Netscape Proxy Server Administrator's Guide

Document Number 007-2911-001

Netscape Proxy Server Administrator's Guide
Document Number 007-2911-001

Contents

List of Examples

List of Figures

List of Tables

Chapter 1

Introduction

Welcome to the world of WebFORCE™. You're about to embark on a fascinating journey. You've chosen the best network and Internet proxy server available, and you'll discover it's quite easy to configure, install, and manage.

What is the Netscape Proxy Server?

The Netscape Proxy Server™ provides access to the Internet and World Wide Web through a firewall machine. The Netscape Proxy Server is the first commercial caching proxy server. It is fully compatible with the Netscape Commerce and Communications Servers, the NCSA httpd server, and other HTTP clients and servers. It offers a powerful alternative to the CERN proxy server.

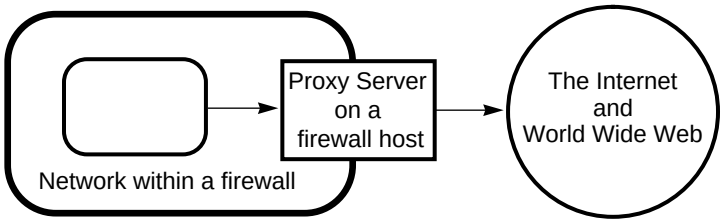

Figure 1-1 Netscape Proxy Server and Firewall

Following is a partial list of features the Netscape Proxy Server offers:

- The Netscape Proxy Server provides safe passage through the firewall, including proxying the secure protocols (SSL, HTTPS, NNTPS).

- The proxy caches documents by copying them to a local filesystem. This makes subsequent client transactions much faster because the proxy doesn't always contact the remote server. This can also dramatically reduce network traffic and the costs associated with it.
- The Netscape Proxy Server was designed with performance and extensibility in mind, so it handles very high numbers of incoming requests with a low impact on the host machine.
- The proxy can filter client transactions by controlling access to remote servers and protocols, and by limiting access to specific documents based on URLs and client hostnames and user names.
- The proxy provides high-level logging of client transactions, including client hostnames or IP addresses, access dates and times, accessed URLs, byte counts of all the transferred data, and success codes per transaction.

What's In This Book?

This manual explains how to configure the Netscape Proxy Server. After you configure your proxy server, you can use this manual to help you maintain your server and learn how the proxy works internally.

System Requirements

The Netscape Proxy Server needs specific software and hardware. Before you can install the Netscape Proxy Server, your computer must meet or exceed the following requirements:

- IRIX™ 5.2 operating system.
- Minimum 32 MB RAM, 64 MB or more is recommended for systems that will handle a lot of traffic.
- 5 MB hard disk space for the server, plus 2-3 MB for log files.
- 1-3 GB recommended hard disk space for the cache directory (3-5 GB for extremely high-impact sites).
- A forms-capable browser (such as Netscape Navigator™).

Some Conventions Used in This Guide

This document uses the following conventions:

Special information that you might not otherwise see or pay any attention to is preceded by **Note:**.

Characters that you type on your keyboard appear in bold type and look **`like this`**.

1. Steps to be followed in sequence have numbers like the one at the beginning of this sentence.

File and pathnames, and Uniform Resource Locators (URLs) in text appear in *italics*, for example */etc/hosts* and *http://www.sgi.com*. Variables that are used in syntax expression, descriptions of syntax expressions, and online button names alos appear in italics.

Items that appear between brackets like **[***yourdomain***]** should be replaced with a value or expression specific to what you're doing or specific to your site.

Chapter 2

Installing the Netscape Proxy Server

This chapter describes how to prepare to install Netscape Proxy Server software on your system, and what the installation does. Your software release notes provide the actual software installation procedure.

Before You Begin Installing

Before you install the Netscape Proxy Server, make sure you have the following items prepared. This makes the installation process much smoother.

Make Sure DNS is Up and Running

When you install the Netscape Proxy Server, some items on the installation forms request either a hostname or an IP address (or multiple entries of the same) as input strings.

- A hostname is a name for a specific computer in the form *system_name.subdomain_name.domain_name*, which is translated into a dotted IP address by a Domain Name Service (DNS). For example, www.sgi.com is the system www in the subdomain sgi in the domain com.
- Internet Protocol (IP) address is a set of numbers, separated by dots, that specifies the actual location of a system on the Internet. For example, the hostname www.sgi.com has the IP address 192.82.208.8.

As you prepare for installation, make sure your Domain Name Service (DNS) is up and running properly. Otherwise, the proxy server can't resolve hostnames and can't connect to any remote hosts.

Create an Alias for the Server

If your server will run on one system among many in a network, you or your system administrator should set up a DNS CNAME record or an alias (such as "proxy") that points to the actual proxy server system. Later, should the need arise, you can change the actual hostname or IP address of the server host without having all the proxy's clients change their browsers.

Create an IRIX User Account

You need to be logged in as root (or superuser) to install the server. However, you don't necessarily want the server to run as root all the time. You probably want the server to have restricted access to your system resources and run under a non-privileged system user account. In this case, you need to create an IRIX user account for the server. When the server launches, it runs as this user. Likewise, any child processes of the server are created with this server user as the process owner.

You can choose the user "nobody" if you wish, but this might not work on some systems. Some systems ship with a UID of -2 for the user "nobody". A UID less than zero generates an error during installation. Check the */etc/passwd* file to see if the UID for "nobody" exists and that it is greater than zero.

If you'd prefer to use an account other than nobody, just create and use a regular IRIX user account. (If you don't know how to create a new user on your system, refer to the *Personal System Administration Guide*.)

Caution: We strongly recommend that you use a dedicated account to further ensure system security.

Choose a Unique Port Number

Port numbers for all network-accessible services are maintained in the file */etc/services*. The standard HTTP port number is 80, but currently there isn't a standard port number for proxies. Commonly used ports are 8000 and 8080. If you use the Netscape Proxy Server's built-in SOCKS daemon, you should use the SOCKS port 1080.

Make sure the port you choose isn't in use. Look at the file */etc/services* on the server system to make sure you don't assign a port number that is used by another service.

If you choose a port number less than 1024, you'll need to be logged in as root or superuser to start the proxy server. After the proxy is bound to the port, the server changes from root or superuser to the user account you specify. If you choose a port number greater than 1024, you don't have to be root or superuser.

Replace an Existing Proxy Server

If you're already running a proxy server, you should install the Netscape Proxy Server to a different port first. After the Netscape Proxy Server is properly set up, shut down the old proxy and then change the new proxy to use the different port.

If you want to install the proxy server using the same port number as the current proxy server, you must shut down the existing server first.

Caution: The proxy server must be installed in an empty directory. The installation process uses */var/ns-proxy* by default.

Root or User Installation

Before you install the server, you should be logged in as the root user unless you meet all of the following conditions:

- You plan to install the server on a port greater than 1024.
- The location where you plan to install the server (the server root directory) is writeable with your current login status.
- The location you plan to use for the cache root is writeable with your current login status.

This means you should be logged in with the user account that the proxy will use. Regardless of your login, make sure the following standard IRIX programs are in your PATH while you run the installation process:

- *cut*
- *expr*
- *find*
- *grep*
- *sed*
- *sort*
- *tee*
- *uniq*
- *whoami*

The installation HTML forms collect data that the installation process later uses to generate the configuration files called *magnus.conf* and *obj.conf* (the proxy uses these files when it runs to control how it works). If you don't understand a setting, you can use the default value and later change it via the Administration forms. If you need more information about the options on the installation forms, see Chapter 3.

