from **NDROOT/configurations**, where NDROOT is, by default:

- AIX: **/usr/lpp/nd/dispatcher/**
- Red Hat Linux: **/opt/nd/dispatcher**
- Solaris: **/opt/nd/dispatcher**
- Windows: **c:\Program Files\nd\dispatcher**

newload

Load a new configuration file into the Dispatcher. The new configuration file will replace the current configuration.

Examples

- To delete a file:

```
ndcontrol file delete file3
```

File (file3) was deleted.
- To load a new configuration file to replace the current configuration:

```
ndcontrol file newload file1.sv
```

File (file1.sv) was loaded into the Dispatcher.
- To append a configuration file to the current configuration and load:

```
ndcontrol file appendload file2.sv
```

File (file2.sv) was appended to the current configuration and loaded.

- To view a report of your files (that is, those files that you saved earlier):

```
ndcontrol file report
```

```
FILE REPORT:
```

```
file1.save
```

```
file2.sv
```

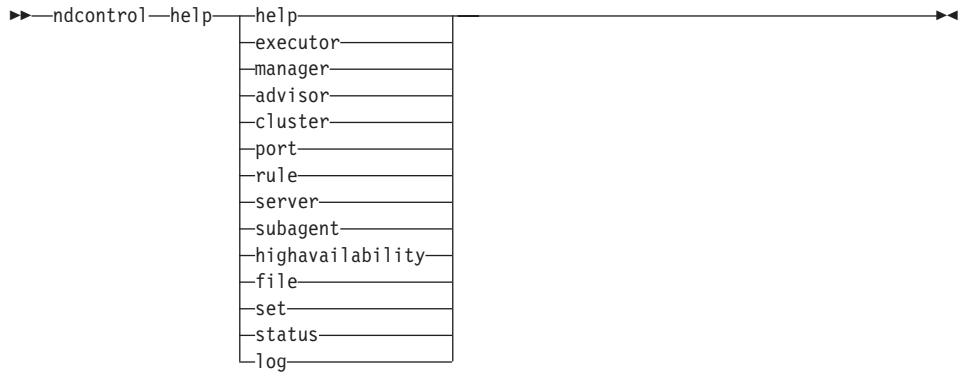
```
file3
```

- To save your configuration into a file named file3:

```
ndcontrol file save file3
```

The configuration was saved into file (file3).

ndcontrol help — display or print help for this command



Examples

- To get help on the ndcontrol command:
ndcontrol help

This command produces output similar to:

HELP COMMAND ARGUMENTS:

Usage: help <help option>

Example: help cluster

help	- print complete help text
executor	- help on executor command
manager	- help on manager command
advisor	- help on advisor command
cluster	- help on cluster command
port	- help on port command
rule	- help on rule command
server	- help on server command
subagent	- help on subagent command
highavailability	- help on high availability command
file	- help on file command
set	- help on set command
status	- help on status command
log	- help on log command

Notice that parameters within <> are variables.

- Sometimes the help will show choices for the variables using | to separate the options:

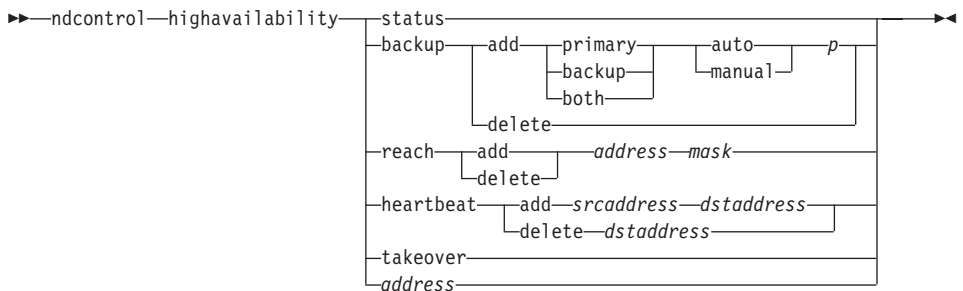
fintimeout <cluster address>|all <time>

-Change FIN timeout

(Use 'all' to change all clusters)

ndcontrol highavailability — control high availability

Note: The ndcontrol high availability syntax diagram does not apply to CBR.



status

Return a report on high availability. Machines are identified as having one of three status conditions or states:

Active A given machine (either a primary, backup, or both) is routing packets.

Standby

A given machine (either a primary, backup, or both) is not routing packets; it is monitoring the state of an **active** Dispatcher.

Idle A given machine is routing packets, and is not trying to establish contact with its partner Dispatcher.

In addition, the **status** keyword returns information about various substates:

Synchronized

A given machine has established contact with another Dispatcher.

Other substates

This machine is trying to establish contact with its partner Dispatcher but has not yet succeeded.

backup

Specify information for either the primary or backup machine.

add

Defines the high availability functions for this machine.

primary

Identifies the Dispatcher machine that has a *primary* role.

backup

Identifies the Dispatcher machine that has a *backup* role.

both

Identifies the Dispatcher machine that has *both* a primary and backup role. This is the mutual high availability feature in which primary and backup roles are associated on a per cluster set basis. See “Mutual high availability” on page 49, for more information.

auto

Specifies an *automatic* recovery strategy, in which the primary machine will resume routing packets as soon as it comes back into service.

manual

Specifies a *manual* recovery strategy, in which the primary machine does not resume routing packets until the administrator issues a **takeover** command.

p[ort]

An unused TCP port on both machines, to be used by Dispatcher for its heartbeat messages. The *portnum* must be the same for both the primary and backup machines.

delete

Removes this machine from high availability, so that it will no longer be used as a backup or primary machine.

reach

Add or delete target address for the primary and backup Dispatchers, the reach advisor send out *pings* from both the backup and the primary Dispatchers to determine how reachable their targets are.

add

Adds a target address for the reach advisor.

delete

Removes a target address from the reach advisor.

address

IP address (dotted-decimal or symbolic) of the target node.

mask

A subnet mask.

heartbeat

Defines the communication session between the primary and backup Dispatcher machines.

add

Tell the source Dispatcher the address of its partner (destination address).

delete

Removes the destination address from the heartbeat information.

srcaddress

Source address. The address (IP or symbolic) of this Dispatcher machine.

dstaddress

Destination address. The address (IP or symbolic) of the other Dispatcher machine.

Note: For mutual high availability, the *srcaddress* and *dstaddress* must be the NFAs of the machines for at least one heartbeat pair.

takeover

Simple high availability configuration (role of the Dispatcher machines are either *primary* or *backup*):

- Takeover instructs a standby Dispatcher to become active and to begin routing packets. This will force the currently active Dispatcher to become standby. The takeover command must be issued on the standby machine and works only when the strategy is **manual**. The substate must be *synchronized*.

Mutual high availability configuration (role of each Dispatcher machine is *both*):

- The Dispatcher machine with the mutual high availability feature contains two clusters which match its partner's. One of the clusters is considered the primary cluster (the partner's backup cluster), and the other is the backup cluster (the partner's primary cluster). Takeover instructs the Dispatcher machine to begin routing packets for the other machine's cluster(s). The takeover command can only be issued when the cluster(s) of the Dispatcher machine are in *standby* state and the substate is *synchronized*. This will force the partner's currently active cluster(s) to change to standby state. The takeover command works only when the strategy is **manual**. See "Mutual high availability" on page 49, for more information.

Notes:

1. Note that the *roles* of the machines (*primary*, *backup*, *both*) do not change. Only their relative *status* (*active* or *standby*) changes.
2. There are three possible takeover *scripts*: *goActive*, *goStandby*, and *goInOp*. See "Using scripts" on page 92.

address

The takeover address value is optional. It should only be used when the role of the machine is *both* primary and backup (mutual high availability configuration). The address specified is the NFA of the Dispatcher machine which normally routes this cluster's traffic. When there is a takeover of both clusters, specify the Dispatcher's own NFA address.

Examples

- To check the high availability status of a machine:
`ndcontrol highavailability status`

Output:

High Availability Status:

```
-----  
Role .....primary  
Recovery Strategy ..... manual  
State ..... Active  
Sub-state..... Synchronized  
Primary host..... 9.67.131.151  
Port .....12,345  
Preferred Target..... 9.67.134.223
```

Heartbeat Status:

```
-----  
Count ..... 1  
Source/Destination ... 9.67.131.151/9.67.134.223
```

Reachability Status:

```
-----  
Count ..... 1
```

- To add the backup information to the primary machine using the automatic recovery strategy and port 80:
`ndcontrol highavailability backup add primary auto 80`
- To add an address that the Dispatcher must be able to reach:
`ndcontrol highavailability reach add 9.67.125.18`
- To add heartbeat information for the primary and backup machines.
Primary - `highavailability heartbeat add 9.67.111.3 9.67.186.8`
Backup - `highavailability heartbeat add 9.67.186.8 9.67.111.3`
- To tell the standby Dispatcher to become active, forcing the active machine to become standby:
`ndcontrol highavailability takeover`

ndcontrol host — configure a remote machine

►►—ndcontrol—host:—*remote_host*—◄◄

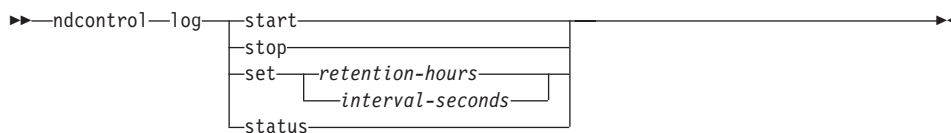
remote_host

The name of the remote Network Dispatcher machine being configured. When typing this command, make sure there is no space between **host:** and *remote_host*, for example:

```
ndcontrol host:remote_name
```

After this command has been issued on the command prompt, enter any valid ndcontrol command you want issued to the remote Network Dispatcher machine.

ndcontrol log — control the binary log file



start

Starts the binary log.

stop

Stops the binary log.

set

Sets fields for binary logging.

retention

The number of hours that binary log files will be kept. The default value for retention is 24.

hours

The number of hours.

intervals

The number of seconds between log entries. The default value for interval is 60.

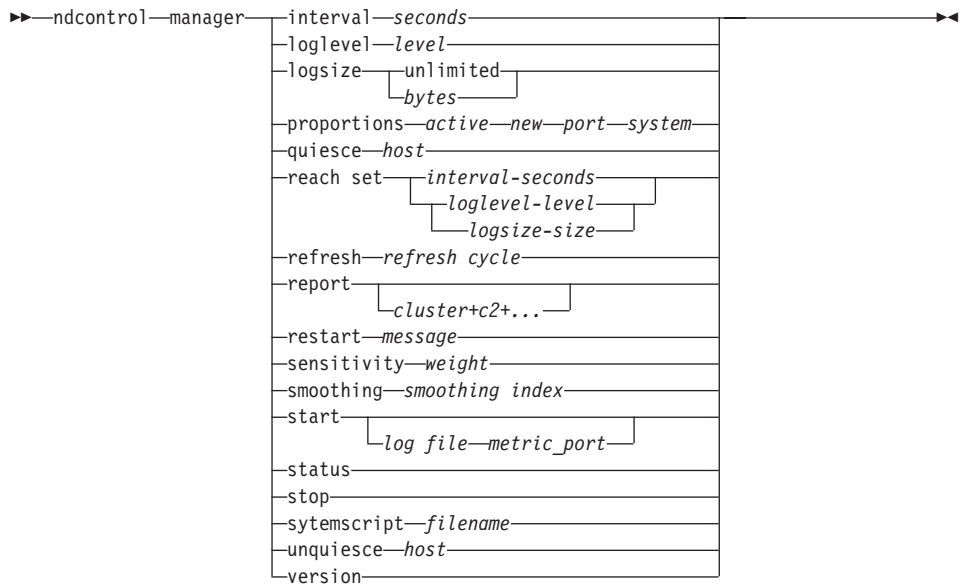
seconds

The number of seconds.

status

Shows the retention and intervals of the binary log.

ndcontrol manager — control the manager



interval

Set how often the manager will update the weights of the servers to the executor, updating the criteria that the executor uses to route client requests.

seconds

A positive number representing in seconds how often the manager will update weights to the executor. The default is 2.

loglevel

Set the logging level for the manager log.

level

The number of the level (0 to 5). The higher the number, the more information that is written to the manager log. The default is 1.

logsize

Set the maximum size of the manager log. When you set a maximum size for the log file, the file will wrap; when the file reaches the specified size, the subsequent entries will be written from the top of the file, overwriting the previous log entries. Log size cannot be set smaller than the current size of the log. Log entries are time stamped so you can tell the order in which they were written. The higher you set the log level, the more carefully you should choose the log size, because you can quickly run out of space when logging at the higher levels.

bytes

The maximum size in bytes for the manager log file. You can specify either a positive number greater than zero, or the word **unlimited**. The log file may not reach the exact maximum size before overwriting because the log entries themselves vary in size. The default value is unlimited.

proportions

Set the proportion of importance for active connections (*active*), new connections (*new*), information from any advisors (*port*), and information from a system monitoring program such as ISS (*system*) that are used by the manager to set server weights. Each of these values, described below, is expressed as a percentage of the total and they therefore always total 100.

active

A number from 0–100 representing the proportion of weight to be given to the active connections. The default is 50.

new

A number from 0–100 representing the proportion of weight to be given to the new connections. The default is 50.

port

A number from 0–100 representing the proportion of weight to be given to the information from advisors. The default is 0.

system

A number from 0–100 representing the proportion of weight to be given to the information from the system metrics, such as from ISS. The default is 0.

quiesce

Specify no more connections be sent to a server. The manager sets the weight for that server to 0 in every port to which it is defined. Use this command if you want to do some quick maintenance on a server and then unquiesce it. If you delete a quiesced server from the configuration and then add it back, it will not retain its status prior to being quiesced.

host

The IP address of the server as either a symbolic name or in dotted decimal format.

reach set

Sets the interval, loglevel, and logsize for the reach advisor.

refresh

Set the number of intervals before querying the executor for a refresh of information about new and active connections.

refresh cycle

A positive number representing the number of intervals. The default is 2.

report

Display a statistics snapshot report.

cluster

The address of the cluster you want displayed in the report. The address can be either a symbolic name or in dotted-decimal format. The default is a manager report display for all the clusters.

Note: Additional clusters are separated by a plus sign (+).

restart

Restart all servers (that are not down) to normalized weights (1/2 of maximum weight).

message

A message that you want written to the manager log file.

sensitivity

Set minimum sensitivity to which weights update. This setting defines when the manager should change its weighting for the server based on external information.

weight

A number from 0 to 100 to be used as the weight percentage. The default of 5 creates a minimum sensitivity of 5%.

smoothing

Set an index that smooths the variations in weight when load balancing. A higher smoothing index will cause server weights to change less drastically as network conditions change. A lower index will cause server weights to change more drastically.

index

A positive floating point number. The default is 1.5.

start

Start the manager.

log file

File name to which the manager data is logged. Each record in the log will be time stamped.

The default file will be installed in the **logs** directory. See “Appendix E. Sample configuration files” on page 251. To change the directory where the log files will be kept, see “Changing the log file paths” on page 160.

metric_port

Port that ISS will use to report system loads. If you specify a metric port, you must specify a log file name. The default metric port is 10004.

status

Display the current status of all the values in the manager that can be set globally and their defaults.

stop

Stop the manager.

systemscript

Defines the command which the Server Monitor Agents (SMA) will issue on the server machines. SMA is a system monitor agent which is unique for the Red Hat Linux platform of IBM Network Dispatcher.

scriptname

The name of a script file or executable which provides a numerical return code (in the range of 0 to 100) indicating a system load value. Two script files are provided which run on the agent in the server machine: **cpuload** (returns the percentage of cpu in use) and **memload** (returns the percentage of memory use). **cpuload** is the default. User written script files or executables must reside in the **opt/nd/sma/scripts** directory on the server machines.

unquiesce

Specify that the manager can begin to give a weight higher than 0 to a server that was previously quiesced, in every port to which it is defined.

version

Display the current version of the manager.

Examples

- To set the updating interval for the manager to every 5 seconds:
`ndcontrol manager interval 5`
- To set the level of logging to 0 for better performance:
`ndcontrol manager loglevel 0`
- To set the manager log size to 1,000,000 bytes:
`ndcontrol manager logsize 1000000`
- To set the relative importance placed on input received by the manager:
`ndcontrol manager proportions 60 35 5 0`
- To specify that no more connections be sent to the server at 130.40.52.153:
`ndcontrol manager quiesce 130.40.52.153`
- To set the number of updating intervals before the weights will be refreshed to 3:
`ndcontrol manager refresh 3`
- To get a statistics snapshot of the manager:
`ndcontrol manager report`

This command produces output similar to:

HOST TABLE LIST	STATUS
9.67.129.221	ACTIVE
9.67.129.213	ACTIVE
9.67.134.223	ACTIVE

9.67.131.18	WEIGHT			ACTIVE % 48		NEW % 48		PORT % 4		SYSTEM % 0	
PORT: 80	NOW	NEW	WT	CONN	WT	CONN	WT	LOAD	WT	LOAD	
9.67.129.221	8	8	10	0	10	0	7	29	0	0	
9.67.134.223	11	11	10	0	10	0	12	17	0	0	
PORT TOTALS:	19	19		0		0		46		0	

9.67.131.18	WEIGHT			ACTIVE % 48			NEW % 48			PORT % 4			SYSTEM % 0		
PORT: 23	NOW	NEW	WT	CONN		WT	CONN		WT	LOAD	WT	LOAD			
9.67.129.213	10	10	10	0		10	0		10	71	0	0			
9.67.134.223	0	0	10	0		10	0		-9999	-1	0	0			
PORT TOTALS:	10	10		0			0			70		0			

ADVISOR	PORT	TIMEOUT
reach	0	unlimited
http	80	unlimited
ftp	21	unlimited

- To restart all the servers to normalized weights and write a message to the manager log file:
ndcontrol manager restart Restarting the manager to update code

This command produces output similar to:

320-14:04:54 Restarting the manager to update code

- To set the sensitivity to weight changes to 10:
ndcontrol manager sensitivity 10
- To set the smoothing index to 2.0:
ndcontrol manager smoothing 2.0

- To start the manager and specify the log file named ndmgr.log (paths cannot be set)
ndcontrol manager start ndmgr.log
- To display the current status of the values associated with the manager:
ndcontrol manager status

This command produces output similar to the following example.

Manager status:

=====

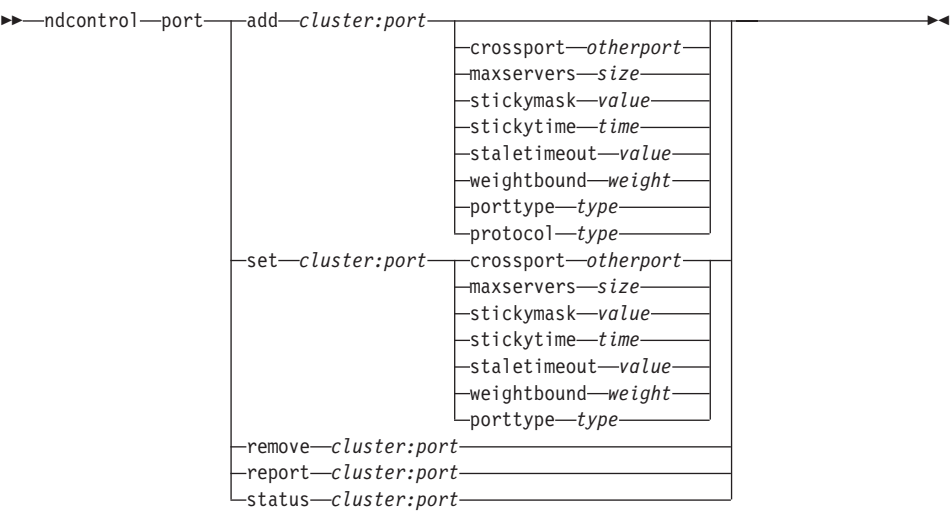
```

Metric port..... 10,004
Manager log filename..... manager.log
Manager log level..... 1
Maximum log Size..... unlimited
Sensitivity level..... 0.05
Smoothing index..... 1.5
Interval (seconds)..... 2
Weights Refresh Cycle..... 2
Active connections proportion..... 0.5
New connections proportion..... 0.5
Port specific proportion..... 0
System proportion..... 0
Reach log level..... 1
Reach maximum log size (bytes)..... unlimited
Reach interval (seconds)..... 7

```

- To stop the manager:
ndcontrol manager stop
- To specify that the manager can begin to give a weight higher than 0 to a server at 130.40.52.153 that was previously quiesced:
ndcontrol manager unquiesce 130.40.52.153
- To display the current version number of the manager:
ndcontrol manager version

ndcontrol port — configure ports



add

Add a port to a cluster. You must add a port to a cluster before you can add any servers to that port. If there are no ports for a cluster, all client requests will be processed locally. You can add more than one port at one time using this command.

cluster

The address of the cluster as either a symbolic name or in dotted-decimal format.

Note: Additional clusters are separated by a plus sign (+).

port

The number of the port. A port number value of 0 (zero) can be used to specify a wildcard port.

Note: Additional ports are separated by a plus sign (+).

crossport

Crossport allows you to expand the sticky/affinity feature across multiple ports so that client requests received on different ports can still be sent to the same server for subsequent requests. For crossport value, specify the *otherport* number for which you want to share the cross port affinity feature. In order to use this feature, the ports must:

- share the same cluster address
- share the same servers

- have the same (nonzero) stickytime value
- have the same stickymask value

To remove the crossport feature, set the crossport value back to its own port number. For more information on cross port affinity feature, see “Cross port affinity” on page 118.

Note: Crossport only applies to the Dispatcher component.

otherport

The value of crossport. The default value is the same as its own *port* number.

maxservers

The maximum number of servers. The default value of maxservers is 32.

size

The value of maxservers.

stickymask

The affinity address mask feature groups incoming client requests based on common subnet addresses. When a client request first makes a connection to the port, all subsequent requests from clients with the same subnet address (designated by that part of the IP address which is being masked) will be directed to the same server. See “Affinity address mask” on page 119, for more information.

Note: The stickymask keyword only applies to the Dispatcher component.

value

The stickymask value is the number of high-order bits of the 32-bit IP address you want to mask. Possible values are: 8, 16, 24, and 32. The default value is 32, which disables the affinity address mask feature.

stickytime

The interval between the closing of one connection and the opening of a new connection during which a client will be sent back to the same server used during the first connection. After the sticky time, the client may be sent to a server different from the first. The stickytime should be set to 1 if using the Server Directed Affinity API.

time

The port sticky time in number of seconds. Zero signifies that the port is not sticky.

staletimeout

The number of seconds during which there can be no activity on a connection before that connection is removed.

Note: For the CBR POP3 or IMAP proxy, `staletimeout` corresponds to the inactivity autologout timer for these protocols. `Staletimeout`, for CBR proxy, defaults to 60 seconds, which overrides the inactivity timeouts for POP3 and IMAP. For more information on `staletimeout` for CBR proxy, see “CBR proxy (for IMAP or POP3)” on page 70.

value

The value of **`staletimeout`** in number of seconds.

weightbound

Set the maximum weight for servers on this port. This affects how much difference there can be between the number of requests the executor will give each server. The default value is 20.

weight

A number from 1–100 representing the maximum weight bound.

porttype

The port type.

Note: Porttype does not apply to CBR.

type

Possible values are **`tcp`**, **`udp`**, and **`both`**.

protocol

The proxy protocol type.

Note: Protocol only applies to the CBR proxy (CBR without WTE). For more information, see “CBR proxy (for IMAP or POP3)” on page 70.

type

Possible values are **`POP3`** or **`IMAP`**.

set

Set the fields of a port.

remove

Remove this port.

report

Report on this port.

status

Show status of servers on this port. If you want to see the status on all ports, do not specify a *port* with this command. Don’t forget the colon, however.

Examples

- To add port 80 and 23 to a cluster address 130.40.52.153:
`ndcontrol port add 130.40.52.153:80+23`

- To add a wildcard port to a cluster address of 130.40.52.153:
ndcontrol port set 130.40.52.153:0
- For the CBR proxy, to add port 20 for POP3 protocol to a cluster address of 9.37.60.91:
cbrcontrol port add 9.37.60.91:20 protocol pop3
- To set the maximum weight of 10 to port 80 at a cluster address of 130.40.52.153:
ndcontrol port set 130.40.52.153:80 weightbound 10
- To set the stickytime value to 60 seconds for port 80 and port 23 at a cluster address of 130.40.52.153:
ndcontrol port set 130.40.52.153:80+23 stickytime 60
- To set the cross port affinity of port 80 to port 23 at a cluster address of 130.40.52.153:
ndcontrol port set 130.40.52.153:80 crossport 23
- To remove port 23 from a cluster address of 130.40.52.153:
ndcontrol port remove 130.40.52.153:23
- To get the status of port 80 at a cluster address of 9.67.131.153:
ndcontrol port status 9.67.131.153:80

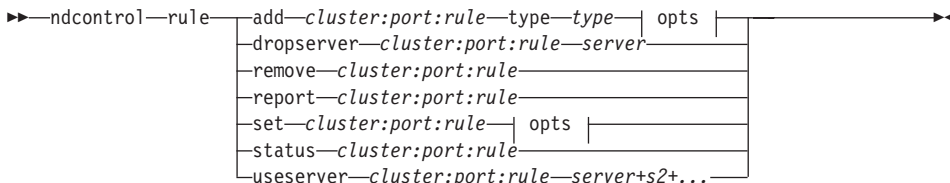
This command produces output similar to:

Port Status:

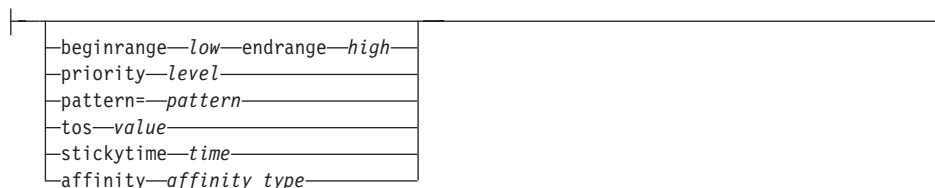
```
Port Number ..... 80
Cluster Address ..... 9.67.131.153
Number of Servers ..... 2
Stale Timeout ..... 30
Weight Bound ..... 20
Maximum Number of Servers ..... 32
Sticky Time ..... 0
Port Type ..... tcp/udp
Cross Port Affinity ..... 80
```

ndcontrol rule — configure rules

Note: The ndcontrol rule (configure rules) does not apply to CBR proxy (CBR without WTE).



opts:



add

Add this rule to a port.

cluster

The address of the cluster as either a symbolic name or in dotted-decimal format.

Note: Additional clusters are separated by a plus sign (+).

port

The number of the port.

Note: Additional ports are separated by a plus sign (+).

rule

The name you choose for the rule. This name can contain any alphanumeric character, underscore, hyphen, or period. It can be from 1 to 20 characters and cannot contain any blanks.

Note: Additional rules are separated by a plus sign (+).

type

The type of rule.

type

Your choices for *type* are:

ip The rule is based on the client IP address.

time The rule is based on the time of day.

connection

The rule is based on the number of connections per second for the port. This rule will work only if the manager is running.

active The rule is based on the number of active connections total for the port. This rule will work only if the manager is running.

port The rule is based on the client port.

Note: Port does not apply to CBR.

true This rule is always true. Think of it as an else statement in programming logic.

service

This rule is based on the type of service (TOS) byte field in the IP header.

Note: Service only applies to the Dispatcher component.

content

This rule describes a regular expression which will be compared to the client requested URLs. This is only valid in CBR.

beginrange

The lower value in the range used to determine whether or not the rule is true.

low

Depends on the type of rule. The kind of value and its default are listed here by the type of rule:

ip The address of the client as either a symbolic name or in dotted-decimal format. The default is 0.0.0.0.

time An integer. The default is 0, representing midnight.

connection

An integer. The default is 0.

active An integer. The default is 0.

port An integer. The default is 0.

endrange

The higher value in the range used to determine whether or not the rule is true.

high

Depends on the type of rule. The kind of value and its default are listed here by the type of rule:

ip The address of the client as either a symbolic name or in dotted-decimal format. The default is 255.255.255.254.

time An integer. The default is 24, representing midnight.

Note: When defining the beginrange and endrange of time intervals, note that each value must be an integer representing only the hour portion of the time; portions of an hour are not specified. For this reason, to specify a single hour—say, the hour between 3:00 and 4:00 am— you would specify a beginrange of **3** and an endrange also of **3**. This will signify all the minutes beginning with 3:00 and ending with 3:59. Specifying a beginrange of **3** and an endrange of **4** would cover the two-hour period from 3:00 through 4:59.

connections

An integer. The default is 2 to the 32nd power minus 1.

active An integer. The default is 2 to the 32nd power minus 1.

port An integer. The default is 65535.

priority

The order in which the rules will be reviewed.

level

An integer. If you do not specify the priority of the first rule you add, Dispatcher will set it by default to 1. When a subsequent rule is added, by default its priority is calculated to be 10 + the current lowest priority of any existing rule. For example, assume you have an existing rule whose priority is 30. You add a new rule and set its priority at 25 (which, remember, is a *higher* priority than 30). Then you add a third rule without setting a priority. The priority of the third rule is calculated to be 40 (30 + 10).

pattern

Specifies the pattern to be used for a content type rule.

pattern

The pattern to be used. For more information, see “Appendix D. Rule types for CBR” on page 249.

Note: Pattern does not apply to Dispatcher.

tos

Specifies the “type of service” (TOS) value used for the **service** type rule.

Note: TOS only applies to the Dispatcher component.

value

The 8 character string to be used for the tos value, where valid characters are: 0 (binary zero), 1 (binary one), and x (don't care). For example: 0xx1010x. For more information, see "Using rules based on type of service (TOS)" on page 111.

stickytime

Specifies the stickytime to be used for a content type rule (CBR only). This option is only valid in the set portion of the rule.

time

Time in seconds.

affinity

Specifies the affinity type to be used for a content type rule (CBR only). This option is only valid in the set portion of the rule.

affinity_type

Possible values are: *clientip* (default) or *cookie*.

dropserver

Remove a server from a rule set.

server

The IP address of the TCP server machine as either a symbolic name or in dotted-decimal format.

Note: Additional servers are separated by a plus sign (+).

remove

Remove one or more rules, separated from one another by plus signs.

report

Display the internal values of one or more rules.

set

Set values for this rule.

status

Display the settable values of one or more rules.

useserver

Insert servers into a rule set.

Examples

- To add a rule that will always be true, do not specify the beginning range or end range.
ndcontrol rule add 9.37.67.100:80:trule type true priority 100
- To create a rule forbidding access to a range of IP addresses, in this case those beginning with "9."

```
ndcontrol rule add 9.37.131.153:80:ni type ip b 9.0.0.0 e 9.255.255.255
```

- To create a rule that will specify the use of a given server from the hour of 11:00 a.m. through the hour of 3:00 p.m.:

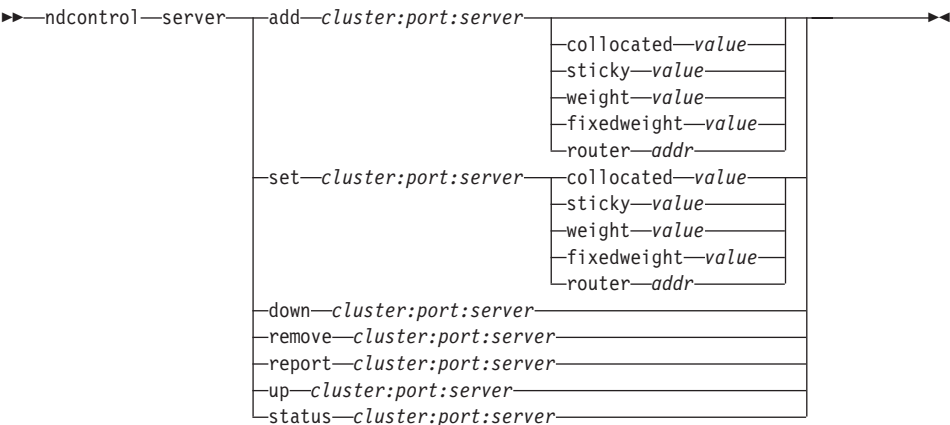
```
ndcontrol rule add cluster1:80:timerule type time beginrange 11 endrange 14
```

```
ndcontrol rule useserver cluster1:80:timerule server05
```

- To create a rule based on the content of the TOS byte field in the IP header:

```
ndcontrol rule add 9.67.131.153:80:tosrule type service tos 0xx1001x
```

ndcontrol server — configure servers



- add**
Add this server.
- cluster*
The address of the cluster as either a symbolic name or in dotted-decimal format.

Note: Additional clusters are separated by a plus sign (+).
- port*
The number of the port.

Note: Additional ports are separated by a plus sign (+).
- server*
The IP address of the TCP server machine as either a symbolic name or in dotted-decimal format.

Note: Additional servers are separated by a plus sign (+).
- collocated**
Collocated allows you to specify if the Network Dispatcher is installed on one of the server machines it is load balancing. The collocated option does not apply to the Windows platform, which cannot support collocated servers.

Note: Collocated applies to the Dispatcher component only.
- value*
Value of collocated: yes or no. Default is no.

sticky

Allows a server to override the stickytime setting on its port. With a default value of “yes,” the server retains the normal affinity as defined at the port. With a value of “no,” the client will *not* return to that server the next time it issues a request on that port regardless of the stickytime setting of the port. This is useful in certain situations when you are using rules. For more information, see “Rule affinity override” on page 119.

value

Value of sticky: yes or no. Default is yes.

weight

A number from 0–100 representing the weight for this server. Setting the weight to zero will prevent any new requests from being sent to the server, but will not end any currently active connections to that server. The default is one-half the specified port’s maximum weight. If the manager is running, this setting will be quickly overwritten.

value

Value of the weight.

fixedweight

The fixedweight option allows you to specify whether you want the manager to modify the server weight or not. If you set the fixedweight value to yes, when the manager runs it will not be allowed to modify the server weight. For more information, see “Manager fixed weights” on page 86.

value

Value of fixedweight: yes or no. Default is no.

router

The address of the router to the remote server. Default is 0, indicating a local server. Note that once a server’s router address is set to something other than zero (indicating a remote server), it cannot be reset to 0 to make the server local again. Instead, the server must be removed, then added again without a router address being specified. Similarly, a server defined as local (router address = 0) cannot be made remote by changing the router address. The server must be removed and added again.