When you submit the installation forms, you'll get an error if you don't have sufficient rights to the server root directory (the directory where you want to install the server). If this happens, you can go back to the install forms and choose another directory, or you can go to the filesystem and change your user permissions, then resubmit the forms. You can also quit the installation, login as root, and redo the installation.

What the Installation Does

After you fill out all of the installation forms and click the link called *Install the Proxy Server*, the actual installation takes place. Before that point, no file outside of the installation working directory is modified.

Some temporary files are written to */tmp* and removed after installation. No other files or directories are modified in any way.

The installation process places all the files under the server root directory that you specified in the installation forms. If you enabled caching, the cache framework is created under the cache root directory. The following files and directories are created under the server root directory:

- *ns_proxy* is the proxy program.
- *ns-gc* is the garbage collector program.
- *utils/pstats* is the access log analyzer program.
- *ns-icons/* contains icons for FTP listings and Gopher menus.
- *admin/config/magnus.conf* is the server's main technical configuration file.
- *admin/config/obj.conf* is the server's object configuration file.
- *admin/config/mime.types* is the file the server uses to convert filename extensions such as *.GIF* into a MIME type such as *image/gif*.
- *admin/config/admpw* is the administrative password file.
- *admin/html/* contains the Administration forms used to configure and maintain the proxy after installation.
- *admin/bin/* contains all CGI programs for the Administration forms.
- *admin/userdb/* contains all user databases.

Restart the Server Automatically

Once the Netscape Proxy Server is installed, it and its child processes run constantly, listening for and accepting requests. If your system crashes or is taken offline, the server processes die with it. Make sure your server is configured for automatic restart on reboot with the following procedure

1. Use the *chkconfig* command to see if the proxy server is set to "on":

   ```
   # chkconfig | grep ns_proxy
   ```

2. If you see:

   ```
   ns_proxy off
   ```

   Enter the following command:

   ```
   # chkconfig ns_proxy on
   ```

and repeat step 1 until you see:

```
ns_proxy on
```

When the system is rebooted, the server starts automatically.

Starting and Stopping the Server Manually

If you should ever need to start the server from the command line, you must log in as root or become superuser and type this at the command-line prompt:

```
# /etc/init.d/ns_proxy start
```

If you should ever need to stop the server manually, log in as root or become superuser, check the full process load using **`ps -el`** to see if other users might be using the server and, if not, type this at the command-line prompt:

```
# /etc/init.d/ns_proxy stop
```

Using the Netscape Proxy Manager

After you install the Netscape Proxy Server files, the proxy server should run without problems. However, you might need to change configuration information (by adding security) or perform general maintenance on the server. All of this is done with the Netscape Proxy Server Manager. The proxy manager is a set of forms you use to change options and control your proxy server. You can view the proxy manager immediately after installation (there is a link to the proxy manager).

To view the proxy manager at any time,

1. Use a forms-capable browser (such as the Netscape Navigator) to point to the URL:

   ```
   http://[servername].[yourdomain].[domain]:[port]/admin/
   ```

2. You'll be prompted for a user name and password. This is the administration user name and password you specified during the installation process. Figure 2-1 shows the proxy manager form.

The rest of this manual describes the forms and options used to manage and maintain your proxy server.

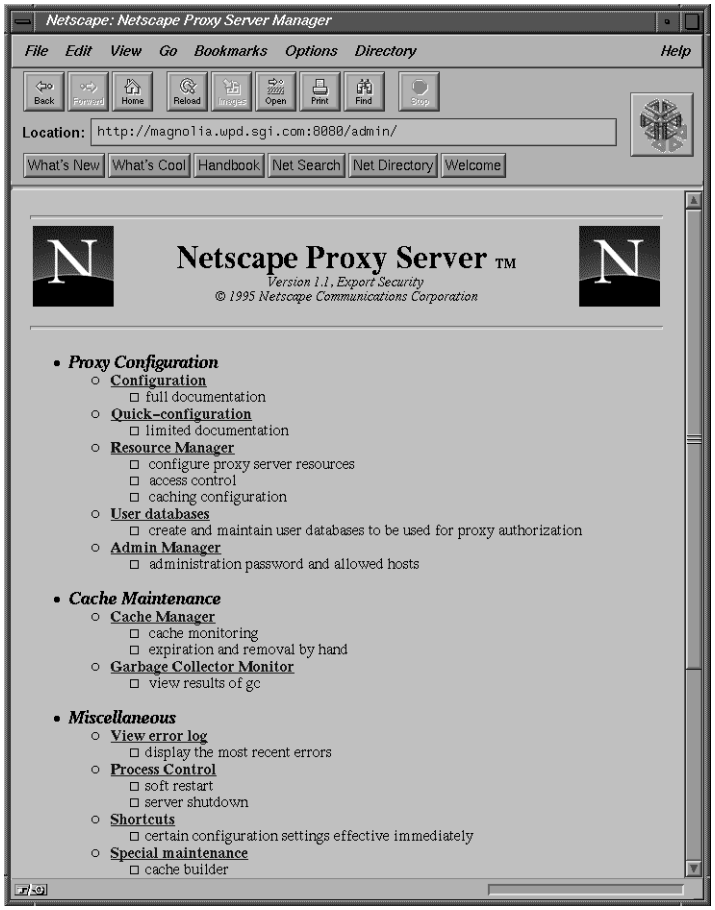

Figure 2-1 Netscape Proxy Manager Form

Troubleshooting Installation

This section describes the most common installation problems and how to solve them.

- You accidentally denied all access to the Administration forms.

Log in as root or with the proxy's user account. In the server root directory, edit the *magnus.conf* file. See "magnus.conf File" on page 70 for more information on this file.

- You don't have access to the proxy.

Log in as root or into the proxy's user account. In the server root directory, edit the *obj.conf* file and remove the following lines:

```
<Client dns="[wildcardpattern]" ip="[wildcardpattern]">
PathCheck fn=deny-service
</Client>
```

The wildcard pattern matches your DNS or IP address. You can also edit the wildcard patterns so that your user account information isn't included. To deny service to everyone *except* a select group, use `*~` before the wildcard pattern (for example, `*~*sgi.com` denies service to everyone not from the sgi.com domain). See "Understanding Wildcard Patterns" on page 34 for more information on wildcard patterns.

- Clients can't locate the proxy server.

First, try using the hostname. If that doesn't work, use a fully qualified name (such as proxy.domain.dom). If that doesn't work, use the dotted IP address.

- Icons don't view properly in client programs.

It's possible the settings in *mime.types* are wrong. View the source of the documents that have the incorrect icons and look at the URL for the image, then make sure the reference in *mime.types* is correct.

- The proxy is slow, and transfers take too long.

If you log files to SYSLOG, you might encounter reduced performance. Switch to using the proxy's error log files instead. The proxy host might need more RAM, or if there are other applications on the proxy host, they might be using CPU cycles, which degrades proxy performance.

You can reduce transfer time by configuring the cache refresh setting. See "Cache Refresh Setting" on page 25 for more information.