Note: Router does not apply for CBR.

addr

Value of the address of the router.

down

Mark this server down. This command breaks all active connections to that server and prevents any other connections or packets from being sent to that server.

remove

Remove this server.

report

Report on this server.

set

Set values for this server.

status

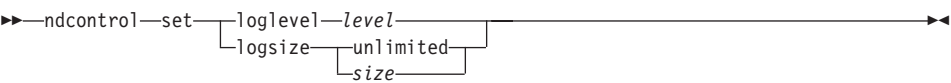
Show status of the servers.

up Mark this server up. Dispatcher will now send new connections to that server.

Examples

- To add the server at 27.65.89.42 to port 80 on a cluster address 130.40.52.153:
`ndcontrol server add 130.40.52.153:80:27.65.89.42`
- To set the server at 27.65.89.42 as nonsticky (rule affinity override feature):
`ndcontrol server set 130.40.52.153:80:27.65.89.42 sticky no`
- To mark the server at 27.65.89.42 as down:
`ndcontrol server down 130.40.52.153:80:27.65.89.42`
- To remove the server at 27.65.89.42 on all ports on all clusters:
`ndcontrol server remove ::27.65.89.42`
- To set the server at 27.65.89.42 as collocated (server resides in the same machine as the Network Dispatcher):
`ndcontrol server set 130.40.52.153:80:27.65.89.42 collocated yes`
- To set the weight to 10 for server 27.65.89.42 at port 80 on cluster address 130.40.52.153:
`ndcontrol server set 130.40.52.153:80:27.65.89.42 weight 10`
- To mark the server at 27.65.89.42 as up:
`ndcontrol server up 130.40.52.153:80:27.65.89.42`
- To add a remote server:
`ndcontrol server add 130.40.52.153:80:130.60.70.1 router 130.140.150.0`

ndcontrol set — configure server log



loglevel
The level at which the ndserver logs its activities.

level
The default value of **loglevel** is 1. The range is 0–5.

logsize
The maximum number of bytes to be logged in the log file.

size
The default value of **logsize** is unlimited.

ndcontrol status — display whether the manager and advisors are running

►►ndcontrolstatus◄◄

Examples

- To see what is running:
ndcontrol status

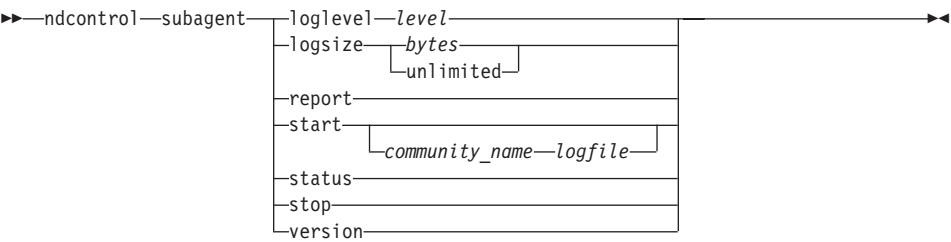
This command produces output similar to:

Executor has been started.
Manager has been started.

ADVISOR	PORT	TIMEOUT
reach	0	unlimited
http	80	unlimited
ftp	21	unlimited

ndcontrol subagent — configure SNMP subagent

Note: Ndcontrol subagent does not apply to CBR.



loglevel

The level at which the subagent logs its activities to a file.

level

The number of the level (0 to 5). The higher the number, the more information that is written to the manager log. The default is 1.

logsize

Set the maximum size of the subagent log. When you set a maximum size for the log file, the file will wrap; when the file reaches the specified size, the subsequent entries will be written from the top of the file, overwriting the previous log entries. Log size cannot be set smaller than the current size of the log. Log entries are time—stamped so you can tell the order in which they were written. The higher you set the log level, the more carefully you should choose the log size, because you can quickly run out of space when logging at the higher levels.

logsize

The maximum number of bytes to be logged in the subagent log file. The default is unlimited.

bytes

The maximum size in bytes for the subagent log file. You can specify either a positive number greater than zero, or the word **unlimited**. The log file may not reach the exact maximum size before overwriting because the log entries themselves vary in size. The default value is unlimited.

report

Display a statistics snapshot report.

start

Start the subagent.

community_name

The name of the SNMP value of community name that you can use as a security password. The default is public.

log file

File name to which the SNMP subagent data is logged. Each record in the log will be time stamped. The default is subagent.log. The default file will be installed in the **logs** directory. See “Appendix E. Sample configuration files” on page 251. To change the directory where the log files will be kept, see “Changing the log file paths” on page 160.

status

Display the current status of all the values in the SNMP subagent that can be set globally and their defaults.

version

Display the current version of the subagent.

Examples

- To start the subagent with a community name of bigguy:
`ndcontrol subagent start bigguy bigguy.log`

Appendix C. Command reference for ISS

This appendix describes how to use the following ISS commands and the configuration keywords for the ISS component of Network Dispatcher:

- “start iss” on page 236
- “isscontrol — control iss” on page 237
- “Configuration file keywords for ISS” on page 238

start iss



- d Debug the daemon startup process. Debug data will be added to the logfile and can help establish why an ISS daemon has failed to start.
- c Use the specified file as the configuration file instead of the default configuration file. The default paths are:
 - **AIX and Solaris:** /etc/iss.cfg
 - **Windows:** \Program Files\IBM\nd\iss\iss.cfg

filename

The name of the configuration file.

- g Enables the administration GUI

portnum

The number that the GUI will bind. This is optional. If no port is specified, it will default to 12099.

- h Display a help message.
- i Run iss interactively.
- l Use the specified file as the log file instead of the default log file. Default paths are:
 - **AIX and Solaris:** /var/log/iss.log
 - **Windows:** \Program Files\IBM\nd\iss\iss.log

filename

The name of the file to use as the log file.

- p Use the specified port for control traffic.

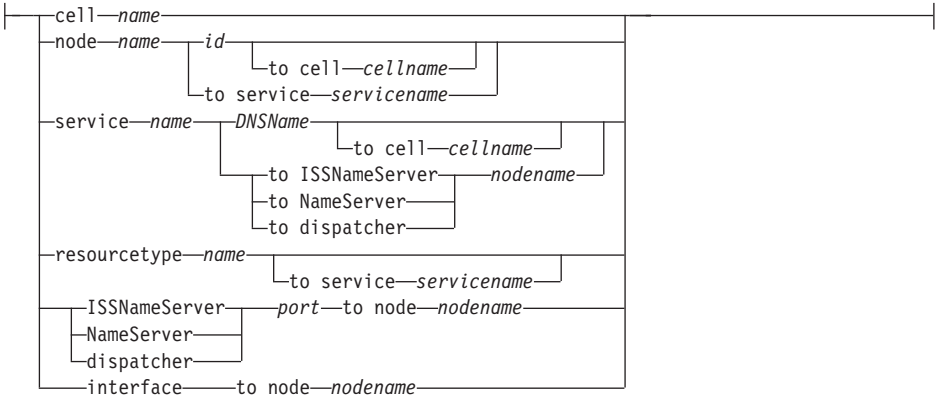
portnum

Defines the port used by RMI to connect to the remote administration GUI.

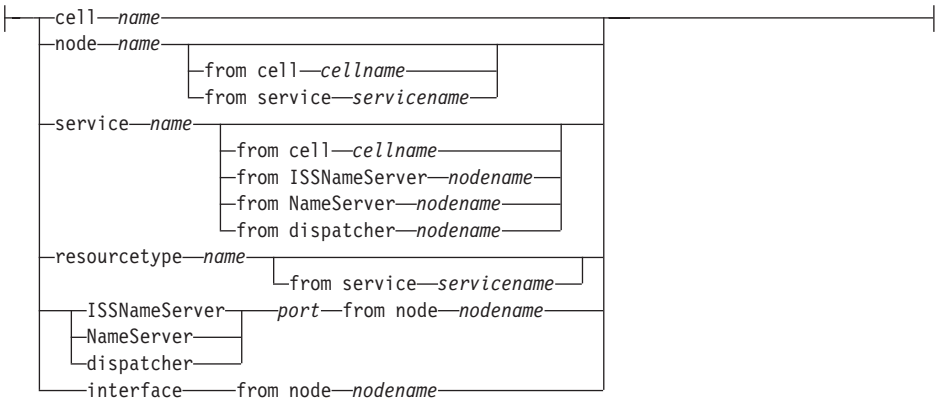
isscontrol — control iss



add operands:



delete operands:



set operands:



Configuration file keywords for ISS

The keywords are placed in alphabetical order.

Alarm

Use this keyword to refer to a **Service**.

Contains a command string to execute if the service runs out of available nodes. The command string is operating-system dependent.

The syntax is:

ALARM *CommandString*

Authkey

Use this keyword to refer to a **Node** or a cell.

This is a default authorization key that will be used by nodes that do not have their own authorization key specified.

The syntax is:

AUTHKEY *HexString*

For convenience, the HexString may be broken up by white space as long as it all fits on one line, and as long as every digit is paired off so the parser may read a byte (two characters) at a time.

For example:

```
AUTHKEY 036454DC 65739CB2 23C68DF1 129C4D61
```

If this keyword is not specified, the authentication will default to a value of all '0's. This value will still be encrypted, but you may find your security is reduced.

If a node does not have a specific **AuthKey** defined, it will inherit the **AuthKey** for the cell.

Cell

This keyword declares another cell to the ISS Monitor.

The syntax is:

```
CELL      Name [Global | Local ]
```

Name must be a Latin-1 string; it will be used as an identifier for the cell when communicating with other cells. The second parameter determines whether the cell described is the local cell that the node will be a member of, or a separate, remote cell that you may wish to communicate with.

A valid config always has 1 local cell. For more information on this keyword, see “Updating the ISS configuration file” on page 146.

Dispatcher

Use this keyword to refer to an **Observer**.

This keyword identifies a Dispatcher as the recipient of ISS load information.

The syntax is:

```
DISPATCHER DNSName [PortNumber]
```

The *DNSName* specifies a node on which to connect a Dispatcher. The *PortNumber* optionally specifies the port on which to start this service. The default PortNumber is 10004.

For example:

```
DISPATCHER      sp1n1
```

HeartbeatInterval

Use this keyword to refer to a **Cell**.

Indicates the time in seconds that ISS allows to elapse before it checks the well-being of the nodes configured in that cell. At every heartbeat, ISS

checks that each server is functional by issuing an ICMP ping command; this is the same as if you enter ping at a UNIX or Windows command line. While this does not test the state of everything on the server, it does at least ensure that the server is reachable at the IP level. If the server fails to respond, ISS will not recommend it.

If you have configured one or more metrics, the ISS agent for each node will send out load information at each heartbeat interval, which allows the ISS monitor to determine the load on each server in the cell, and, if necessary, to remove a node from the recommended list.

The syntax is:

```
HeartbeatInterval heartbeatinterval
```

For example, to initiate a heartbeat every 30 seconds, enter the following:

```
HeartbeatInterval 30
```

If this keyword is not specified, the heartbeat interval will default to 10 seconds.

Note that the product of HeartbeatInterval and HeartbeatsPerUpdate is considered the update interval.

HeartbeatsPerUpdate

Use this keyword to refer to a **Cell**.

Accepts a non-negative integer that indicates the number of heartbeats after which ISS will determine whether a domain name server update or a Dispatcher update is required. This allows you to configure an update interval that is relatively large while using a relatively fast heartbeat, so that the well-being of the top-ranked server is checked frequently, but without putting too much load on the name server by updating it too often. The update interval is determined from the product of HeartbeatInterval and HeartbeatsPerUpdate.

You can use this keyword to optimize the overall responsiveness of the service in specific user workload scenarios. The syntax is:

```
HeartbeatsPerUpdate heartbeat_count
```

For example, to prevent the name server updates from occurring more frequently than once every three heartbeats, enter the following:

```
HeartbeatsPerUpdate 3
```

In this example, if HeartbeatInterval is 10 seconds, you have limited the frequency of updates of the name server or the Dispatcher to once every 30 seconds.

If this keyword is not specified, the HeartbeatsPerUpdate factor will default to 3.

HeartbeatsToNetFail

Use this keyword to refer to a **Cell**.

The number of heartbeats after which a node will be regarded as unreachable and will not be recommended; it will, however, continue to be regarded as running.

The syntax is:

HeartbeatsToNetFail *number*

HeartBeatsToNodeFail

Use this keyword to refer to a **Cell**.

The number of heartbeats after which a node is considered to have failed completely. HeartbeatsToNodeFail must be equal to or greater than HeartbeatstoNetFail.

ISS will still heartbeat to a failed node, the failed status triggers two things: packets and service updates will no longer be routed to a failed node. If the failed node is the monitor, then an election will be held. Since there is still a heartbeat ping being sent to a failed node, it if returns to life it will be brought back into the cell.

The syntax is:

HeartbeatsToNodeFail *number*

Interface

Use this keyword to refer to **Node**.

A DNS name or internet address (in dotted decimal notation) that specifies a particular network adapter card or alias address on a machine with multiple adapters or aliases. Addresses specified via this **Interface** keyword can then be added to the NodeList of a service in order to specify which interface of a node that a service should use.

The syntax is:

Interface IPAddress

where IPAddress is in dotted decimal notation or a DNSName.

InternalNet

Use this keyword to refer to a **Node**.

A DNS name or Internet address (in dotted decimal notation) that specifies the network interface of a machine used for ISS network traffic (communications with the ISS monitor). This keyword allows you to define separate network interfaces for ISS monitor traffic and for normal service traffic.

The syntax is:

InternalNet *DNSName*

ISSNameServer

Use this keyword to refer to an **Observer**.

This keyword identifies an ISS replacement nameserver as an Observer. ISSD must be running on that node. The ISSD daemon will start the nameserver automatically. The default PortNumber is 53.

The syntax is:

ISSNameServer *DNSName* [*PortNumber*]

LogLevel

Use this keyword to refer to a **Cell**.

The syntax is:

LogLevel [0 | 1 | 2 | 3 | 4]

Where:

- 0** Provides no debugging data. Higher numbers provide greater level of detail.

Metric

Use this keyword to refer to a **ResourceType**. Specifies the type of measurement ISS uses.

The syntax is:

METRIC [*Internal* | *External*] *string*

Where:

Internal identifies a given measurement method as being provided by the system. The following metrics are internally specified:

CPUload

This measures the percentage of the CPU that is being utilized. The **Policy** keyword is automatically set to **Min**.

FreeMem

This returns the amount of free physical memory as a percentage. Policy should be set to **Max**.

Note: All internal metrics may not be available on all operating systems.

External is a user-defined metric.

If **Internal**, *String* is the name of one of the system-provided measurement methods. If **External**, *String* is a command to execute to take a load

measurement. That command must return a number as its only output. If multiple operating systems are being used, you must ensure that the command returns appropriate results on each platform.

MetricLimits

Use this keyword to refer to an **ResourceType**.

The first parameter sets the point at which the node should be returned to active participation. The second parameter of this keyword sets the point at which the node resource should be removed from active participation in the Cell, because the load has reached a critical stage. The former is usually significantly lower than the latter, to avoid the phenomenon of resources returned to service too soon, only to quickly rise again to the fail limit.

The syntax is:

```
MetricLimits RecoverLimit FailLimit
```

For example, in defining the limits of our CPU load metric, we might code:

```
MetricLimits 90 95
```

This would cause the resource to be removed at 95 percent utilization, and returned only when it was back down to 90 percent.

MetricNormalization

Use this keyword to refer to a **ResourceType**.

This keyword sets reasonable limits for the metric to which it applies. The first parameter sets the lower limit; the second parameter sets the upper limit. The units indicated are specific to the object of the metric. For example, in measuring CPU load, which is stated in percentage points, reasonable limits on the lower and upper ends would be 0 and 100, respectively.

The syntax is:

```
MetricNormalization LowerLimit UpperLimit
```

For example:

```
MetricNormalization 0 100
```

MetricWeight

A floating-point number, used when one metric is a more important factor than another metric in judging load. Once the normalized metric values have been determined, each value is multiplied by its MetricWeight. The default value for this parameter is 1.0.

The syntax is:

```
MetricWeight <number>
```

NamedData

Use this keyword to refer to a **NameServer**.

Defines the path and filename of the datafile where the DNS NameServer configuration data is stored which ISS needs to know about.

The syntax is:

NamedData *pathname*

NamedRev

Use this keyword to refer to a **NameServer**.

Defines the path and filename of the datafile for ISS to update in order to update reverse name-to-address mappings. This keyword must be provided if reverse datafile updates are required. The omission of this keyword from a cell definition indicates to ISS that reverse updates are not required. The syntax is:

NamedRev *pathname*

NameServer

Use this keyword to refer to an **Observer**.

This keyword identifies a DNS nameserver as an Observer. The default PortNumber is 53.

The syntax is:

NameServer *DNSName* [*PortNumber*]

Note: Adding and deleting namservers can be changed in a global cell using the GUI or the command line.

Node

Declares a node to be a member of the current cell. Every cell declaration must contain an entry for each member node.

The syntax is:

Node *DNSName* *IDNumber*

Where *DNSName* is the hostname that will resolve to the correct service IP address for that node, and *IDNumber* is a unique number that identifies the node and determines its priority level for being selected as monitor.

For example:

Node	dart	3
------	------	---

Note: Adding and deleting nodes can be changed in a global cell using the GUI or the command line.

NodeList

Use this keyword to refer to a **Service**.

Lists the nodes that a Service may use.

The syntax is:

```
NodeList DNSName [DNSName...]
```

The names of the nodes must be separated by white space. At least one node must be listed. This keyword may occupy as many lines as necessary.

For example:

```
Nodelist dart esme delta
```

NodeWeight

Use this keyword to refer to a **Node**.

The syntax is:

```
NodeWeight Number
```

The NodeWeight is a floating-point number used in calculating the node's priority. After an aggregate load score has been calculated for each node, each of these load scores is then multiplied by the appropriate NodeWeight to determine that node's priority. Thus, a higher NodeWeight will result in a higher priority.

```
NodeWeight 3.3
```

NotISSAgent

Use this keyword to refer to a **Node**.

When present, this keyword specifies that the node is not running ISS and so it cannot return load information (or run the monitor function). This should only be used together with the RoundRobin Selection Method, and is not recommended unless it is not possible to run ISS on the node.

If you modify from the GUI or isscontrol and set issagent=true for that node, then all other nodes should be updated to reflect that change.

NotMonitor

Use this keyword to refer to a **Node**.

When present, this keyword specifies that the node is not to be used as the ISS monitor.

The syntax is:

```
NotMonitor
```

Observer

The Observer keyword is no longer used in an ISS config file. An Observer can be one of ISSNameServer, Nameserver, or Dispatcher.

Overflow

Use this keyword to refer to a **Service**.

This keyword identifies a server on which to fall back if all the other nodes in the NodeList are unable to provide the service.

The syntax is:

```
Overflow DNSName
```

For example:

```
Overflow delta
```

PingCache

Use this keyword to refer to a **Cell**.

Specifies that ISS should limit the size of the ping triangulation cache to the number of kilobytes specified. If Ping cache is set to 0, ping triangulation will not operate in that cell. The default is 0.

The syntax is:

```
PingCache number
```

For example:

```
PingCache 10
```

Policy

Use this keyword to refer to a **ResourceType**.

Indicates whether a metric function is to be minimized or maximized. The syntax is:

```
POLICY { MIN | MAX }
```

The default value is MIN.

For example, a metric based on the number of active users would regard the smallest value as the best and would have the following entry:

```
POLICY MIN
```

A metric function based on CPU idle time, on the other hand, would regard the maximum value as best and, therefore, the policy would be specified as the following:

```
POLICY MAX
```

PortNumber

Use this keyword to refer to a **Cell**.

Place this keyword at the start of any line in the configuration file to tell ISS the port number you used when you configured the Cell.

Note: This keyword performs the same function as **ISS_Port** did in previous versions. If fact, ISS_Port may still be used, if desired.

It is recommended you put this entry near the top of the ISS Configuration file, together with the other general definitions.

You can choose any port which is not already in use. Ports which are used are listed in the Services file. The services can be found in the following directories.

- AIX and Solaris — /etc/services
- Windows — c:\winnt\system32\drivers\etc\services

You should add an entry for the port you select for the ISS cell.

If this keyword is not specified, the port number will default to 7139.

The syntax is:

```
PortNumber port_number
```

ResourceList

Use this keyword to refer to a **Service**.

This keyword lists the resources, required by a given service, that reside on a given node. As with the NodeList keyword, these parameters may number from one to many, and may take up as many lines as necessary.

The syntax is:

```
ResourceList ResourceTypeName [ResourceTypeName...]
```

For example:

```
ResourceList CPU    NetworkInterface    VMPages
```

ResourceType

This keyword defines a category of resources. A ResourceType carries with it all the attributes that define a Metric. Some ResourceTypes are defined internally (see the entry for **Metric** above), and do not need to be configured by the user. For each node in the nodelist for a service, ISS creates an instance of that ResourceType.

The syntax is:

```
ResourceType Name
```

Here is an example of a user-defined ResourceType:

```
ResourceType CPUprocess
Metric External ps -ef | wc -l
MetricLimits 800 1000
Policy Min
```

SelectionMethod

Use this keyword to refer to an Service.

This keyword specifies how the ISS Monitor is to select the best node currently able to provide a Service. For more information about this keyword, see “Updating the ISS configuration file” on page 146.

The syntax is:

```
SelectionMethod [RoundRobin | Best] | StatRR]
```

For example:

```
SelectionMethod RoundRobin
```

Service

This keyword identifies a function that relies upon multiple resources, which may include anything you wish to measure load on, such as CPUs, disks, server process, and so on. For more information about this keyword, see “Updating the ISS configuration file” on page 146.

The syntax is:

```
Service Name DNSName [cluster address] [PortNumber]
```

For example:

```
Service      WWWService      www.issdev.hursley.ibm.com.  129.40.161.74      21
```

Note: Adding and deleting services can be changed in a global cell using the GUI or the command line.

Servicelist

Use this keyword to refer to an **Observer**.

A list of services to be performed.

The syntax is:

```
Servicelist ServiceName [ServiceName...]
```

For example:

```
Servicelist WWWService
```

StickyGarbageCollectionInterval

This keyword defines the number of requests that must be made to the ISSNameServer before garbage collection will occur. Garbage collection is run to clean out all timed out sticky table entries. To run garbage collection more often, set this interval to a smaller value. The default is 100. StickyTime value must be nonzero for garbage collection to run.

StickyTime

This keyword allows an ISSNameServer to control the client-server affinity. StickyTime operates much like affinity in Dispatcher. Set StickyTime to the number of seconds the ISSNameServer should return the same address to a particular requester. If StickyTime value is zero (default), then sticky is turned off.

Appendix D. Rule types for CBR

This appendix describes how to use the following CBR rule types.

Rule (pattern) syntax:

In the rule (pattern) syntax, there cannot be any spaces and special characters.

wildcard (*)

(matches 0 to x of any character)

(used for logic grouping

) used for logic grouping

& logical AND

| logical OR

! logical NOT

Reserved keywords

CBR reserved keywords are always followed by an equal sign ('=').

client Client IP address

url URL in request

protocol
 Protocol section of URL

path Path section of URL

refer Referred URL (quality of service)

Here are some of the examples.

`url=http://*/*.gif`

`client=9.32.*`

`!(path=*.jpeg)`

`(path=index/*.gif & protocol=httpd) | (client=9.1.2.3)`

Appendix E. Sample configuration files

This appendix contains sample configuration files for the Dispatcher and the ISS functions of IBM Network Dispatcher.

Sample Network Dispatcher configuration files

Sample files are located in the /dispatcher/samples/ directory.

Dispatcher Configuration file—AIX, Red Hat Linux, and Solaris

```
#!/bin/ksh
#
# configuration.sample - Sample configuration file for the
# Dispatcher component
#
#
# Ensure the root user is the one executing this script.
#
# iam='whoami'

# if [ "$iam" != "root" ]if [ "$iam" != "root" ]
# then
# echo "You must login as root to run this script"
# exit 2
# fi

#
# First start the server
#
# ndserver start
# sleep 5

#
# Then start the executor
#
# ndcontrol executor start

#
# The Dispatcher can be removed at any time using the
# "ndcontrol executor stop" and "ndserver stop" commands to
# stop the executor and server respectively prior to removing
# the Dispatcher software.
#
# The next step in configuring the Dispatcher is to set the
# NFA (non-forwarding address) and the cluster address(es).
#
# The NFA is used to remotely access the Dispatcher machine
# for administration or configuration purposes. This
# address is required since the Dispatcher will forward packets
```

```

# to the cluster address(es).
#
# The CLUSTER address is the hostname (or IP address) to
# which remote clients will connect.
#
# Anywhere in this file, you may use hostnames and IP
# addresses interchangeably.
#

# NFA=hostname.domain.name
# CLUSTER=www.yourcompany.com

# echo "Loading the non-forwarding address"
# ndcontrol executor set nfa $NFA

#
# The next step in configuring the Dispatcher is to create
# a cluster. The Dispatcher will route requests sent to
# the cluster address to the corresponding server machines
# defined to that cluster. You may configure and server
# multiple cluster address using Dispatcher.

# Use a similar configuration for CLUSTER2, CLUSTER3, etc.
#

# echo "Loading first CLUSTER address "
# ndcontrol cluster add $CLUSTER

#
# Now we must define the ports this cluster will use. Any
# requests received by the Dispatcher on a defined port will
# be forwarded to the corresponding port of one of the server
# machines.
#

# echo "Creating ports for CLUSTER: $CLUSTER"

# ndcontrol port add $CLUSTER:20+21+80

#
# The last step is to add each of the server machines to the
# ports in this cluster.
# Again, you can use either the hostname or the IP address
# of the server machines.
#

# SERVER1=server1name.domain.name
# SERVER2=server2name.domain.name
# SERVER3=server3name.domain.name

# echo "Adding server machines"
# ndcontrol server add $CLUSTER:20+21+80:
# $SERVER1+$SERVER2+$SERVER3

#

```

```

# We will now start the load balancing components of the
# Dispatcher. The main load balancing component is called
# the manager and the second load balancing components are the
# advisors. If the manager and advisors are not running the
# Dispatcher sends requests in a round-robin format. Once the
# manager is started, weighting decisions based on the number
# of new and active connections is employed and incoming
# requests are sent to the best server. The advisors give the
# manager further insight into a servers ability to service
# requests as well as detecting whether a server is up. If
# an advisor detects that a server is down it will be
# marked down (providing the manager proportions have been
# set to include advisor input) and no further requests will be
# routed to the server.

# The last step in setting up the load balancing components
# is to set the manager proportions. The manager updates the
# weight of each of the servers based on four policies:
# 1. The number of active connections on each server.
# 2. The number of new connections to each server.
# 3. Input from the advisors.
# 4. Input from the system level advisor (ISS).
# These proportions must add up to 100. As an example, setting
# the manager proportions to
# ndcontrol manager proportions 48 48 0 0
# will give active and new connections 48% input into the
# weighting decision, the advisors will contribute 4% and
# the system input will not be considered.
#
# NOTE: By default the manager proportions are set to 50 50 0 0
#

# echo "Starting the manager..."
# ndcontrol manager start

# echo "Starting the FTP advisor on port 21 ..."
# ndcontrol advisor start ftp 21
# echo "Starting the HTTP advisor on port 80 ..."
# ndcontrol advisor start http 80
# echo "Starting the Telnet advisor on port 23 ..."
# ndcontrol advisor start telnet 23
# echo "Starting the SMTP advisor on port 25 ..."
# ndcontrol advisor start smtp 25
# echo "Starting the POP3 advisor on port 110 ..."
# ndcontrol advisor start pop3 110
# echo "Starting the NNTP advisor on port 119 ..."
# ndcontrol advisor start nntp 119
# echo "Starting the SSL advisor on port 443 ..."
# ndcontrol advisor start ssl 443
#

# echo "Setting the manager proportions..."
# ndcontrol manager proportions 58 40 2 0

#

```

```

# The final step in setting up the Dispatcher machine is to
# alias the Network Interface Card (NIC).
#
# NOTE: Do NOT use this command in a high availability
# environment. The go* scripts will configure the NIC and
# loopback as necessary.
# ndcontrol cluster configure $CLUSTER

# If your cluster address is on a different NIC or subnet
# from the NFA use the following format for the cluster configure
# command.
# ndcontrol cluster configure $CLUSTER tr0 0xfffff800
# where tr0 is your NIC (tr1 for the second token ring card, en0
# for the first ethernet card) and 0xfffff800 is a valid
# subnet mask for your site.
#

#
# The following commands are set to the default values.
# Use these commands as a guide to change from the defaults.
# ndcontrol manager loglevel 1
# ndcontrol manager logsize unlimited
# ndcontrol manager sensitivity 5.000000
# ndcontrol manager interval 2
# ndcontrol manager refresh 2
#
# ndcontrol advisor interval ftp 21 5
# ndcontrol advisor loglevel ftp 21 1
# ndcontrol advisor logsize ftp 21 unlimited
# ndcontrol advisor timeout ftp 21 unlimited
# ndcontrol advisor interval telnet 23 5
# ndcontrol advisor loglevel telnet 23 1
# ndcontrol advisor logsize telnet 23 unlimited
# ndcontrol advisor timeout telnet 23 unlimited
# ndcontrol advisor interval smtp 25 5
# ndcontrol advisor loglevel smtp 25 1
# ndcontrol advisor logsize smtp 25 unlimited
# ndcontrol advisor timeout smtp 25 unlimited
# ndcontrol advisor interval http 80 5
# ndcontrol advisor loglevel http 80 1
# ndcontrol advisor logsize http 80 unlimited
# ndcontrol advisor timeout http 80 unlimited
# ndcontrol advisor interval pop3 110 5
# ndcontrol advisor loglevel pop3 110 1
# ndcontrol advisor logsize pop3 110 unlimited
# ndcontrol advisor timeout pop3 110 unlimited
# ndcontrol advisor interval nntp 119 5
# ndcontrol advisor loglevel nntp 119 1
# ndcontrol advisor logsize nntp 119 unlimited
# ndcontrol advisor timeout nntp 119 unlimited
# ndcontrol advisor interval ssl 443 5
# ndcontrol advisor loglevel ssl 443 1
# ndcontrol advisor logsize ssl 443 unlimited
# ndcontrol advisor timeout ssl 443 unlimited
#

```

Dispatcher Configuration file—Windows

The following is a sample Network Dispatcher configuration file called **configuration.cmd.sample** for use with Window.

```
@echo off
rem configuration.cmd.sample - Sample configuration file for the
rem Dispatcher component.
rem

rem ndserver must be started via Services

rem

rem
rem Then start the executor
rem
rem call ndcontrol executor start

rem

rem The next step in configuring the Dispatcher is to set the
rem NFA (non-forwarding address) and to set the cluster
rem address(es).
rem

rem The NFA is used to remotely access the Dispatcher
rem machine for administration configuration purposes. This
rem address is required since the Dispatcher will forward
rem packets to the cluster address(es).

rem
rem The CLUSTER address is the hostname (or IP address) to which
rem remote clients will connect.
rem

rem Anywhere in this file, you may use hostnames and IP
rem addresses interchangeably.
rem NFA=[non-forwarding address]
rem CLUSTER=[your clustername]
rem

rem set NFA=hostname.domain.name
rem set CLUSTER=www.yourcompany.com

rem echo "Loading the non-forwarding address"
rem call ndcontrol executor set nfa %NFA%

rem
rem The following commands are set to the default values.
rem Use these commands to change the defaults

rem call ndcontrol executor set fintimeout 30
rem call ndcontrol executor set fincount 4000
rem
rem The next step in configuring the Dispatcher is to create
```

```

rem a cluster. The Dispatcher will route requests sent to
rem the cluster address to the corresponding server machines
rem defined to that cluster. You may configure and server
rem multiple cluster addresses using Dispatcher.
rem Use a similar configuration for CLUSTER2, CLUSTER3, etc.
rem

rem echo "Loading first CLUSTER address "
rem call ndcontrol cluster add %CLUSTER%

rem
rem Now we must define the ports this cluster will use. Any
rem requests received by the Dispatcher on a defined port
rem will be forwarded to the corresponding
rem port of one of the server machines.
rem

rem echo "Creating ports for CLUSTER: %CLUSTER%"
rem call ndcontrol port add %CLUSTER%:20+21+80

rem
rem The last step is to add each of the server machines to
rem the ports in this cluster. Again, you can use either the
rem hostname or the IP address of the server machines.
rem

rem set SERVER1=server1name.domain.name
rem set SERVER2=server2name.domain.name
rem set SERVER3=server3name.domain.name

rem echo "Adding server machines"
rem call ndcontrol server add %CLUSTER%:20+21+80:
rem %SERVER1%+%SERVER2%+%SERVER3%

rem
rem We will now start the load balancing components of the
rem Dispatcher. The main load balancing component is called
rem the manager and the second load balancing components are the
rem advisors. If the manager and advisors are not
rem running the Dispatcher sends requests in a round-robin
rem format. Once the manager is started, weighting decisions
rem based on the number of new and active connections is
rem employed and incoming requests are sent to the best
rem server. The advisors give the manager further insight
rem into a servers ability to service requests as well as
rem detecting whether a server is up. If an advisor detects
rem that a server is down it will be marked down (providing the
rem manager proportions have been set to include advisor
rem input) and no further requests will be routed to the server.
rem The last step in setting up the load balancing
rem components is to set the manager proportions. The
rem manager updates the weight of each of the servers based
rem on four policies:

rem 1. The number of active connections on each server