Chapter 3

Configuring the Proxy Server

This chapter describes how to configure the Netscape Proxy Server by using the configuration forms from the Proxy Server Manager. The configuration forms control the entire proxy server; the Resource Manager (discussed in the following chapter) controls specific access to the proxy server based on URL patterns. This chapter also discusses how to create and modify user databases.

Using the Proxy Server Manager Configuration Forms

Note: You must restart the server for your changes to take place. After you submit the configuration forms, you get a pointer to a script that restarts your server.

If you installed your proxy server and then realized you need to reconfigure some aspect of the entire server, you can use either the Full Configuration or Quick Configuration forms. The Full Configuration forms contain more documentation, but the options you can configure are identical.

The following sections reference forms and options in both the Full Configuration and the Quick Configuration.

Basic Server Configuration

This form lets you change the basic aspects of your server. Change as many options as you want, then click the *Submit this form* button. The changes you make here take effect only after you restart the server; a link is provided for restart after you submit the form.

The link to the script that restarts the proxy server makes the server internally restart (called a soft restart because it doesn't interrupt proxy service). The proxy rereads its configuration files and restarts its child

processes. The script finds the proxy's parent process ID from the *logs/pid* file and sends the hang-up signal (-HUP) to that process ID. For more information about stopping the proxy server and then starting it (called a hard restart), see "Starting and Stopping the Server Manually" on page 10.

Server Name

The server name is the full hostname of your proxy server host. When clients access your proxy, they use this name. The format for the Server Name is [*hostname*].[*yourdomain*].[*domain*]. For example, if your full domain name is sgi.com, you could install a proxy server with the name proxy.sgi.com.

This name is used in redirections automatically generated by the proxy server. This way it knows what DNS name or alias to use, and clients always see one consistent name.

Server Port Number

Server Port Number specifies the TCP port the server listens to. The number you choose is used by proxy users when they configure their Web browsers (such as the Netscape Navigator) to use the proxy server (they must specify a server name and port number).

Port numbers for all network-accessible services are maintained in the */etc/services* file on IRIX systems. The standard Telnet port number is 23, and the standard HTTP port number is 80. But because the proxy isn't a regular HTTP server, you shouldn't use port 80. Proxies haven't been assigned an official port number yet.

If you aren't sure the port number you plan to use is available, look at the contents of */etc/services* on the server host.

Technically, the proxy port number can be any port from 1 to 65535. If you aren't running as root or superuser when you install or start the proxy, you'll have to use a number above 1024.

A recommended proxy port number is 8080. When configuring client programs to use this proxy server, you have to include both the hostname

and the port number. For example, you would use the following in the proxy preferences dialog box in the Netscape Navigator:

```
proxy.sgi.com 8080
```

If you use proxy's SOCKS daemon feature, the proxy should listen to the standard SOCKS port (1080).

See "Choose a Unique Port Number" on page 6 for more information.

Proxy Server User

Proxy Server User specifies an IRIX user account that the proxy uses (all the proxy processes are assigned to this user account).

You don't need to specify a Proxy Server User if you chose a port number greater than 1024 and aren't running as root (in this case, you don't need to be logged in as root to start the proxy). If you don't specify a user account, the proxy runs with the user account you start it with, so make sure that when you start the proxy server, you use the correct user account.

Even if you need to start the server as root, you don't want it to run as root all the time. You want it to have restricted access to your system resources and run as a nonprivileged user. The user name you enter as the Proxy Server User should already exist as a normal IRIX user account. When the server starts, it runs as this user.

If you want to avoid creating a new user account, you can choose the user "nobody" or an account used by another HTTP server running on the same host. However, on some systems the user "nobody" can own files but not run programs. See "Create an IRIX User Account" on page 6 for more information.

Server Processes

Whenever people use the proxy server, the proxy uses background processes to service their requests. You can specify the number of processes dedicated to the proxy. These processes are spawned when the server starts and remain idle until needed. Base your choice on achieving a balance between system load and server requests:

- On a high-demand system, the server needs many of these processes (for example, 80 processes) to handle many simultaneous requests.
- On a low-demand system (less than a dozen users, only a few simultaneous connections active at any given time), 20-40 processes should be sufficient.

Note: The number of processes the proxy can use is limited by the process table for the computer the proxy runs on.

Each process uses around 200K of RAM when idle and 300-500K when active. If you specify more processes than can fit simultaneously in main memory, the system starts swapping (using virtual memory), which considerably slows down proxy service. All proxy processes must fit in the main memory simultaneously to make the proxy efficient.

To use the table in Figure 3-1 you must know how long requests take and how many requests the proxy receives per second.

- To find the average service time per request, use the `pstats` utility with the access log file (in the extended format). See "Using the pstats Utility" on page 58 for more information on using *pstats*.
- To estimate the average number of new requests per second, view the access log during peak hours. Use *tail* with the **-f** option to continuously view the access log file as the proxy adds entries to it. As entries are added, base your estimate on the number of users and how active they are.

Figure 3-1 presents a table showing the suggested number of processes based on average request service time and number of requests. Use this figure to determine the number of processes for your proxy server.

Average service time per request (seconds)

Average number of new requests per second	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	10	10	10	15	15	20	20	20	25	25	30	30	30	35	35	40
2	10	15	15	20	25	25	30	30	35	40	40	45	45	50	55	55
3	15	20	20	25	30	35	40	40	45	50	55	60	60	65	70	75
4	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90
5	20	25	30	40	45	50	55	60	70	75	80	85	90	100	105	110
6	25	30	35	45	50	60	65	70	80	85	95	100	105	115	120	130
7	25	35	40	50	60	65	75	80	90	100	105	115	120	130	140	145
8	30	40	45	55	65	75	85	90	100	110	120	130	135	145	155	165
9	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180
10	35	45	55	70	80	90	100	110	125	135	145	155	165	180	190	200
12	40	55	65	80	95	105	120	130	145	160	170	185	195	210	225	235
14	45	60	75	90	105	120	135	150	165	180	195	210	225	240	255	270
16	55	70	85	105	120	140	155	170	190	205	225	240	255	275	290	
18	60	80	95	115	135	155	175	190	210	230	250	270	285			
20	65	85	105	130	150	170	190	210	235	255	275	295				
22	70	95	115	140	165	185	210	230	255	280	300					
24	75	100	125	150	175	200	225	250	275	300						
26	85	110	135	165	190	220	245	270	300							
28	90	120	145	175	205	235	265	290								
30	95	125	155	190	220	250	280									
35	110	145	180	220	255	290										
40	125	165	205	250	290											
45	140	185	230	280												
50	155	205	255													

Figure 3-1 Suggested Number of Server Processes

You can change the number of processes at any time using the Administration forms. If the server appears slow or is not responding (especially during peak hours), you should increase the number of processes available to the proxy. You might have to increase the RAM or the size of the operating system's process table before you increase the number of processes.

Proxy Timeout

Because the proxy connects to servers on remote hosts, there is always a risk that the remote host won't respond within a reasonable period of time, either because of network problems or trouble on the remote host. To prevent the proxy from waiting long periods for a response, you can set a timeout. After a specified number of seconds without receiving new data, the proxy terminates the connection.

This time limit is *not* the same as the maximum time allowed for data transfer. The Proxy Timeout is the maximum time between successive network data packets.

When the remote server uses server-push (actively "pushes" a new document out to the client) and the delay between pages is longer than the proxy timeout, you can have problems. Instead, use client-pull (request an updating document), which sends multiple requests to the proxy.