```



```

rem 2. The number of new connections for each server
rem 3. Input from the advisors.
rem 4. Input from the system level advisor (ISS).
rem
rem These proportions must add up to 100. As an example,
rem setting the manager proportions to ndcontrol manager
rem proportions 48 48 0 0 will give active and new
rem connections 48% input into the weighting decision,
rem the advisor will contribute 4% and the system input will
rem not be considered.
rem
rem NOTE: By default the manager proportions are set to 50 50
rem 0 0

rem echo "Starting the manager..."
rem call ndcontrol manager start

rem echo "Starting the FTP advisor on port 21 ..."
rem call ndcontrol advisor start ftp 21
rem echo "Starting the HTTP advisor on port 80 ..."
rem call ndcontrol advisor start http 80
rem echo "Starting the Telnet advisor on port 23 ..."
rem call ndcontrol advisor start telnet 23
rem echo "Starting the SMTP advisor on port 25 ..."
rem call ndcontrol advisor start smtp 25
rem echo "Starting the POP3 advisor on port 110 ..."
rem call ndcontrol advisor start pop3 110
rem echo "Starting the NNTP advisor on port 119 ..."
rem call ndcontrol advisor start nntp 119
rem echo "Starting the SSL advisor on port 443 ..."
rem call ndcontrol advisor start ssl 443
rem

rem echo "Setting the manager proportions..."
rem call ndcontrol manager proportions 58 40 2 0

rem
rem The final step in setting up the Dispatcher machine is
rem to alias the Network Interface Card (NIC).
rem
rem NOTE: Do NOT use this command in a high availability
rem environment. The go* scripts will configure the NIC and
rem loopback as necessary.
rem
rem ndcontrol cluster configure %CLUSTER%

rem If you cluster address is on a different NIC or subnet
rem from the NFA use the following format for the cluster
rem configure command.
rem ndcontrol cluster configure %CLUSTER% tr0 0xfffff800
rem where tr0 is your NIC (tr1 for the second token ring card,
rem en0 for the first ethernet card) and 0xfffff800 is
rem a valid subnet mask for your site.
rem

```

```

rem
rem The following commands are set to the default values.
rem Use these commands to guide to change from the defaults.
rem call ndcontrol manager loglevel 1
rem call ndcontrol manager logsize unlimited
rem call ndcontrol manager sensitivity 5.000000
rem call ndcontrol manager interval 2
rem call ndcontrol manager refresh 2
rem
rem call ndcontrol advisor interval ftp 21 5
rem call ndcontrol advisor loglevel ftp 21 1
rem call ndcontrol advisor logsize ftp 21 unlimited
rem call ndcontrol advisor timeout ftp 21 unlimited
rem call ndcontrol advisor interval telnet 23 5
rem call ndcontrol advisor loglevel telnet 23 1
rem call ndcontrol advisor logsize telnet 23 unlimited
rem call ndcontrol advisor timeout telnet 23 unlimited
rem call ndcontrol advisor interval smtp 25 5
rem call ndcontrol advisor loglevel smtp 25 1
rem call ndcontrol advisor logsize smtp 25 unlimited
rem call ndcontrol advisor timeout smtp 25 unlimited
rem call ndcontrol advisor interval http 80 5
rem call ndcontrol advisor loglevel http 80 1
rem call ndcontrol advisor logsize http 80 unlimited
rem call ndcontrol advisor timeout http 80 unlimited
rem call ndcontrol advisor interval pop3 110 5
rem call ndcontrol advisor loglevel pop3 110 1
rem call ndcontrol advisor logsize pop3 110 unlimited
rem call ndcontrol advisor timeout pop3 110 unlimited
rem call ndcontrol advisor interval nntp 119 5
rem call ndcontrol advisor loglevel nntp 119 1
rem call ndcontrol advisor logsize nntp 119 unlimited
rem call ndcontrol advisor timeout nntp 119 unlimited
rem call ndcontrol advisor interval ssl 443 5
rem call ndcontrol advisor loglevel ssl 443 1
rem call ndcontrol advisor logsize ssl 443 unlimited
rem call ndcontrol advisor timeout ssl 443 unlimited
rem

```

Sample advisor

The following is a sample advisor file called **ADV_sample**.

```

/**
 * ADV_sample: The Network Dispatcher HTTP advisor
 *
 *
 * This class defines a sample custom advisor for Network Dispatcher.
 * Like all advisors, this custom advisor extends the function of the
 * advisor base, called ADV_Base. It is the advisor base that actually
 * performs most of the advisor's functions, such as reporting loads back
 * to the Network Dispatcher for use in the Network Dispatcher's weight
 * algorithm. The advisor base also performs socket connect and close
 * operations and provides send and receive methods for use by the advisor.
 * The advisor itself is used only for sending and receiving data to and
 * from the port on the server being advised.
 * The TCP methods within the advisor base are timed to calculate the load.

```

```

* A flag within the constructor in the ADV_base
* overwrites the existing load with the new load returned from the advisor
* if desired.
*
* Note: Based on a value set in the constructor, the advisor base supplies
* the load to the weight algorithm at specified intervals. If the actual
* advisor has not completed so that it can return a valid load, the advisor
* base uses the previous load.
*
* NAMING
*
* The naming convention is as follows:
*
* - The file must be located in the following Network Dispatcher
*   Directories:
*
*   - For dispatcher - nd/dispatcher/lib/CustomAdvisors/
*                     (nd\dispatcher\lib\CustomAdvisors on NT)
*   - For CBR        - nd/cbr/lib/CustomAdvisors/
*                     (nd\cbr\lib\CustomAdvisors on NT)
*
* - The Advisor name must be preceded with "ADV_". The advisor can
*   be started with only the name, however; for instance, the "ADV_sample"
*   advisor can be started with "sample".
*
* - The advisor name must be in lowercase.
*
* With these rules in mind, therefore, this sample is referred to as:
*
*   <base directory>/samples/CustomAdvisors/
*
* Advisors, as with the rest of Network Dispatcher, must be compiled with
* the prereq version of Java.
* To ensure access to Network Dispatcher classes, make sure that the
* ibmnd.jar file (located in the lib subdirectory of the base directory)
* is included in the system's CLASSPATH.
*
* Methods provided by ADV_Base:
*
* - ADV_Base (Constructor):
*
*   - Params
*     - String sName = Name of the advisor
*     - String sVersion = Version of the advisor
*     - int iDefaultPort = Default port number to advise on
*     - int iInterval = Interval on which to advise on the servers
*     - String sDefaultLogFileName = Unused. Must be passed in as "".
*     - boolean replace = True - replace the load value being calculated
*                           by the advisor base
*                           False - add to the load value being calculated
*                                 by the advisor base
*   - Return
*     - Constructors do not have return values.

```

```

*
* Because the advisor base is thread based, it has several other methods
* available for use by an advisor. These methods can be referenced using
* the CALLER parameter passed in getLoad().
*
* These methods are as follows:
*
* - send - Send a packet of information on the established socket
*          connection to the server on the specified port.
*   - Params
*     - String sDataString - The data to be sent is sent in the form of a
*       string
*   - Return
*     - int RC - Whether the data was successfully sent or not: zero
*               indicates data was sent; a negative integer indicates an
*               error.
*
* - receive - Receive information from the socket connection.
*   - Params
*     - StringBuffer sbDataBuffer - The data received during the receive
*       call
*   - Return
*     - int RC - Whether the data was successfully received or not; zero
*               indicates data was sent; a negative integer indicates an error.
*
* If the function provided by the advisor base is
* not sufficient, you can create the appropriate function within the
* advisor and the methods provided by the advisor base will then be
* ignored.
*
* An important question regarding
* the load returned is whether to apply it to the load being generated
* within the advisor base, or to replace it; there are valid instances of
* both situations.
*
* This sample is essentially the Network Dispatcher HTTP advisor. It
* functions very simply:
* a send request--an http head request--is issued. Once a response is
* received, the getLoad method terminates, flagging the advisor base to
* stop timing the request. The method is then complete. The information
* returned is not parsed; the load is based on the time required
* to perform the send and receive operations.
*/

```

```

package CustomAdvisors;
import com.ibm.internet.nd.advisors.*;

public class ADV_sample extends ADV_Base implements ADV_MethodInterface
{
    String COPYRIGHT = "(C) Copyright IBM Corporation 1997,
                       All Rights Reserved.\n";

    static final String  ADV_NAME          = "Sample";
    static final int     ADV_DEF_ADV_ON_PORT = 80;
    static final int     ADV_DEF_INTERVAL  = 7;
}

```

```

// Note: Most server protocols require a carriage return ("\r") and line
// feed ("\n") at the end of messages. If so, include them in your
// string here.
static final String ADV_SEND_REQUEST =
    "HEAD / HTTP/1.0\r\nAccept: */*\r\nUser-Agent: " +
    "IBM_Network_Dispatcher_HTTP_Advisor\r\n\r\n";

/**
 * Constructor.
 *
 * Parms: None; but the constructor for ADV_Base has several parameters
 * that must be passed to it.
 */
public ADV_sample()
{
    super( ADV_NAME,
        "2.0.0.0-03.27.98",
        ADV_DEF_ADV_ON_PORT,
        ADV_DEF_INTERVAL,
        "", // not used
        false);
    super.setAdvisor( this );
}

/**
 * ADV_AdvisorInitialize
 *
 * Any Advisor-specific initialization that must take place after the
 * advisor base is started.
 * This method is called only once and is typically not used.
 */
public void ADV_AdvisorInitialize()
{
    return;
}

/**
 * getLoad()
 *
 * This method is called by the advisor base to complete the advisor's
 * operation, based on details specific to the protocol. In this sample
 * advisor, only a single send and receive are necessary; if more complex
 * logic is necessary, multiple sends and receives can be issued.
 * For example, a response might be received and parsed. Based on the
 * information learned thereby, another send and receive could be issued.
 *
 * Parameters:
 *
 * - iConnectTime - The current load as it refers to the length of time it
 *                  took to complete the connection to the server through
 *                  the specified port.

```

```

*
* - caller - A reference to the advisor base class where the Network
*             Dispatcher-supplied methods are to perform simple TCP
*             requests, mainly send and receive.
*
* Results:
*
* - The load - A value, expressed in milliseconds, that can either be
*             added to the existing load, or that can replace the existing load,
*             as determined by the constructor's "replace" flag.
*
* The larger the load, the longer it took the server to respond;
* therefore, the higher the weight will be within Network Dispatcher
* regarding load balancing.
*
* If the value is negative, an error is assumed. An error from an
* advisor indicates that the server the advisor is trying to reach is
* not accessible and has been identified as being down.
* Network Dispatcher will not attempt to load balance to a server that
* is down. Network Dispatcher will resume load balancing to the server
* when a positive value is received.
*
* A value of zero is typically not returned; Network Dispatcher handles
* a load of zero in a special way. Zero is assumed to indicate an
* unknown status, and Network Dispatcher gives the server a high
* weight in response.
*/
public int getLoad(int iConnectTime, ADV_Thread caller)
{
    int iRc;
    int iLoad = ADV_HOST_INACCESSIBLE; // -1

    // Send tcp request
    iRc = caller.send(ADV_SEND_REQUEST);
    if (iRc >= 0)
    {
        // Perform a receive
        StringBuffer sbReceiveData = new StringBuffer("");
        iRc = caller.receive(sbReceiveData);

        // If the receive is successful, a load of zero is returned.
        // This is because the "replace" flag is set to false,
        // indicating that the load built within the base advisor is
        // to be used.
        // Since nothing was done with the returned data, additional
        // load is not necessary.

        // Note: it is known that the advisor base load will not be
        // zero, therefore a zero load will
        // not be returned for use in calculating the weight.
        if (iRc >= 0)
        {
            iLoad = 0;
        }
    }
}

```

```

    return iLoad;
}

} // End - ADV_sample

```

Sample ISS configuration files

Dispatcher Sample configuration file for ISS

The sample files can be found in the /iss/samples/ directory.

```

# This is a sample configuration file used to supply an
# additional load value to IBM Network Dispatcher.
# There is only one (local) cell, and one service running.
# Two Metrics are combined to create the load value for
# the service. Since there are two Dispatcher machines
# running in a Highly Available configuration,
# both dispatchers are listed as observers.

# Parameters for the whole cell
# In this cell, ITL, the Monitor will attempt to reach
# each node every 5 seconds (HeartbeatInterval). After 3
# Intervals, or 15 seconds, each service will be executed.
Cell      ITL      local
AuthKey    10043572 ADE4F354 7298FAE3 1928DF54 12345678
LogLevel   INFO
HeartbeatInterval      5
HeartbeatsPerUpdate    3
PortNumber              7139

# Individual node data

# The node dispatch1 has ISS monitor priority 1 and will
# run the iss process in ISS monitor mode. The node dispatch2
# has monitor priority 2 and can run as a backup ISS monitor
# when dispatch1 is down. The last node, server2, is
# declared NotMonitor. The iss processes on this node will run
# as ISS agents, collecting load statistics, but cannot
# assume the role of an ISS monitor.
# The nodes dispatch2 and server1 are equipped with
# additional resources and are considered to have an
# advantage over the other servers and should be
# recommended more often than the others.
# Therefore they are given a higher weight.

Node    dispatch1.raleigh.ibm.com    001
Node    dispatch2.raleigh.ibm.com    002
NodeWeight 1.5
Node    server1.raleigh.ibm.com      003
NodeWeight 1.5
Node    server2.raleigh.ibm.com      098
NotMonitor
NodeWeight 1.0

```

```
# The service is configured to depend on two resources -- CPU
# availability and the process count. Load balancing is
# therefore performed based on CPU utilization and the number
# of processes currently running. However, ISS will not
# schedule work for nodes that are unreachable on the network.
```

```
# In this configuration, CPU utilization has more value than
# the process count for determining a better performance
# from a server. Therefore, the CPU value will be considered
# 2.0 times more valuable than the ProcessCount.
```

```
# Definition of "CPU" ResourceType
```

```
# The specified MetricLimits indicate that a node will
# not be used if its CPU usage goes over 95% and will not
# be put back in the list until CPU usage goes back down
# to 80%.
```

```
ResourceType      CPU
Metric Internal    CPULoad
MetricNormalization 0      100
MetricLimits      80      95
Policy Min
MetricWeight 2.0
```

```
# Definition of "ProcessCount" ResourceType
```

```
# The metric associated with ProcessCount is assigned a range
# of (0-250). If "ps -fe | wc -l" returns a value greater
# than 250, the metric is assigned the upper limit of 250.
# When the value of this metric on a node from active
# participation in the Service that it is associated with.
# The node is returned to active participation when the
# metric has a value less than or equal to 200.
```

```
ResourceType      ProcessCount
Metric External    ps -ef | wc -l
MetricNormalization 0      250
MetricLimits      200      225
Policy Min
MetricWeight 1.0
```

```
# The one configured service, eNDSservice, is an identifying
# string for the service, and is used when balancing between
# cells. In this configuration, it is used simply to identify
# what the service is for and define which servers to collect
# the selected metric values from.
```

```
# The name Placeholder is not needed when you are using ISS
# to update the Dispatcher. The "cluster address",
# Dispatch_cluster, and port 80, are used to signify which
# set of Dispatcher servers are relevant to this service, and
# where ISS should send the load information to. ISS itself
# does no load balancing in this scenario.
```



```

Service      eNDSERVICE      Placeholder Dispatch_cluster 80
NodeList     dispatch2.raleigh.ibm.com
server1.raleigh.ibm.com server2.raleigh.ibm.com
ResourceList CPU      ProcessCount
SelectionMethod Best

```

```

# The machine dispatch1.raleigh.ibm.com and
# dispatch2.raleigh.ibm.com have an eNetwork Dispatcher
# configured to listen for load values on port 10004. Since
# Dispatcher is running in a High Availability configuration,
# both Dispatchers need to be listed.

```

```

Dispatcher      dispatch1.raleigh.ibm.com 10004
ServiceList eNDSERVICE

```

```

Dispatcher      dispatch2.raleigh.ibm.com 10004
ServiceList eNDSERVICE

```

Replace DNS Sample configuration file for ISS

```

# This ISS configuration file illustrates how ISS can be
# used with as a limited DNS name server to perform load
# balancing for two clusters of servers. A different load
# balancing method (RoundRobin and Best) is used for each
# cluster. StatRR is another possible Selection Method to
# consider using.

```

```

# Basic parameters of ITL cell.

```

```

Cell      ITL local
AuthKey   036454DC 65739CB2 23C68DF1 129C4D61
LogLevel  Trace
HeartbeatInterval 15
HeartbeatsPerUpdate 2
PortNumber 11364

```

```

# Nodes in ITL cell

```

```

# The node monitor1 has ISS monitor priority 1 and will
# run the iss process in ISS monitor mode. The node monitor2 has
# monitor priority 2 and can run as a backup ISS monitor when
# monitor1 is down. The other nodes are declared NotMonitor.
# The iss processes on these nodes will run as ISS agents or
# perform other special ISS functions. They can not assume
# the role of an ISS monitor.