Server Root

The Netscape Proxy Server and its support files were installed in the central directory listed in this section of the form. This directory contains the proxy server program, log files, configuration files, and administrative forms. Certain accessory programs were configured at installation time to directly use this directory. In this version of the Netscape Proxy Server, changing the Server Root is not possible.

Connecting To and Disconnecting From the Network

You can connect the proxy server host to or disconnect from the network. This makes it convenient to install the proxy on a portable host that you can use for demonstrations. (See Chapter 5 for more information on caching.)

When the proxy server is disconnected from the network, documents are returned directly from the cache—the proxy can't do up-to-date checks, so the documents are retrieved very quickly.

Also, if no network is selected, connections never hang, because the proxy server is aware of the missing network and never tries to connect to a remote

server. You can use this setting when the network is down but the server host is up.

Server Logging Configuration

Under normal operation, the Netscape Proxy Server maintains two log files in your server root directory.

- The Access Log keeps track of who accesses the proxy. It can record host names or IP addresses. This file can help you optimize the number of processes the proxy needs.
- The Error Log keeps track of proxy errors. Errors are reported to a file or to the system's syslog facility.

Access Log File

You can keep track of every user who accesses your proxy—including those who try to access it but are denied service.

The access log is saved in a file whose name and directory you specify. Use the Log Format and the Remote Hostname Logging options on the same form to control what host (user) information is saved in the file.

Log Format

You can configure your proxy server to use either the common log file format used by other servers or the extended format specific to the proxy server. The common log file reports

- the date and time of the request (using the proxy's time zone)
- the actual request (method, URL, protocol)
- returned HTTP status code
- size of the document returned

Example 3-1 shows sample extended log entries. Each entry is a single, no-breaking line in the file. The file should have no empty lines. (The lines in Example 3-1 are formatted for the printed page.)

The extended log file format facilitates accounting based on byte counts. In addition to the information in the previous list, the following byte counts are available in the extended log file format in this order:

- Remote server's response status code (HTTP only)
- Remote server's response content length
- Client's request POST data size (content-length)
- The POST data size actually forwarded to the remote server
- Client's request header size
- Proxy's response header length to the client
- Proxy's request header size to the remote server (HTTP only)
- Remote server's response header length to the proxy (HTTP only)
- Transfer time in seconds

Example 3-1 Sample Extended Log File Entries (Each entry is a single line in the file and there are no blank lines)

```
helium.sgi.com--[23/Jan/1995:11:09:15 -0800] "GET http://home.sgi.com/ HTTP/1.0" 200 2573
200 2573 - - 224 188 289 188 11

helium.sgi.com--[23/Jan/1995:11:09:24 -0800] "GET http://home.sgi.com/icons/welcome.gif
HTTP/1.0" 200 567 200 567 - - 210 190 275 190 9

helium.sgi.com--[23/Jan/1995:11:09:24 -0800] "GET http://home.sgi.com/icons/logo.gif
HTTP/1.0" 200 5611 200 5611 - - 209 191 274 191 9

helium.sgi.com--[23/Jan/1995:11:09:39 -0800] "GET http://www.sgi.com/cgi-bin/gadget
HTTP/1.0" 200 1645 200 1645 - - 225 158 290 158 2

helium.sgi.com--[23/Jan/1995:11:09:39 -0800] "GET http://www.sgi.com/icons/gadget-icon.gif
HTTP/1.0" 200 881 200 881--241 190 306 190 1

oxygen.sgi.com - - [24/Jan/1995:02:57:44 -0800] "GET http://home.sgi.com/people/ari/
HTTP/1.0" 200 7718 304 - - - 266 190 331 190 1
```

Remote Hostname Logging

If you set up the proxy server to be used only by local hosts, you can simply log the IP address of the requesting host. This saves precious CPU cycles

when doing a DNS lookup and relieves the impact on the DNS server. These are the two remote hostname logging options:

- Log client DNS hostname. This might have an impact on the performance of the proxy server, and the DNS server might become a bottleneck.
- Log only the IP addresses.

If your access control is DNS-based, you can log the hostnames without affecting performance, because a reverse DNS lookup is required in any case. You can avoid an inpact on performance by having access control based on IP number.

The Error Log File

Error Log specifies the file where the server logs its errors. You can use the SYSLOG facility instead of logging the errors to a file.

Note: The directory and file must be writeable by the proxy's user account.

If you choose error logging to go to a file under the Server Root directory in */logs/errors*, you can easily view the file using the Netscape Proxy Manager. Click the View error log link in the Miscellaneous section of the Proxy Manager page.

Appendix A contains a description of the errors stored in the error log file.

Default Caching Configuration

The Caching configuration form controls the major aspects of caching the proxy server. You can configure the finer aspects (resource by resource) of the cache through the Resource Manager, which lets you specify different configuration parameter values for URLs matching a given wildcard pattern.

See Chapter 5, "Caching," for more information on what caching is, how it works with the proxy server, and how you should configure it for your proxy.

Enabling Caching

Caching reduces network impact and offers faster response times for clients using the proxy. If caching is turned on, the proxy copies documents from the remote server to its local filesystem. When another client accesses the same file, the proxy returns the file from the cache.

Special care is taken by the proxy server to guarantee that documents are up to date. The Netscape Garbage Collector daemon performs cache cleanup on a regular basis, ensuring that the cache doesn't get too full or get cluttered with old documents.

The directory you specify is the parent directory for the cache. All cached files appear in an organized directory structure under this caching directory. If you change the cache directory name or move it to another location, you have to tell the proxy the new location. For more information on moving or splitting the cache directory, see "Moving or Splitting the Cache Directory" on page 52.

Cache Size

The amount of disk space available for the proxy cache has considerable impact on cache performance. If the disk space is too small, the garbage collector must remove valuable cached documents just to make room on the disk.

Large cache sizes are best because the more documents cached, the less network traffic load and faster response times the proxy will provide. Also, cached documents are removed only if they are not needed anymore. Cache size can never be too large; the excess space simply remains unused.

Note: The Proxy Manager allows up to 4 GB for the cache. To increase this, manually edit *magnus.conf*. See "magnus.conf File" on page 70.

Netscape Proxy Caching is designed to work efficiently at any size between 10 MB and 4 GB. The exact cache size you choose depends on the number of people using your proxy server. For a single user cache, 20 to 50 MB is usually enough. For a proxy that caches a lot of documents, you should allocate an entire 1 or 2 GB disk partition for the cache. You can also have the cache split on multiple disk partitions, see "Moving or Splitting the Cache Directory" on page 52 for more information.

Note: You might encounter problems with caching if the filesystem where the cache root resides has less disk space than the cache size you specify.

Cache Lock Timeout

While the Netscape Proxy is writing a cache file, the file is locked so that no other proxy process can access it until it is successfully written to the cache. Sometimes (although rarely) a locked file might be left in the cache. A locked file might be left if the system is suddenly shut down or if the proxy server process terminates for some other reason.

To recover from situations like this, you can specify a timeout period after which a lock can be broken by another proxy process. You should choose a timeout period that encompasses the largest file transfer time you expect for your proxy (for example, 15 minutes).

Locked files left in the directory aren't a serious problem—they make some transactions take a longer time. When a client accesses a document through the proxy and the proxy looks in the cache, it sees the lock file and assumes that another process is active. The proxy then goes to the remote server and passes the document directly to the client. However, if the proxy receives a request for the same document after the lock timeout period has elapsed and it sees the same lock file, the proxy determines that the transaction failed, so it deletes the lock file and restarts the transaction with the remote server.

Specifying a short timeout causes processes to discard large files that are only partially copied when the same file is requested again. Discarding large files doesn't cause any harm other than degradeing performance.

Very long timeouts prevent caching of a resource that wasn't successfully cached during the previous retrieve because of race conditions. A race condition terminates the proxy process prematurely; race conditions include a KILL signal received by the process, a fatal software error, or a system crash.

Race conditions are so rare that you should feel free to set the timeout to an hour without degrading the cache performance.

Transfers interrupted by the client or broken connections to the remote server are not race conditions. The Netscape Proxy easily recovers from these other conditions and releases the lock accordingly.

Cached Protocols

You can control which documents are cached in two ways. The Cached Protocols option lets you specify which documents to cache based on their protocol; the Resource Manager lets you specify more specific rules based on URL wildcard patterns.

Note: The Cached Protocols option supersedes any other caching settings.

Caching must be turned on for a given protocol before any URL of that type can be cached. For example, if you turn on caching in the Resource Manager for a specific URL pattern (such as *ftp://ftpdir/**), that setting has no effect if the entire FTP protocol isn't cached.

By default, the proxy caches the HTTP protocol but doesn't cache FTP or Gopher. See "Cache Refresh Setting" on page 25 for information on caching with FTP and Gopher.

Caching Strategy

Usually, everything that can be cached is cached. This includes all non-protected documents that don't expire immediately and weren't a result of script processing. Scripts often have (intentional) side effects that might affect the document content (in the case of forms data) and that certainly affect the remote server. However, the script results might be cached if the script explicitly allows this with appropriate Expires or Last-Modified headers.

- Cache everything unless explicitly forbidden. If you have a large cache and plan to cache many types of documents, use this option. Every document is cached (except the ones that can't be cached—usually CGI script output), unless you forbid caching for a specific document (via the Resource Manager).
- Cache only documents explicitly marked for caching. This disables caching by default and lets you explicitly enable caching for certain URLs by using the Resource Manager. This is the best way to handle caching if you have a small cache or if you want to cache only a small set of documents with specific types (for example, caching only specific HTTP documents).

Cache Refresh Setting

The proxy server can do an up-to-date check for every HTTP request it gets (FTP and Gopher don't have a way to check if a file is up to date—only a way to retrieve a new copy), or it can do the up-to-date check only after a refresh interval has elapsed.

If you set a time interval, choose one that you consider safe for the information you store. For example, if you store information that rarely changes, use a high number (several days). If your data changes constantly, you'll want the files to be checked each time they are accessed.

During the refresh time, you risk getting an outdated file. But if the interval is short enough (a few hours), you eliminate most of this risk while getting notably faster response times.

Because FTP and Gopher protocols don't include a method for checking that a document is up to date, having a default refresh time is the only way to make FTP and Gopher caching effective. If you don't want to use a Refresh Setting for HTTP documents, you can use the Resource Manager to specify a refresh only for FTP and Gopher documents.

Cache Expiration Setting

The Netscape Proxy Server can use the time a file was last modified to estimate how long a document is likely to remain unchanged. For example, a GIF file that was last changed three months ago would take longer to expire (and have an up-to-date check done) than an HTML file that was updated 2 days ago. This option works only with the HTTP protocol.

The expiration time is used with the Cache Refresh setting; the shorter of the two intervals is used.

If you use this feature, you need to select a factor to use in the estimation. The factor is multiplied by the time between the last modification and the time that the document last had an up-to-date check. Smaller values make the proxy behave more cautiously and check documents more often.

For example, if you set the factor to 0.1, a document that was last modified ten days ago is estimated to remain unchanged for one day (10 x 0.1). The proxy returns the document directly from the cache if less than one day has

passed since the proxy last performed an up-to-date check. However, if the Cache Refresh setting is set to less than one day, the proxy does the up-to-date check if that time has elapsed. The proxy always uses the smaller value (Cache Refresh or Cache Expiration) that makes it update the files most frequently.

Remote Server Unavailable

If a document is in the cache and the proxy determines the file needs to be updated but the remote server is unavailable (for a variety of reasons), you can configure the proxy to return the document from the cache instead of returning an error saying the remote server is down (you'll get the error message if this option is not on).

To use this option, you must specify how old a document can be when sent from the cache. For example, you can specify that a document be no older than 7 days. If someone tries to access a document that is older than 7 days, the proxy won't return it from the cache—it returns an error indicating that an error occurred when contacting the remote server.

An error while contacting the remote server can occur for different reasons. There can be a long timeout period (30 to 60 seconds with DNS lookups) before the proxy detects the error, or sometimes the error can occur immediately.

Even though this feature can increase the usability of the World Wide Web, it also involves a risk of sending stale data from the cache during the time that the remote server is unavailable. It can also return documents that no longer exist on the remote server or from servers that no longer exist.

Configuring the Garbage Collector

The Netscape Garbage Collector Daemon traverses the cache directory and removes old files. It runs through the cache regularly one or more times per day. The garbage collector daemon is started automatically by the same script that starts the proxy server—whenever the proxy is running, so is the garbage collector.

If the garbage collector is accidentally killed, you can start the it manually with the *ns-gc* command. Use the **-d** option to specify the cache root

directory. The garbage collector uses the same configuration file as the proxy server.

Note: If you haven't enabled caching on the proxy, settings made to the garbage collector have no effect and the garbage collector won't be started by the startup script.

See Chapter 5 for more information about the garbage collector and how it works with the cache.

Garbage Collection Times

You can choose up to three garbage collection times (once a day is sufficient if you have a large cache that doesn't fill easily). A hyphen means no garbage collection. If all three collection times are marked with a hyphen (-), the default garbage collection time is used (3:00 a.m.).

Garbage Collector Process Priority

The garbage collector is often run in lower priority; select a priority (nice) value. Zero means normal priority; 39, the lowest priority. If you have a busy CPU and you set a low priority (you use a high nice value), garbage collection might not run as scheduled.

See "What is the Garbage Collector?" on page 56 for more information on the garbage collector and what happens if garbage collection doesn't occur and the cache gets full.

Mapping URLs to Mirror Servers

The Configuration Manager lets you map URLs to another (mirror) server. You specify a URL prefix to map and where to map it. When a client accesses the proxy with a mirrored URL, the proxy gets the requests from the mirrored server and not from the original server. You can also use mapping to send requests to mirror servers only when the requests come from specific clients.

For example, suppose you have a heavily loaded Web server called hi.load.com that you want mirrored to another server (mirror.load.com). You would add this mapping in the Configuration Manager:

```
Map URL Prefix: http://hi.load.com
Mirror site prefix: http://mirror.load.com
```

The source URL prefix must be unescaped, but in the destination (mirror) URL prefix illegal characters must be escaped.

Do not use trailing slashes in the prefixes! You can leave the remaining fields (host restrictions) blank.

Editing Existing Mappings

You can edit your existing mappings by clicking the link at the bottom of the mappings form.

All mappings are grouped in sections, so you can change any mapping and click the *Change this mapping* button to save your edits. You can edit the prefix, the mirror site, and the host and IP addresses that are affected by the mirror mapping.