```

```

Node      monitor1.raleigh.ibm.com 01
Node      monitor2.raleigh.ibm.com 02
Node      nameserver.raleigh.ibm.com 31
NotMonitor

```

```

# Definition of "Memory" ResourceType

```

```

# The metric associated with Memory is an Internal metric of

```

```
# ISS, FreeMem, that is available on AIX and NT. It is
# assigned a range of (0-100). It returns the percentage of
# free memory. Since a higher percentage would be desirable,
# the policy should be Max. When the value of this metric
# on a node is less than 10, the node is removed from active
# participation in the Service that it is associated with.
# The node is returned to active participation when the
# metric has a value greater than or equal to 20.
```

```
ResourceType      Memory
Metric            Internal  FreeMem
MetricNormalization 0    100
MetricLimits      20    10
Policy            Max
```

```
# Definition of Services available in ITL cell.
```

```
# The first service, service1, has the DNS name
# cluster1.raleigh.ibm.com associated with it. ISS will serve
# requests which are addressed to cluster1.raleigh.ibm.com.
# Also associated with service1 are the nodes monitor1 and
# monitor2.
# Since the observer is an ISSNameServer, ISS will allow
# the ISS name server running on the nameserver node to
# resolve the DNS name cluster1.raleigh.ibm.com to the IP
# address of either monitor1 or monitor2 and then send the
# resolved address back to the DNS name server.
```

```
# Since the SelectionMethod is RoundRobin, this mapping of the
# name cluster1 to one of the two IP addresses is done using a
# simple round-robin algorithm. A new address is selected once
# every 60 seconds (HeartbeatInterval * HeartbeatsPerUpdate)
# and once every 20 seconds (HeartbeatInterval) ISS checks
# that the server whose address is currently recommended
# is functional at its ICMP layer, and not too heavily
# loaded.
# If at any time there are no servers available, the name
# cluster1 will be removed from the service and the alarm
# triggered, which in this case makes an entry into a file
# called /tmp/issalarm.log.
```

```
Service          service1 cluster1.raleigh.ibm.com
NodeList          monitor1.raleigh.ibm.com
                  monitor2.raleigh.ibm.com
SelectionMethod   RoundRobin
Alarm echo "cluster1 ran out of servers at 'date'" >>
/tmp/issalarm.log
```

```
# The second service, service2, has the DNS name
# cluster2.raleigh.ibm.com associated with it. Also associated
# with service2 are the nodes monitor1 and monitor2. Since
# the observer is an ISSNameServer, ISS will allow the
```

```

# ISS nameserver running on the nameserver node to resolve
# the DNS name cluster2.raleigh.ibm.com to the IP address of
# either monitor1 or monitor2 and then send the resolved
# address back to the client.

# Since the SelectionMethod is Best and the ResourceList
# keyword specifies Memory, the name cluster2.raleigh.ibm.com
# will be mapped to the IP address of the best of the 2
# servers. The best server in this case is the one with
# the highest percentage of free memory as reported by the
# internal metric FreeMem.
# If all the servers in cluster2 have exceeded the free memory
# limit then the IP address of the overflow server
# (nameserver) will be used.

Service                service2 cluster2.raleigh.ibm.com
NodeList               monitor1.raleigh.ibm.com
                      monitor2.raleigh.ibm.com
ResourceList           Memory
SelectionMethod         Best
Overflow               nameserver.raleigh.ibm.com

# Defining a NameServer Observer

# The ISSNameServer observer running on the node nameserver
# subscribes to both services defined in this file. ISS will
# use the ISS name server to load balance the two clusters of
# servers.

ISSNameServer          nameserver.raleigh.ibm.com  53
ServiceList            service1 service2

# End of ISS configuration file -----

```

A Two Tier Sample configuration file for ISS

```

# This ISS configuration file illustrates how ISS can be
# used with the Dispatcher in a 2-tiered architecture. The
# bottom tier consists of two clusters of servers. Each
# cluster itself has two servers. Two Dispatchers are used
# to load balance these clusters. The top tier consists of
# the two Dispatcher nodes. In this example, ISS performs two
# functions:

1. Performs DNS load balancing for the top tier.
2. Provides the Dispatchers with additional server load
information so that they can load balance the bottom tier
more accurately.
# -----
# Basic parameters of ITL cell.

Cell                   ITL local
AuthKey                036454DC 65739CB2 23C68DF1 129C4D61
LogLevel               Info
HeartbeatInterval      15

```

```

HeartbeatsPerUpdate  2
PortNumber           11364

# Nodes in ITL cell

# The node monitor1 has ISS monitor priority 1 and will
# run the iss process in ISS monitor mode. The node monitor2
# has monitor priority 2 and can run as a backup ISS monitor
# when monitor1 is down. The other nodes are declared
# NotMonitor.
# The iss processes on these nodes will run as ISS agents or
# perform other special ISS functions. They can not assume
# the role of an ISS monitor. The IP addresses 9.67.128.101
# and 9.67.128.102 are the "cluster addresses" of the two
# Dispatcher nodes.

Node      monitor1.raleigh.ibm.com 01
Node      monitor2.raleigh.ibm.com 02
Node      server1.raleigh.ibm.com  51
NotMonitor
Node      server2.raleigh.ibm.com  52
NotMonitor
Node      9.67.128.101              81
NotMonitor
InternalNet dispatch1.raleigh.ibm.com
Node      9.67.128.102              82
NotMonitor
InternalNet dispatch2.raleigh.ibm.com

# Definition of "CPU" ResourceType

# On AIX, "vmstat 3 2 | tail -1 | awk '{print $16}'" returns
# the percentage of CPU Idle time during a 3 seconds
# sampling period. This value should be maximized when you
# want to identify the least loaded server.
# (On Solaris, use: "vmstat 3 2 | tail -1 | awk
# '{print $22}'").
# When the value of this metric on a node is less than 20,
# the node is removed from active participation in the
# Service that it is associated with. The node is returned
# to active participation when the metric has a value
# greater than or equal to 60.

ResourceType      CPU
Metric            External  vmstat 3 2 | tail -1 | awk
                  '{print $16}'
MetricNormalization 0    100
MetricLimits       60  20
Policy             Max

# Definition of Services available in ITL cell.

# The service dispatch1_http is intended to be used by
# a Dispatcher observer.
# The DNS name pool1 of this service is needed as a place

```

```

# holder but is not significant and will be ignored by ISS
# and the Dispatcher. However, "cluster address" and the port
# number are significant and must be specified.
# The service top_tier will be used only by the ISSNameServer
# or NameServer types of observer. In this case, the DNS name
# pool1.raleigh.ibm.com is significant but "cluster address"
# and port number specifications are not required.

Service                dispatch1_http pool1 9.67.128.101 80
NodeList               server1.raleigh.ibm.com
server2.raleigh.ibm.com
ResourceList           CPU

Service                top_tier pool1.raleigh.ibm.com
NodeList               9.67.128.101 9.67.128.102
ResourceList           CPU
SelectionMethod         Best

# Definition of Observers in ITL cell

# The type of the first observer is ISSNameServer. This
# observer subscribes to the service top_tier. The ISS
# replacement name server will be started on the node
# monitor1 and will attempt to resolve the DNS name
# pool1.raleigh.ibm.com to the IP address of the best of the
# two servers of the top_tier service (9.67.128.101 and
# 9.67.128.102). The "best" server in this case is the one
# with the highest amount of idle cpu as reported by the
# "vmstat 3 2 | tail -1 | awk '{print $16}'" command.

# It should be noted that 9.67.128.101 and 9.67.128.102 are
# the "cluster addresses" of the Dispatcher nodes
# dispatch1.raleigh.ibm.com and dispatch2.raleigh.ibm.com
# respectively.

# The type of the other observers in the ITL cell is
# Dispatcher.
# For these observers, the role of ISS is to gather
# information.
# Since the Dispatcher running on the node
# dispatch1.raleigh.ibm.com subscribes to the service
# dispatch1_http, ISS will evaluate the metrics associated
# with service dispatch1_http (CPU) once every 30 seconds
# (HeartBeatInterval HeartbeatsPerUpdate) and send the
# results to this Dispatcher. The Dispatcher running on the
# node dispatch1 will use this load information, along with
# other sources of information, to perform load balancing.
# ISS performs similar information gathering functions
# for the Dispatcher running on node dispatch2.

# Assuming that all the appropriate DNS data files have been
# set up correctly, when a client attempts to connect to port

```

```

# 80 of pool1.raleigh.ibm.com, the ISS replacement name server
# running on the node monitor1.raleigh.ibm.com will be asked
# to resolve the name pool1.raleigh.ibm.com. This name
# server will return the "cluster address" of the "best"
# Dispatcher in the top tier. We will assume for the present
# that this address is 9.67.128.101.

# When the client attempts to connect to port 80 of
# 9.67.128.101, the Dispatcher running on the node
# dispatch1.raleigh.ibm.com (whose cluster address is
# 9.67.128.101) will perform the appropriate actions so that
# the client is connected to port 80 of its "best" server.
# For this setup, ISS does the load balancing at the top tier
# and the Dispatcher does the load balancing at the bottom tier.

ISSNameServer      monitor1.raleigh.ibm.com    53
ServiceList        top_tier

Dispatcher          dispatch1.raleigh.ibm.com  10004
ServiceList        dispatch1_http

Dispatcher          dispatch2.raleigh.ibm.com  10004
ServiceList        dispatch1_http

# End of ISS configuration file -----

```

Ping Triangulation Sample configuration file for ISS

```

# Ping Triangulation Sample Configuration File for ISS
# -----
# This text shows two config files. One should be used by nodes in
# the Hursley cell, which describes Hursley as a local cell and
# Raleigh as a global cell. The other should be used by nodes in the
# Raleigh cell, which describes Raleigh as a local cell and Hursley
# as global.
#
# In this way, a particular machine can communicate with all the other
# nodes in its cell, and knows enough about the remote cells to
# exchange ping information with them, and redirect traffic to them.
# -----
#
# SAMPLE FILE ONE -- for use in cell Hursley. Raleigh is nominated "global",
# because we communicate with the nodes in that cell only at a global,
# wide-area network level

Cell      ITL  global
Node      spnode1.raleigh.ibm.com 1
Node      spnode2.raleigh.ibm.com 2
Node      spnode3.raleigh.ibm.com 3
Node      spnode4.raleigh.ibm.com 4

# Between all the cells, exactly one ISSNameServer should be configured.
# Since the ISSNameServer is a part of the Raleigh (ITL) cell, it must
# be specified in the global definition. The ISSNameserver must also
# be configured on the same machine that has been designated as the monitor.

```

```
ISSNameserver spnode1.raleigh.ibm.com 53
```

```
# For a given service to be redirected globally, we need to know that it is
# available in the global cell we are redirecting to, as well as the local
# cell we are redirecting from.
#
# This matching is done using the service identifying name, which is the first
# parameter after the Service keyword. (It is recommended that this service
# name be derived from the DNS name of the service). The SelectionMethod and
# NodeList are also required. The SelectionMethod must be the same in all cells.
```

```
Service WWWService www.ourservice.com
SelectionMethod Best
NodeList spnode1.raleigh.ibm.com
        spnode2.raleigh.ibm.com
        spnode3.raleigh.ibm.com
        spnode4.raleigh.ibm.com
```

```
# Now we describe the Hursley cell:
```

```
Cell Hursley          local
LogLevel              info
HeartbeatInterval     10
HeartbeatsPerUpdate   3
PortNumber             7139
PingCache             10
```

```
# Individual node data
```

```
#
```

```
# These nodes are all in the same local subdomain: hursley.ibm.com, so we do
# not need to quote their fully qualified DNS names.
```

```
Node spn1 1
Node spn2 3
Node spn3 4
Node spn4 5
```

```
# We are interested in the CPU load on these machines and use the Internal
# CPULoad metric. Since it measures percentage utilization, the limits for
# normalization are 0 and 100.
```

```
ResourceType CPU
Metric Internal CPULoad
MetricNormalization 0 100
Policy Min
```

```
Service WWWService www.ourservice.com
NodeList spn2 spn3 spn4 spn1
ResourceList CPU
SelectionMethod Best
```

```
# --- End of config file for cell Hursley
```

```
# SAMPLE FILE TWO -- for use in cell ITL
```

```
# First we describe the Hursley cell. Since this config file is used in
# Raleigh (ITL) cell, we describe Hursley as a global cell, and do not need
# as much information about a global cell as we do for a local cell. However,
# we do need fully-qualified DNS names for nodes which are not in our local
# local subdomain.
```

```
Cell Hursley global
Node spn1.hursley.ibm.com 1
Node spn2.hursley.ibm.com 2
Node spn3.hursley.ibm.com 3
Node spn4.hursley.ibm.com 4
```

```
# Now we describe the required Service information.
# The selectionmethod and nodelist are also required.
```

```
Service WWWService www.ourservice.com
SelectionMethod Best
NodeList spn1.hursley.ibm.com
        spn2.hursley.ibm.com
        spn3.hursley.ibm.com
        spn4.hursley.ibm.com
```

```
# Finally, we describe ITL, which is a local cell in this section
```

```
Cell ITL local
HeartbeatInterval 10
HeartbeatsPerUpdate 3
PortNumber 7139
PingCache 10
```

```
# Individual node data
Node spnode1.raleigh.ibm.com 1
Node spnode2.raleigh.ibm.com 2
Node spnode3.raleigh.ibm.com 3
Node spnode4.raleigh.ibm.com 4
```

```
# We use a different metric here (AIX specific) - this is only for local use,
# the Hursley cell does not need to know about it
```

```
ResourceType CPU
Metric External vmstat 3 2 | tail -1 | awk '{print $16}'
MetricNormalization 0 100
Policy Max
```

```
# The SelectionMethod is not defined explicitly for this service
# because it will default to BEST.
```

```
Service WWWService www.ourservice.com
ResourceList CPU
NodeList spnode1 spnode2 spnode3 spnode4

ISSNameserver spnode1.raleigh.ibm.com 53
ServiceList WWWService
```



```
# End of config file for cell ITL.
```

```
# End of ISS configuration file -----
```

Appendix F. Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make them available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that IBM product, program or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785, U.S.A.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Site Counsel
IBM Corporation
P.O. Box 12195
3039 Cornwallis Avenue
Research Triangle Park, NC 27709-2195
USA

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement.

This document is not intended for production use and is furnished as is without any warranty of any kind, and all warranties are hereby disclaimed including the warranties of merchantability and fitness for a particular purpose.

This product includes computer software created and made available by CERN. This acknowledgement shall be mentioned in full in any product which includes the CERN computer software included herein or parts thereof.

Trademarks

The following terms are trademarks of IBM Corporation in the United States or other countries or both.

AIX

IBM

IBMLink

LoadLeveler

OS/2

NetView

HP is a trademark of the Hewlett-Packard Company.

Linux is a registered trademark of Linus Torvalds.

Microsoft, Windows 2000, and Windows NT are trademarks or registered trademarks of the Microsoft Corporation.

Red Hat is a registered trademark of Red Hat, Inc.

Solaris is a trademark of Sun Microsystems, Inc.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Tivoli is a non-registered trademark for Tivoli

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

Glossary

A

ACK. A control bit (acknowledge) occupying no sequence space, which indicates that the acknowledgment field of this segment specifies the next sequence number the sender of this segment is expecting to receive, hence acknowledging receipt of all previous sequence numbers.

address. The unique code assigned to each device or workstation connected to a network. A standard IP address is a 32-bit address field. This field contains two parts. The first part is the network address; the second part is the host number.

advisor. The advisors are a function of the Dispatcher component. Advisors collect and analyze feedback from individual servers and inform the manager function.

agent. (1) In systems management, a user that, for a particular interaction, has assumed an agent role. (2) An entity that represents one or more managed objects by (a) emitting notifications regarding the objects and (b) handling requests from managers for management operations to modify or query the objects.

alias. An additional name assigned to a server. The alias makes the server independent of the name of its host machine. The alias must be defined in the domain name server.

API. Application programming interface. The interface (calling conventions) by which an application program accesses operating system and other services. An API is defined at source code level and provides a level of abstraction between the application and the kernel (or other privileged utilities) to ensure the portability of the code.

B

backup. In high availability for the Dispatcher, the partner of the primary machine. It monitors the status of the primary machine and takes over if necessary. See also high availability, primary.

begin range. In rules-based load balancing, a lower value specified on a rule. The default for this value depends on the type of rule.

bottom tier. In a two-tiered IBM Network Dispatcher configuration, that portion of the configuration that describes the relationship between the Network Dispatchers and the clustered servers. See also top tier.

C

CBR. Content Based Routing. A component of IBM Network Dispatcher.

cell. In ISS, a cell is a set of locally managed machines, or *nodes*, close enough to one another on the network and close enough geographically to be administered as a single logical unit.

CGI. Common Gateway Interface. A standard for the exchange of information between a Web server and an external program. The external program can be written in any language supported by the operating system, and performs tasks not usually done by the server, such as forms processing.

CGI script. A CGI program written in a scripting language such as Perl or REXX that uses the Common Gateway Interface to perform tasks not usually done by the server, such as forms processing.

client. A computer system or process that requests a service of another computer system or process. For example, a workstation or personal computer requesting HTML documents from a Lotus Domino Go Webserver is a client of that server.

cluster. In the Dispatcher, a group of TCP or UDP servers that are used for the same purpose and are identified by a single hostname. See also cell.

cluster address. In the Dispatcher, the address to which clients connect.

clustered server. A server that the Dispatcher groups with other servers into a single, virtual server. IBM Network Dispatcher balances TCP or UDP traffic among these clustered servers.

collocate. When you don't have a dedicated machine, Dispatcher is installed on the same machine it is load balancing.