Check *Remove this mapping* and then click the *Change this mapping* button to remove the mapping from the proxy configuration.

URL Control

The Configuration Manager haS an option that starts the Resource Manager. See Chapter 4 for information on using the Resource Manager.

SOCKS Daemon Configuration

Netscape Proxy Server includes a SOCKS daemon that understands the usual *sockd.conf* file format used by other SOCKS daemons. See “Structure of sockd.conf” on page 107 for information on this file format. (The Netscape Proxy Server supports SOCKS version 4.)

The SOCKS daemon is a generic firewall daemon that controls access through the firewall on a point-to-point basis. By default, the SOCKS daemon features are disabled, but you can enable them through this form.

SOCKD Configuration File

The Netscape SOCKS daemon understands the same configuration file format as the other SOCKS daemons; the default location is */etc/sockd.conf*. You can specify another filename and directory.

Remote Identity Check

The SOCKS daemon can verify the identity of remote users by connecting to the client host's identity daemon (*identd*), if it is running on their host.

There are three levels of identity checking:

- Strict. The remote identity daemon is contacted, and the given user name is verified. If the remote host isn't running *identd* or if verification fails, access is denied to the client.
- Loose. The remote identity daemon is contacted to verify the user name, but if the host isn't running *identd* or if verification fails, the user name check is bypassed and access allowed.
- None. The remote identity daemon isn't contacted, and the user name is never verified.

SOCKD Logging

Netscape SOCKD provides four different ways to log SOCKS requests. Log entries can be sent to the common log (a separate log file) or to the SYSLOG facility. The log format is either Netscape Proxy's extended log file format or the usual format used by regular SOCKS daemons.

- Common Log. Uses the common log file and the same format in it. This uses the special names SOCKS-CONNECT and SOCKS-BIND as method names to denote the two SOCKS commands, CONNECT and BIND.
- Common Log Format, Separate File. Uses the common log file format, but stores the log entries in a separate file.

- Syslog. Makes the usual SOCKD type log entries to the *syslog* facility.
- Syslog-style, to a Separate File. Makes *syslog*-type log entries, but to a separate file.

Separate Log Name

If you use one of the two options that store log entries in separate files (common-separate and syslog-separate), you'll need to specify the directory and filename where you want the entries stored.

Filter Outgoing Headers

You can configure the headers the proxy filters out from the request (usually for security reasons). For example, you might want to prevent the *From:* header from going out because it reveals the user's email address (although Netscape Navigator does not send it unless it is specifically configured to do that).

This feature doesn't affect headers that are specially handled, generated by the proxy itself, or necessary to make the protocol work properly (such as If-modified-since and Forwarded).

The fact that the Forwarded header cannot be stopped from originating from a proxy isn't really a security hole because the remote server can detect the connecting proxy host from the connection.

However, in a proxy chain, a Forwarded header coming from an inner proxy can be suppressed by an outer proxy. This is recommended if it is not desirable to have the inner proxy hostname revealed to the remote server.

The value of this setting must be a wildcard pattern that matches the HTTP header field name up to, but not including, the colon. Two examples are:

```
from
(from|user-agent)
```

Access Control

Access Control lets you restrict access to a resource according to the client's hostname or IP address.

You can either protect your entire server, or select a resource to apply it to. If you get to the Access Control form directly from the Proxy Manager Configuration form, you are modifying the *entire* proxy server. If you get to this form from the Resource Manager, you are modifying access only to the resource you chose in the Resource Manager. This form specifies what you are modifying. If you want to specify a host that is not allowed, start the pattern with *~.

Caution: Access Control affects this configuration interface; if you specify a set of hosts that excludes the hosts with access to the administration forms, you cannot access these forms after you restart the proxy server. There are two ways to fix this. You can manually edit the configuration file *obj.conf* in *admin/config* under your Server Root directory (see "obj.conf File" on page 86 for more information).

You can also access the Proxy Manager forms from the local host using the URL *http://localhost:port/admin/.* Localhost is literally "localhost," so don't replace it with the local hostname, but you do need to use the correct port number. This assumes the */etc/hosts* file includes the entry "127.0.0.1 localhost". In that case, use the numeric loopback address:

`http://127.0.0.1:[`*port*`]/admin/`

User Databases

User data bases are lists of users who can access the proxy server. Each user has a user name and password. User databases control who has access to the Internet and other resources through the proxy server.

The Netscape Proxy Server stores its user files in a high-speed format called a DBM. This format can search an infinitely large database with one filesystem read (normal files search the database linearly).

The Netscape Proxy Server stores its databases in the directory *admin/userdb* in the server root. When specifying a database, use only the name, not the full path.

Creating a User Database

To create a user database for your proxy server, type a name for the database. Don't type a path because all databases are stored in `admin/userdb`. The database name can be up to 256 characters.

Type a password for this database. The password can be up to 8 characters. Retype the password to ensure correctness. When you click the *Submit this form* button, the proxy creates the database, then lets you jump to the page where you can add users to the new database.

Adding, Editing, and Removing Users in a Database

To add, edit, or remove a user to a database:

1. Type the name of the database and the password for the database.
2. Check Add User, Edit User, or Remove User, depending on what you want to do.
3. Type a user name. This is the name the user types when authenticating with the proxy. It can be up to 254 characters.
4. If adding or editing a user name, type a password for the user. It can be up to 8 characters. Type it twice to ensure correctness. The user uses this password when authenticating with the proxy.

   If you want to change the user name, first remove the user and then add the changed user name. If you're removing a user, you don't need to type the password.
5. Click the *Submit this form* button. The user name and password are added to the database. You can return to adding names to the database, or you can return to the User Databases page.

Converting an NCSA or Text File to a User Database

The Netscape Proxy Server stores its databases in the server root, in the directory *admin/userdb*. The NCSA-style file can reside anywhere.

The format of the file to convert should look something like this:

```
user1:password1
user2:password2
user3:password3
user4:password4
```

Passwords are not normally stored in clear text. Usually, they are encrypted so that you cannot read them. If the file is an NCSA file, the passwords will already be encrypted. If the passwords aren't encrypted, you can have the converter encrypt the passwords for you.

You need to choose a name for the converted file. You don't need to type a full path name because the user databases are always kept in *admin/userdb*.

You also need to specify a password for the user database; type it twice to ensure accuracy. You need this password to add, remove, or edit users.

Changing a Database Password

You can change the administrative password for a database. Type the name of the database whose password you want to change. Type its current password, then type the new password twice (to ensure accuracy). Click the *Submit this form* button. The proxy stores the new password.

Removing an Existing Database

You can remove a database from the *admin/userdb* directory. This deletes the file from the directory.

Type the name of the database you want to delete. Type the password for the database. Click the *Submit this form* button. The proxy deletes the file from the filesystem.

Understanding Wildcard Patterns

In many parts of the proxy server configuration (especially the Resource Manager), you specify wildcard patterns to represent one or more items to configure. For example, to restrict access to the proxy, you specify a wildcard pattern that matches all of the clients who should get access.

Wildcard patterns use special characters. If you want to use one of these characters without the special meaning, precede it with a backslash (\) character.

`*` matches zero or more characters.

`?` matches exactly one character, and it can be any character.

`(this|that)` is an **or** expression. It matches either the substring `this` or the substring `that`. The substrings can contain other special characters such as `*` or `$`.

`$` matches the end of the string. This is useful in **or** expressions.

`[abc]` matches one occurrence of the characters a, b, or c. Within these expressions, the only character that needs to be escaped in this is], all others are not special.

`[a-z]` matches one occurrence of a character between a and z.

`[^az]` matches any character except a or z.

`*~` followed by another expression removes any pattern matching the expression from the match list.

Wildcard Usage Examples

`*.sgi.com` matches any string ending with the characters `.sgi.com`.

`(core|flop).sgi.com` matches either `core.sgi.com` or `flop.sgi.com`.

`198.93.9{23}.???` matches a numeric string starting with either `198.93.92` or `198.93.93` and ending with any three characters.

`*.*` matches any string with a period in it.

`*~sgi-*` matches any string starting with `sgi-`.

`*.sgi.com~quark.sgi.com` matches any host from domain sgi.com except for a single host quark.sgi.com.

`*.sgi.com~(quark|neutrino|energy).sgi.com` matches any host from domain sgi.com except for hosts quark.sgi.com, neutrino.sgi.com, and energy.sgi.com.

`*.com~*.sgi.com` matches any host from domain com except hosts from subdomain sgi.com.

Chapter 4

Using the Resource Manager

The Resource Manager lets you configure specific aspects of the proxy server. You choose or create a resource to modify, click the **Submit this form** button to see a list of options you can configure, and then click a link to the option you want to change for that resource.

Hint: To map URLs to mirror sites, use the URL Management page.

A resource can be a single URL, a wildcard pattern that specifies many URLs, or your entire proxy server. A resource is simply a bunch of URLs that are grouped together and have a common configuration. For example, you might want to inhibit caching of certain URLs or specify different cache parameters for certain URLs.

Choosing a Resource to Configure

You can configure the entire proxy server, choose an existing resource from a list, or create one by typing a wildcard pattern that matches only the resources you want to configure. See Table 4-1 for a list of examples.

Table 4-1 Sample Resource Wildcard Patterns

Wildcard Pattern	What it Configures
ftp://*	All FTP requests.
http://*	All HTTP requests.
gopher://*	All Gopher requests.
party	All requests containing the exact string party.
http://home.sgi.com/*	Any request for documents on the home.sgi.com host.
connect://*:443	All SSL (secure) transactions to HTTPS port.

After you choose a resource, click the *Submit this form* button. A list of options you can change for the resource you chose appears. Click the link to change the option.

Configuring a Resource

You can configure many different aspects of a resource. After you select or create a resource to configure, you can change one or more aspects of that resource.

Note: You must restart the server after you configure resources.

This section describes the options you can configure for each resource. Keep in mind that not all options are available for every resource.

Enable/Disable Proxying of a Resource

This option lets you turn off proxying for the resource you chose. This means you can restrict access to one or more URLs by turning off proxying for that resource. This isa global way to deny all access to a resource.

This form tells you the resource you are modifying.

- *Enable proxying of this resource* means the proxy lets clients access this resource (providedthey pass the other security and authorization checks). When you enable proxying for a resource, you can select specific methods to proxy. For example, you might allow all HTTP GETs to be proxied, but you restrict all POSTs. The methods include GET, HEAD, POST, and CONNECT (for SSL proxying).
- *Disable proxying of this resource* means no one can use the proxy to retrieve this resource.

Control Access to This Resource Through This Proxy

This option takes you to the Access Control form where you can specify which users can access the resource you chose (based on hostnames or IP addresses). The level of restricting access is more specific than enabling or

disabling proxying for a resource because you can specify exactly who can get to a resource (whereas enabling and disabling works for anyone who has access to the proxy).

Hint: If you want to specify only a certain host that is not allowed, start the pattern with *~.

You can specify a wildcard pattern of host names or IP addresses that are allowed to access this resource through this proxy server. For example, to allow only users from the sgi.com domain, you would use:

```
Allowed hosts: *.sgi.com
```

You can also remove the restrictions by checking the option called "Remove these host settings", use default access control for this resource. See "Access Control" on page 31 for more information on using Access Control.

Require Proxy Authentication

Note: Proxy user authentication is supported by Netscape Navigator versions 1.1 and later.

Proxy user authentication restricts user access through the proxy. The proxy makes users type in a username and password before giving them access through the proxy (the usernames and passwords are stored in proxy user databases). After users type their name and password, they aren't asked to re-enter the information during that browsing session (the Netscape Navigator remembers the password).

The easiest way to enforce proxy user authentication is to enable it in the entire server, which means that all proxy users must authenticate themselves. If you don't want every user authenticating themselves, you can make users authenticate themselves only when they select a specific resource (URL).

The Proxy user authentication transmits usernames and passwords without encryption. This isn't a problem if you have a trusted network in between your proxy and its clients (for example, your users are on a closed company network).

Before the proxy can use user authentication, you need to create a user database (see "Creating a User Database" on page 32). The database lists all the users (and their passwords) who have access to this authenticated resource.

To enable user authentication, set the following options:

- *User database* is the name of the database the proxy uses when authenticating users before giving them access to the resource (URL).
- *Realm* is a text string that the user's client program (Netscape Navigator) displays so that the user can identify which proxy server they are authenticating to. For example, the realm in the administration database is "Proxy Server Administration."
- *Allowed users* is a wildcard pattern of users in the database that you want to have access to the resource. For example, * would be all of the users in database, k* would be only the users whose username started with the letter K, and (ari|nathan|darin) would allow three users named ari, nathan, or darin.

You can also specify a wildcard pattern of hosts or IP addresses that can access the resources without proper user authentication. This lets local users view the documents in the resource, but restricts access to a limited set of offsite people.

- *Exempted hosts* is a wildcard pattern of hostnames you want to have unlimited access to the resource. For example, you could type *.sgi.com.
- *Exempted IP addresses* is a wildcard pattern of dotted IP addresses that you want to give access to the selected resource. For example, you could type 198.95.251.*.

Inhibit/Enable Caching of This Resource

You can control which resources the proxy caches, provided more general caching options are set. The most general caching option is caching entire protocols (set up through the Proxy Manager configuration forms). The next detail of caching is the caching strategy, and the finest detail is caching based on resources.

See Chapter 5 for more information on caching.

Limit the Length of Cached Queries for This Resource

You can limit the length of queries that are cached, or you can completely inhibit caching of queries. The longer the query, the less likely it is to be repeated, and the less useful it is to cache.

Note: Cached queries only work with HTTP documents.

The same caching restrictions still apply: the access method has to be GET, the document must not be protected, and the response must have at least a Last-modified header or an Expires header. This requires the query engine to indicate that the query result document can be cached. If only the Last-modified header is present, the query engine should support conditional GET method (with an If-modified-since header) in order to make caching effective; otherwise it should return an Expires header.

- To disable caching queries for this resource, choose Queries Not Cached in the list of numbers of characters per query.
- To enable caching queries for this resource, specify the maximum length in characters for queries you want to cache. Longer queries aren't cached.

Set Caching Parameters for This Resource

You can set caching parameters for resources that specify if the resource is cached or not, and if it is cached, how it is cached.

By default, the proxy caches all documents. If you don't want the resource cached, check "Do not cache this resource".

If you choose to cache this resource, you can specify how you want it cached. The proxy can always check that the document is up to date, or it can do the up-to-date check only if the document is expired.