Note: Collocated only applies to the AIX, Red Hat Linux, and Solaris operating systems.

cross port affinity. Cross port affinity is the affinity (sticky) feature expanded to cover across multiple ports. See also sticky time.

D

default. A value, attribute, or option that is assumed when none is explicitly specified.

destination address. The address of the high availability partner machine to which heartbeats and responses are sent.

Dispatcher. A component of IBM Network Dispatcher that efficiently balances TCP or UDP traffic among groups of individual linked servers. The Dispatcher machine is the server running the Dispatcher code. Dispatcher, ISS, and CBR are all called components of IBM Network Dispatcher. Each of these components have functions.

dotted-decimal notation. The syntactical representation for a 32-bit integer that consists of four 8-bit numbers, written in base 10 and separated by periods (dots). It is used to represent IP addresses.

E

end range. In rules-based load balancing, a higher value specified on a rule. The default for this value depends on the type of rule.

executor. One of several Dispatcher functions. The executor routes requests to the TCP or UDP servers, and also monitors the number of new, active, and finished connections and does garbage collection of completed or reset connections. The executor supplies the new and active connections to the manager function.

F

FIN. A control bit (finis) occupying one sequence number, which indicates that the sender will send no more data or control occupying sequence space.

FIN state. The status of a transaction that has finished. Once a transaction is in FIN state, the IBM Network Dispatcher garbage collector can clear the memory reserved for the connection.

Firewall. A computer that connects a private network, such as a business, to a public network, such as the Internet. It contains programs that limit the access between two networks. See also *proxy gateway*.

FTP (File Transfer Protocol). An application protocol used for transferring files to and from network computers. FTP requires a user ID and sometimes a password to allow access to files on a remote host system.

G

gateway. A functional unit that interconnects two computer networks with different architectures.

H

heartbeat. A simple packet sent between two Dispatcher machines in high availability mode used by the standby Dispatcher to monitor the health of the active Dispatcher.

high availability. A Dispatcher feature in which one Dispatcher can take over the function of another, should that part fail.

host. A computer, connected to a network, that provides an access point to that network. A host can be a client, a server, or both simultaneously.

host name. The symbolic name assigned to a host. Host names are resolved to IP addresses through a domain name server.

HTML. Hypertext Markup Language. The language used to create hypertext documents. Hypertext documents include links to other documents that contain additional information about the highlighted term or subject. HTML controls the format of text and position of form input areas, for example, as well as the navigable links.

HTTP (Hypertext Transfer Protocol). The protocol used to transfer and display hypertext documents.

I

Internet. The worldwide collection of interconnected networks that use the Internet suite of protocols and permit public access.

ICMP. Internet Control Message Protocol. A message control and error-reporting protocol between a host server and a gateway to the Internet.

IMAP. Internet Message Access Protocol. A protocol allowing a client to access and manipulate electronic mail messages on a server. It permits manipulation of remote message folders (mailboxes), in a way that is functionally equivalent to local mailboxes.

intranet. A secure, private network that integrates Internet standards and applications (such as Web browsers) with an organization's existing computer networking infrastructure.

IP. Internet Protocol. A connectionless protocol that routes data through a network or interconnected networks. IP acts as an intermediary between the higher protocol layers and the physical layer.

IP address. Internet Protocol address. The unique 32-bit address that specifies the actual location of each device or workstation in a network. It is also known as an Internet address.

IPSEC. Internet Protocol Security. A developing standard for security at the network or packet processing layer of network communication.

ISS. Interactive Session Support. A component of IBM Network Dispatcher.

L

LAN. Local Area Network. A computer network of devices connected within a limited geographical area for communication and which can be connected to a larger network.

loopback alias. An alternative IP address associated with the loopback interface. The alternative address has the useful side effect of not advertising on a real interface.

loopback interface. An interface that bypasses unnecessary communications functions when the information is addressed to an entity within the same system.

M

MAC address. A LAN or a LAN emulation concept.

managed node. In Internet communications, a workstation, server, or router that contains a network management agent. In the Internet Protocol (IP), the managed node usually contains a Simple Network Management Protocol (SNMP) agent.

manager. One of several Dispatcher functions. The manager sets weights based on internal counters in the executor and feedback provided by the advisors. The executor then uses the weights to perform load balancing.

mark down. To break all active connections to a server and stop any new connections or packets from being sent to that server.

mark up. To allow a server to receive new connections.

metric. In ISS, a process or command that returns a numeric value that can be used in load balancing on the network, for example, the number of users currently logged on.

MIB. (1) Management Information Base. A collection of objects that can be accessed by means of a network management protocol. (2) A definition for management information that specifies the information available from a host or gateway and the operations allowed.

multiple address collocation. Multiple address collocation allows the customer to specify the address of the collocated server to be different than the nonforwarding address (NFA) in the configuration. See also collocate.

mutual high availability. Mutual high availability allows two Dispatcher machines to be both primary and backup for each other. See also backup, high availability, primary.

N

netmask. For Internet subnetting, a 32-bit mask used to identify the subnetwork address bits in the host portion of an IP address.

network management station. In the Simple Network Management Protocol (SNMP), a station that executes management application programs that monitor and control network elements.

NNTP. Network News Transfer Protocol. A TCP/IP protocol for transferring news items.

node. In ISS, a single machine, which may have one or more IP addresses over time. A node is a member of one and only one *cell*.

nonforwarding address (nfa). The primary IP address of the Network Dispatcher machine, used for administration and configuration.

O

observer. In ISS, a process that subscribes to load-balancing reports on a given resource.

P

packet. The unit of data that is routed between an origin and a destination on the Internet or any other packet-switched network.

PICS. Platform for Internet Content Selection. PICS-enabled clients allow the users to determine which rating services they want to use and, for each rating service, which ratings are acceptable and which are unacceptable.

ping. A command that sends Internet Control Message Protocol (ICMP) echo-request packets to a host, gateway, or router with the expectation of receiving a reply.

POP3. Post Office Protocol 3. A protocol used for exchanging network mail and accessing mailboxes.

port. A number that identifies an abstracted communication device. Web servers use port 80 by default.

primary. In high availability for the Dispatcher, the machine that starts out as the machine actively routing packets. Its partner, the backup machine, monitors the status of the primary machine and takes over if necessary. See also backup, high availability.

priority. In rules-based load balancing, the level of importance placed upon any given rule. The Dispatcher evaluates rules from the first priority level to the last priority level.

private network. A separate network on which Dispatcher communicates with clustered servers for performance reasons.

protocol. The set of rules governing the operation of functional units of a communication system if communication is to take place. Protocols can determine low-level details of machine-to-machine

interfaces, such as the order in which bits from a byte are sent; they can also determine high-level exchanges between application programs, such as file transfer.

Q

quiesce. To end a process by allowing operations to complete normally.

R

reach. In Dispatcher, an advisor that issues pings to a given target and reports whether that target is responding.

reach address. In high availability for the Dispatcher, the address of the target to which the advisor should issue pings to see if the target is responding.

resource. In ISS, a specific device needed to provide a service.

resourceType. In ISS, a formalized category to which various resources can belong. The Resource Type can be configured once and its parameters will subsequently apply to all resources belonging to it.

RMI. Remote Method Invocation. A method used by a programmer, using the Java programming language and development environment, can write object-oriented programs in which objects on different computers can interact in a distributed network.

root user. The unrestricted authority to access and modify any part of the AIX, Red Hat Linux, or Solaris operating system, usually associated with the user who manages the system.

route. The path of network traffic from origin to destination.

RPM. Red Hat Package Manager.

rule. In rules-based load balancing, a mechanism for grouping servers such that a server can be chosen based on information other than the destination address and port.

rule type. In rules-based load balancing, an indicator of the information that should be evaluated to determine whether a rule is true.

S

scalable. Pertaining to the capability of a system to adapt readily to a greater or lesser intensity of use, volume, or demand. For example, a scalable system can efficiently adapt to work with larger or smaller networks performing tasks of varying complexity.

server. A computer that provides shared services to other computers over a network; for example, a file server, a print server, or a mail server.

server address. The unique code assigned to each computer that provides shared services to other computers over a network; for example, a file server, a print server, or a mail server. A standard IP address is a 32-bit address field. The server address can be either the dotted decimal IP address or the host name.

server machine. A server that the Dispatcher groups with other servers into a single, virtual server. The Dispatcher balances traffic among the server machines. Synonymous with clustered server.

service. A function provided by one or more nodes; for example, HTTP, FTP, Telnet.

shell. The software that accepts and processes command lines from a user's workstation. The Korn shell is one of several UNIX shells available.

SMA. Server Monitor Agent. The system monitor agent which is installed on servers running on a Red Hat Linux platform. SMA provides system specific metrics to the Dispatcher manager.

SMTP. Simple Mail Transfer Protocol. In the Internet suite of protocols, an application protocol for transferring mail among users in the Internet environment. SMTP specifies the mail exchange sequences and message format. It assumes that the Transmission Control Protocol (TCP) is the underlying protocol.

SNMP. Simple Network Management Protocol.

source address. In high availability for the Dispatcher, the address of the high availability partner machine that sends heartbeats.

SPARC. Scalable processor architecture.

SSL. Secure Sockets Layer. A popular security scheme developed by Netscape Communications Corp. along with RSA Data Security Inc. SSL allows the client to authenticate the server and all data and requests to be encrypted. The URL of a secure server protected by SSL begins with https (rather than http).

strategy. In high availability for the Dispatcher, a keyword for specifying how recovery takes place following the failure of the active machine.

sticky time. The interval between the closing of one connection and the opening of a new connection during which a client will be sent back to the same server used during the first connection. After the sticky time, the client may be sent to a server different from the first.

subnet mask. For Internet subnetworking, a 32-bit mask used to identify the subnetwork address bits in the host portion of an IP address.

SYN. A control bit in the incoming segment, occupying one sequence number, used at the initiation of a connection, to indicate where the sequence numbering will start.

T

TCP. Transmission Control Protocol. A communications protocol used on the Internet. TCP provides reliable host-to-host exchange of information. It uses IP as the underlying protocol.

TCP/IP. Transmission Control Protocol/Internet Protocol. A suite of protocols designed to allow communication between networks regardless of the communication technologies used in each network.

TCP server machine. A server that IBM Network Dispatcher links with other servers into a single, virtual server. IBM Network Dispatcher balances TCP traffic among the TCP server machines. Synonymous with clustered server.

Telnet. Terminal emulation protocol, a TCP/IP application protocol for remote connection service. Telnet allows a user at one site to gain access to a remote host as if the user's workstation were connected directly to that remote host.

timeout. The time interval allotted for an operation to occur.

top tier. In a two-tiered IBM Network Dispatcher configuration, that portion of the configuration that describes the relationship between the ISS Monitor and the Network Dispatchers. See also bottom tier.

TOS. Type of service. A one byte field in the IP header of the SYN packet.

two-tiered configuration. An implementation of a network using two functions of IBM Network Dispatcher. The Interactive Session Support (ISS) Monitor in the *top* tier manages the routing of client requests to the Dispatcher machines in the *bottom* tier. The Dispatcher machines route client requests to the clustered servers.

U

UDP. User Datagram Protocol. In the Internet suite of protocols, a protocol that provides unreliable, connectionless datagram service. It enables an application program on one machine or process to send a datagram to an application program on another machine or process. UDP uses the Internet Protocol (IP) to deliver datagrams.

V

VPN. Virtual Private Network (VPN). A network comprised of one or more secure IP tunnels connecting two or more networks.

W

Web. The network of HTTP servers that contain programs and files, many of them hypertext documents that contain links to other documents on HTTP servers. Also World Wide Web.

WTE. Web Traffic Express (WTE). A caching proxy server that can help speed up end-user response time through highly-efficient caching schemes. Flexible PICS filtering helps network administrators control access to Web-based information at one central location.

wizard. A dialog within an application that uses step-by-step instructions to guide a user through a specific task.

WLM. Workload Manager. An advisor provided with Dispatcher. It is designed to work only in conjunction with servers on OS/390 mainframes running the MVS Workload Manager (WLM) component.

Index

A

- adding
 - cluster 197, 199
 - port to a cluster 59, 219, 221
 - server to a port 59, 228, 230
- address mapping file
 - example of 114
- advisors, Network Dispatcher function
 - customize 102
 - interval for 87, 193, 195
 - list of 100, 193, 194, 195
 - name of 193
 - port for 193
 - proportion of importance in load balancing 214
 - report on the state of 194, 195
 - starting 60, 194, 196
 - stopping 194, 196
 - timeout period for a report 87, 195, 196
 - version of 195, 196
 - WTE advisor 101
- affinity (sticky)
 - affinity address mask 26, 119
 - client IP affinity (CBR) 121
 - cookie affinity (stickytime) 120
 - cross port affinity 26, 118, 119, 219
 - how it works 116
 - rule affinity override 27, 119
 - SDA (Server Directed Affinity) 117
 - sticky (rule affinity override) 119, 120, 229
 - StickyGarbageCollectionInterval (ISS) 248
 - stickymask 118, 119, 220
 - stickytime 116, 118, 220
 - StickyTime (ISS) 140, 248
- affinity address mask 26, 119, 220
- alarm 150
- ALARM keyword, ISS 238
- alias
 - the loopback device 61
 - Linux kernel patch 61, 65
 - the network interface card 57
- Authkey keyword, ISS 238

B

- backup, high availability 48, 207
 - configuring 89
- best configuration, ISS 131, 132
- binary logging and statistics 121, 161
- boot file 153

C

- CBR component
 - CBR proxy (for IMAP or POP3) 26, 70
 - cbrserver 70
 - configuring 80
 - inactivity timeout 221
 - proxy protocol 221, 222
 - staletimeout 221
 - CBR w/WTE (HTTP) 71
 - configuring 80
 - configuration
 - overview of tasks 73
 - setting up the CBR machine 75
 - load-balancing settings 84
 - planning considerations 69
 - starting and stopping 171
- cbrkeys 162
- cell 146
- Cell, ISS 239
- cells, ISS
 - definition 127
 - measurement types
 - custom 130
 - LoadLeveler 130
 - round-robin 130
 - setting up 127
- changing
 - FIN count 159
 - FIN time-out 159
 - stale timer 160
- checking for
 - extra route 64
- cluster
 - adding 197, 199
 - changing the FIN count for 159
 - changing the FIN time-out for 159
 - configure the address 57
 - defining 197, 199
- cluster (*continued*)
 - displaying
 - status of this cluster 199
 - removing 198, 199
 - wildcard 57
- collocate, Network Dispatcher and server 27, 50, 55, 59, 60, 180, 228, 230
- collocated (keyword) 50, 59, 228, 230
- commands
 - ifconfig 58, 96
 - to alias the loopback device 61
 - ndconfig 58, 96
 - ndcontrol
 - to configure a remote host 211
 - to configure high availability 207
 - to configure rules 223
 - to control the advisor 60, 193
 - to control the binary log 212
 - to control the executor 200
 - to control the manager 60, 213
 - to define a cluster 197
 - to define a port 59, 219
 - to define a server 59, 228
 - to define the nonforwarding address 56, 200, 203
 - to display if the manager and advisors are running 232
 - to get help 206
 - to save the current configuration to a file 204
- netstat
 - to check IP addresses and aliases 64
- route
 - to delete an extra route 64, 65
- configuration file, creating 146
- configuration wizard 4
- configuring Dispatcher
 - planning your configuration 44
- connections, setting proportion of importance 85, 214, 216
- Content based routing 25

cookie affinity (CBR stickytime) 120
cross port affinity 26, 118, 219
custom (customizable) advisor 102

D

data file 154
default.cfg 56
defining
 cluster 197, 199
 nonforwarding address 56, 200, 203
 port to a cluster 59, 219, 221
 server to a port 59, 228, 230
deleting
 cluster 198, 199
 extra route 65
 port from a cluster 221, 222
 server from a port 230
diagnosing problems
 advisors not working 179
 CBR will not run 181
 cbrserver command is
 stopped 182
 common problems and
 solutions 178, 180, 181, 182
 could not find local node in the
 config file 184
 Dispatcher, Microsoft IIS, and
 SSL do not work 179
 Dispatcher and server will not
 respond 178
 Dispatcher high availability not
 working 178
 Dispatcher requests not
 routed 178
 Dispatcher will not run 178
 error message when trying to
 view online Help 180
 extra routes 179
 ISS file problems 183
 ISS not recognising isscontrol or
 GUI commands 183
 ISS sending loads/ Dispatcher
 reports wrong values 184
 ISS session problems 183
 ISS will not run 182
 Network Dispatcher machine
 locks up when attempting to
 collocate 180
 port numbers used by CBR 177
 port numbers used by ISS 176
 port numbers used by the
 Dispatcher 176
 receive CBR proxy error when
 trying to add a port 182

diagnosing problems (*continued*)
 server not responding (CBR
 w/WTE) 181
 SNMPD not working 179
 starting ISS from the Services
 panel produces an error 182
 Syntactical or configuration
 error 181
 The ndcontrol or ndadmin
 command fails 180
 trace and debug facilities for
 ISS 184
 Unable to add heartbeat 178
 unable to get ISS GUI to
 "Connect to Host" 182
dispatcher cluster address 150
Dispatcher function
 configuration
 setting up a private
 network 113
 guidelines for using 31
 load-balancing settings 84
 advisor intervals 87
 advisor timeouts 87
 manager intervals 86
 proportion of importance
 given to status
 information 85
 sensitivity threshold 88
 smoothing index 88
 weights 86
Dispatcher keyword, ISS 239
dispatcher port 150
displaying
 global values and their default
 settings
 for an advisor 194, 196
 for the manager 216, 218
 internal counters 200, 202
 list of
 advisors currently providing
 metrics 193, 195
 report on the state of an
 advisor 194, 195
 statistics report 215, 216
 status of
 a cluster or all clusters 199
 servers on a port 221, 222
 version number
 of advisor 195, 196
 of manager 216, 218
DNS—Ignore mode, ISS 129
DNS—Replace mode, ISS 129
DNS—Update mode, ISS 129
down, marking a server as 229, 230

DPID2 164

E

ending an interactive session 170
establishing an interactive
 session 170
Ethernet NIC
 ibmnd.conf
 configuring for Solaris 55
examples
 managing local and remote
 servers 40
 managing local servers 33, 34,
 35, 37, 38
 quick start 1
executor
 starting 202, 203
 stopping 202, 203
explicit linking 113
external metrics 132
extra routes 64

F

FIN count 159
FIN count limit
 changing 159
FIN time out
 changing 159
ftp advisor 193

G

garbage collection 159
getting help 206
goActive 93
goldle 94
goInOp 93
goStandby 93
graphical user interface (GUI) 5
GUI 5

H

HeartbeatInterval keyword, ISS 239
HeartbeatsPerUpdate keyword,
 ISS 240
HeartbeatsToNetFail keyword,
 ISS 241
HeartbeatsToNodeFail keyword,
 ISS 241
help, how to get it 206
high availability 25, 48, 88, 207
 configuring 89
 mutual 26, 49, 90, 198, 199, 209
 primaryhost 198, 199
HP-UX, setting up TCP servers
 running 61
http advisor 193

- ibmnd.conf
 - configuring for Solaris 55
- ifconfig 96
- ifconfig command 58, 61
- IMAP advisor 27
- installing
 - on AIX 10
 - on Red Hat Linux 14
 - on Solaris 17
 - on Windows 21
- internal metrics 132
- InternalNet eyword, ISS 241
- interval, setting how often
 - the advisor queries the servers 87, 193, 195
 - the manager queries the executor 87, 214, 216
 - the manager updates the weights to the executor 86, 213, 216
- ISS cells
 - definition 127
 - measurement types
 - custom 130
 - LoadLeveler 130
 - round-robin 130
 - setting up 127
- iss.cfg 143, 144, 145, 146
- ISS function
 - boot file 153
 - configuration file, updating 146
 - configuration methods
 - best 131
 - round-robin 131
 - statistical round-robin 131
 - data file 154
 - ending an interactive session 170
 - establishing an interactive session 170
 - guidelines for using 31
 - ISS backup, configuring 157
 - iss.cfg 143, 144, 145, 146
 - ISS client machines,
 - configuring 156
 - overview 29, 143
 - planning considerations 127
 - reconfiguring 170
 - reverse data file 155
 - starting 168
 - starting the TCP/IP name server 155
 - stopping 170
 - TCP/IP name server, setting up 151

- ISS function (*continued*)
 - trace and debug facilities 184

- ISS keywords
 - ALARM 238
 - Authkey 238
 - Cell 239
 - Dispatcher 239
 - HeartbeatInterval 239
 - HeartbeatsPerUpdate 240
 - HeartbeatsToNetFail 241
 - HeartbeatsToNodeFail 241
 - InternalNet 241
 - ISSNameServer 242
 - LogLevel 242
 - Metric 242
 - MetricLimits 243
 - MetricNormalization 243
 - NamedData 243
 - NamedRev 244
 - NameServer 244
 - Node 244
 - NodeList 244
 - NodeWeight 245
 - NotISSAgent 245
 - NotMonitor 245
 - Observer 245
 - Overflow 245
 - PingCache 246
 - Policy 246
 - PortNumber 246
 - Resources 247
 - ResourceType 247
 - SelectionMethod 247
 - Service 248
 - ServiceList 248
 - StickyGarbageCollection
 - Interval 248
 - StickyTime 248
- isskeys 162
- ISSNameServer, ISS 242

J

- Java development kit (JDK) 9, 41, 67, 125
- Java runtime environment (JRE) 14, 17, 19, 42, 68, 69, 126

K

- keys
 - cbrkeys 162
 - isskeys 162
 - ndkeys 162
- keyword
 - alarm 150
 - dispatcher cluster address 150
 - dispatcher port 150

- keyword (*continued*)
 - metriclimits 148
 - metricnormalization 148
 - metricweights 149
 - odelist 149
 - overflow 150
 - policy 148
 - resourcelist 149
 - selectionmethod 149

- keywords, ISS
 - ALARM 238
 - Authkey 238
 - Cell 239
 - Dispatcher 239
 - HeartbeatInterval 239
 - HeartbeatsPerUpdate 240
 - HeartbeatsToNetFail 241
 - HeartbeatsToNodeFail 241
 - InternalNet 241
 - ISSNameServer 242
 - LogLevel 242
 - Metric 242
 - MetricLimits 243
 - MetricNormalization 243
 - NAME 146
 - NamedData 243
 - NamedRev 244
 - NameServer 244
 - Node 244
 - NodeList 244
 - NodeWeight 245
 - NotISSAgent 245
 - NotMonitor 245
 - Observer 245
 - Overflow 245
 - PingCache 246
 - Policy 246
 - PortNumber 246
 - Resources 247
 - ResourceType 247
 - SelectionMethod 247
 - Service 248
 - ServiceList 248
 - StickyGarbageCollection
 - Interval 248
 - StickyTime 248

L

- load-balancing settings
 - (optimizing) 84
- log
 - binary, for server statistics 121
 - file, setting the name of
 - for the advisor 194
 - for the manager 215

- log (*continued*)
 - level, setting
 - for the advisor 160, 193, 195
 - for the manager 213
 - size, setting
 - for the advisor 160, 193, 195
 - for the manager 213, 216
- logging and statistics 121
- LogLevel keyword, ISS 242

M

- manager
 - fixed weight 27, 86
 - starting 60, 215, 218
 - stopping 216, 218
 - version of 216, 218
- marking a server as being
 - down 229, 230
 - up 230
- maximum weight, setting
 - for servers on a specific port 86, 221, 222
- metric 146, 148
- Metric 242
- metric configuration
 - external metrics 132
 - internal metrics 132
- metric configuration, ISS 132
- metric type
 - metric 148
- metriclimits 148
- MetricLimits keyword, ISS 243
- metricnormalization 148
- MetricNormalization 243
- metrics for ISS 130
- metricweight 149
- migrating 9
- migration, ISS 150
- multiple address collocation 27, 59
- mutual high availability 26, 49, 89, 90
 - primaryhost 198, 199
 - scripts 93
 - takeover 92

N

- NAME keyword, ISS 146
- named.boot file 153
- named.data file 154
- named.rev file 155
- NamedData eyword, ISS 243
- NamedRev keyword, ISS 244
- NameServer, ISS 244
- ndconfig 96
- ndconfig command 58

- ndcontrol command
 - advisor 60, 193
 - cluster 197
 - executor 56, 200
 - file 204
 - help 206
 - highavailability 207
 - host 211
 - log 212
 - manager 60, 213
 - port 59, 219
 - rule 223
 - server 59, 228
 - status 232
- ndkeys 107, 162
- netstat command 64
- Network Dispatcher
 - benefits 25
 - configuration examples 31
 - configuring
 - Dispatcher function 55, 75
 - ISS function 143
 - functions 23, 28
 - guidelines for using 31
 - hardware requirements 41, 67, 125
 - installing 9
 - operating and managing 159, 167
 - overview 23
 - planning considerations 31, 41, 125
 - quick start example 1
 - software requirements 41, 67, 125
 - troubleshooting 173
- Network Dispatcher function
 - configuration
 - overview of tasks 53
 - setting up the Network Dispatcher machine 55
 - setting up the TCP server machines 60
 - overview 28
 - stopping 159
- new connections, setting proportion of importance 85, 214
- new features 26
 - affinity address mask 26
 - CBR wizard 26
 - cross port affinity 26
 - IMAP advisor 27
 - IMAP/POP3 CBR proxy 26
 - manager fixed weights 27
 - multiple address collocation 27

- new features 26 (*continued*)
 - mutual high availability 26
 - rule affinity override 27
 - type of service (TOS) rule 27
 - WAS (WebSphere Application Server) advisor 27
 - Windows 2000 support 26
- NIC
 - alias 57
 - ethernet (for Solaris) 55
 - mapping (for Windows) 58
- Node, ISS 244
- odelist 127, 149
- NodeList keyword, ISS 244
- NodeWeight Iss Keyword, ISS 245
- nonforwarding address
 - defining 56
 - setting 200, 203
- NotISSAgent keyword, ISS 245
- NotMonitor keyword, ISS 245
- NT, setting up TCP servers running on 61

O

- Observer keyword, ISS 245
- OS/2, setting up TCP servers running 61
- overflow 150
- Overflow keyword, ISS 245

P

- passive FTP, setting a port for 59
- Ping triangulation, ISS 139
- PingCache keyword, ISS 246
- planning considerations 127
- planning for installation 23, 41, 125
- policy 148
- Policy keyword, ISS 246
- pool 127
- portnumber 146
- PortNumber keyword, ISS 246
- ports
 - adding 219, 221
 - defining to a cluster 59, 219, 221
 - displaying
 - status of servers on this port 221, 222
 - for advisors 193
 - removing 221, 222
 - setting the maximum weight 86, 221, 222
 - wildcard 59
- primaryhost 90, 198, 199
- private key
 - for remote authentication 161

- private network, using with
 - Dispatcher 113
- product components 43
- proportion of importance for load balancing, setting 85, 214, 216
- public key
 - for remote authentication 161

Q

- quick start example 1
- quiescing a server 214, 216

R

- remote administration 18, 161
- remove
 - cluster 198, 199
 - extra route 65
 - port from a cluster 221, 222
 - server from a port 230
- requirements
 - AIX 9, 41, 67
 - Red Hat Linux 14, 41, 67
 - Solaris 17, 42, 68
 - Windows 19, 42, 68, 126
- resource 146
- resourcelist 149
- ResourceList 127
- Resources keyword, ISS 247
- resourcetype 146
- ResourceType keyword, ISS 247
- restart all servers to normalized weights 215, 217
- reverse data file 155
- RMI (Remote Method Invocation) 161
- round-robin configuration, ISS 131, 132
- route command 64, 65
- rule affinity override
 - server 27, 119, 229, 230
- rules-based load balancing 108
 - active connections to port 110
 - always true 111
 - client IP address 109
 - client port 111
 - connections per second 110
 - content of request 112
 - time of day 109
 - type of service (TOS) 27, 111

S

- saving the current configuration to a file 204
- scripts 92
- SDA (Server Directed Affinity) 117
- Secure Sockets Layer 28, 59, 220

- selection method 130
- selectionmethod 149
- SelectionMethod keyword, ISS 247
- sensitivity to weights update, setting 88, 215, 217
- server
 - adding 228, 230
 - collocated 228, 230
 - defining to a port 59, 228, 230
 - fixedweight 229
 - marking as being down 229, 230
 - marking as being up 230
 - nonsticky (rule affinity override) 27, 229, 230
 - quiescing 214, 216
 - removing 230
 - restarting all to normalized weights 215, 217
 - setting the weight 230
 - unquiescing 216, 218
- Server Directed Affinity (SDA) 117
- Server Monitor Agent (SMA) 24, 28
 - configuration example 38
 - ndcontrol manager
 - systemscript option 216
 - overview 106
- Service keyword, ISS 248
- Servicelist keyword, ISS 248
- setting
 - cluster address 59
 - how often the manager should query the executor 87, 214, 216
 - interval time
 - for the advisor to query the servers 87, 193, 195
 - for the manager to update the executor 86, 213, 216
 - logging level
 - for the advisor 160, 193, 195
 - for the manager 213
 - maximum size of the log
 - for the advisor 160, 193, 195
 - for the manager 213, 216
 - maximum weight
 - for servers on a specific port 86, 221, 222
 - name of log file 194
 - for the manager 215
 - nonforwarding address 55
 - proportion of importance in load balancing 214, 216
 - sensitivity to weights update 88, 215, 217
 - smoothness index 88, 215, 217

- setting (*continued*)
 - timeout for an advisor
 - report 87, 195, 196
 - weight for a server 214, 216, 218, 230
- settings, displaying all global values
 - for an advisor 194, 196
 - for the manager 216, 218
- showing
 - global values and their default settings
 - for an advisor 194, 196
 - for the manager 216, 218
 - internal counters 200, 202
 - list of
 - advisors currently providing metrics 193, 195
 - report on the state of an advisor 194, 195
 - statistics report 215, 216
 - status of
 - a cluster or all clusters 199
 - servers on a port 221, 222
 - version number
 - of advisor 195, 196
 - of manager 216, 218
- Simple Network Management Protocol (SNMP) 162
- smoothing index 88
- smoothness index, setting 88, 215, 217
- SNMP 162
- Solaris, setting up TCP servers running 61
- SSL 28, 59, 220
- stale timer, changing 160
- starting
 - advisor 60, 194, 196
 - executor 202, 203
 - ISS 168
 - manager 60, 215, 218
 - Network Dispatcher 159
- statistical round-robin configuration 131
- statistics snapshot report, displaying 215, 216
- status, displaying
 - all cluster 199
 - one cluster 199
 - servers on a specific port 221, 222
 - whether the manager and advisor are running 232
- sticky (affinity)
 - affinity address mask 26, 119

- sticky (affinity) (*continued*)
 - client IP affinity (CBR) 121
 - cookie affinity (stickytime) 120
 - cross port affinity 26, 118, 119, 219
 - how it works 116
 - rule affinity override 27, 119
 - SDA (Server Directed Affinity) 117
 - sticky (rule affinity override) 119, 120, 229
 - StickyGarbageCollectionInterval (ISS) 248
 - stickymask 118, 119, 220
 - stickytime 116, 118, 220
 - StickyTime (ISS) 140, 248
- stopping
 - advisor 194, 196
 - executor 202, 203
 - ISS 170
 - manager 216, 218
 - Network Dispatcher 159
- subagents 163
- syntax diagram, reading 187
- system metrics, setting proportion of importance 85, 214

T

- timeout, setting for an advisor 87, 195, 196
- trademarks 276
- troubleshooting

- advisors not working 179
- CBR will not run 181
- cbrserver command is
 - stopped 182
- common problems and solutions 178, 180, 181, 182
- could not find local node in the config file 184
- Dispatcher, Microsoft IIS, and SSL do not work 179
- Dispatcher and server will not respond 178
- Dispatcher high availability not working 178
- Dispatcher requests not routed 178
- Dispatcher will not run 178
- error message when trying to view online Help 180
- extra routes 179
- ISS file problems 183
- ISS not recognising isscontrol or GUI commands 183

- troubleshooting (*continued*)
 - ISS sending loads/ Dispatcher reports wrong values 184
 - ISS session problems 183
 - ISS will not run 182
 - Network Dispatcher machine
 - locks up when attempting to collocate 180
 - port numbers used by CBR 177
 - port numbers used by ISS 176
 - port numbers used by the Dispatcher 176
 - receive CBR proxy error when trying to add a port 182
 - server not responding (CBR w/WTE) 181
 - SNMPD not working 179
 - starting ISS from the Services panel produces an error 182
 - Syntactical or configuration error 181
 - The ndcontrol or ndadmin command fails 180
 - trace and debug facilities for ISS 184
 - Unable to add heartbeat 178
 - unable to get ISS GUI to "Connect to Host" 182

U

- up, marking a server as 230

V

- version, displaying
 - advisor 195, 196
 - manager 216, 218

W

- WAS (WebSphere Application Server) advisor 27, 103
- weight
 - how the manager sets 86
 - setting
 - boundary for all servers on a port 86, 221, 222
 - for a server 230
- wide area support 94
 - and remote advisors 96
- wildcard cluster 57, 199
 - to combine server configurations 114
 - to load balance firewalls 115
 - with WTE for transparent proxy 116
- wildcard port 59, 222
 - ping advisor 101

- wildcard port 59, 222 (*continued*)
 - to direct unconfigured port traffic 116
- Windows, setting up TCP servers running on 61
- wizard, configuring
 - CBR 26
 - Dispatcher 4
- workload manager advisor (WLM) 105
- WTE 72
 - configure for CBR 76
- WTE advisor 101

Readers' Comments — We'd Like to Hear from You

IBM® Network Dispatcher
User's Guide
Version 3.0 for Multiplatforms

Publication No. GC31-8496-05

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Complete	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to find	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to understand	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Well organized	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Applicable to your tasks	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.



Fold and Tape

Please do not staple

Fold and Tape



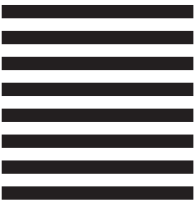
NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Information Development
Department CGMD / Bldg 500
P.O. Box 12195
Research Triangle Park, NC
27709-9990



Fold and Tape

Please do not staple

Fold and Tape



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC31-8496-05