See Chapter 5 for more information on caching and caching options.

Use Another Proxy for Retrieving This Resource

You can have the proxy access another proxy for some resources instead of accessing the remote server. This means you can chain proxies together. Chaining is a good way to organize several proxies behind a firewall. It also lets you build hierarchical caching. Figure 4-1 illustrates how each proxy server has a small cache that a specific group of users has access to. Each proxy also has access to the proxy with the large cache.

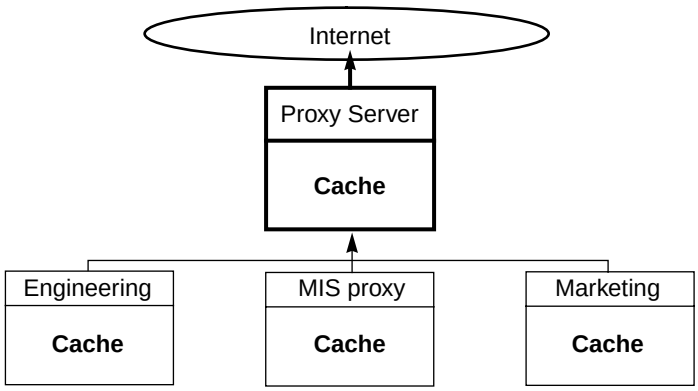

Figure 4-1 Chaining Priorities

Use SOCKS When Retrieving This Resource

You can configure the proxy to connect to a remote server using a SOCKS server for this resource.

To use a SOCKS daemon to retrieve URLs matching the selected resource, type the SOCKS server hostname and port number. See "SOCKS Daemon Configuration" on page 28 for more information about SOCKS daemons.

Customize Error Messages

Netscape Proxy Server lets you override the default error messages for errors that are generated by the configuration system.

Ordinarily, the proxy responds and sends a message to the client. This message is generic and not always helpful to the user.

You can't customize errors that occur when the proxy is contacting a remote server. Those errors are generated on the fly by the proxy, which tries to make them as informative as possible by using all the dynamic data available.

Customizable errors are

- 401 Unauthorized (for Administration forms only). The server requires HTTP user authorization to allow access to the Administration forms, and the client either provided none or its HTTP authorization was insufficient.
- 403 Forbidden. The user tried to access a file or directory for which permission is never allowed.
- 404 Not Found. The client asked for a filesystem path that doesn't exist or the server was configured to tell the client that it doesn't exist. If you use access control, changing the response to this error lets you tell people nicely that they don't have that access to your proxy.
- 407 Proxy Authorization Required. The proxy requires proxy authorization, and the client either didn't provide any, or it was insufficient. Also, the client software might not support proxy authorization. The Netscape Navigator version 1.1 (and newer versions) supports this authorization.
- 500 Server Error. Server errors mean that an error has occurred within the server that prevents it from finishing the request. Server errors mainly happen because of misconfiguration, CGI programs exiting early or otherwise failing, or host resources such as swap space being exhausted.

Remove All Your Settings to This Resource

You can remove resources from the proxy. When you select a resource and use this form, you remove the entire resource and all of its settings (because resources are objects in the *obj.conf* file, this form deletes the object and its settings from the *obj.conf* file).

Caution: Be careful of the resources you remove. You shouldn't remove the HTTP, FTP, or Gopher resources unless you want to entirely disable proxying for them.

View Cached Information for This Resource

You can view all the cached documents that apply to the resource you chose. For example, if you choose the *http://** resource, this form displays all the HTTP documents in the cache.

This form is the same as the one you'd see if you chose the Cache Manager from the Administration forms and used the same wildcard pattern as the resource. See "Using the Cache Manager" on page 53 for more information on this option.

Expire All Cached Items for This Resource

You can expire all documents for a resource. Doing this means the next time the proxy receives a request for the document, it does an up-to-date check and possibly refreshes the document from the remote server. This guarantees that the document will be current. FTP and Gopher documents are automatically refreshed the next time they are requested because there is no way to do up-to-date checks on those types of documents.

Remove All the Items Cached for This Resource

This option lets you remove all documents from the cache that match the resource wildcard pattern. Use this rarely, if at all.

Chapter 5

Caching

This chapter describes how the proxy server caches documents. It also describes how the cache is configured with the Administration forms and maintained with the garbage collector. (See "Default Caching Configuration" on page 21 for information on the specific Administration form cache settings.)

How Caching Works

Caching reduces network impact and offers faster response times for clients using the proxy server. When you install the Netscape Proxy Server you specify a disk cache directory and size. The installation creates a cache directory structure that the proxy uses to store documents from remote servers (see "Cache Directory Structure" on page 50 for a description and Figure 5-5 for an illustration of the cache directory structure).

When a client requests a new document from the proxy, the proxy copies the document from the remote server to its local filesystem in addition to sending the document to the client.

When another request comes for the same file, the proxy returns the file from the cache if the file is up to date. If the proxy determines the file is not up to date, it refreshes the document from the remote server and then sends it to the client. See Figure 5-3 for a diagram of this decision process.

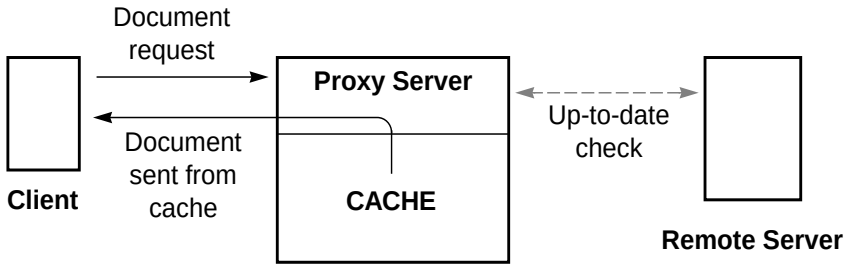

Figure 5-1 The Proxy Up-to-Date Check

Is the Cache Used?

When the proxy starts, it reads in configuration information that tells it if caching is used, and if so, what documents and protocols are cached. If you chose not to set up caching during the installation process, you can use the Proxy Manager's Caching Configuration form to set up a cache system. The proxy needs to know the cache location (cache root directory) and the maximum size of the cache.

If you set up a cache directory, the proxy uses two caching options to determine if a document is cached. The proxy checks if the document's protocol is cached (cached protocols are HTTP, FTP, or Gopher). If the protocol is cached, the proxy uses the caching strategy to see how the protocol is cached.

Even if "everything" is cached, the proxy caches only GET method documents—and only if they have either a Last-Modified or Expires header (or both). Figure 5-2 illustrates caching protocol.

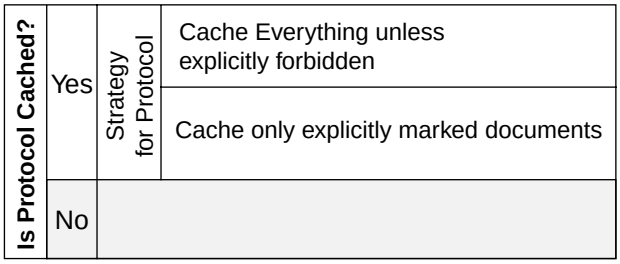

Figure 5-2 Relation of Caching Protocols and Caching Strategy

Is a Document Up to Date?

You can configure the proxy to guarantee that documents are up to date before sending them from the cache. The Cache Refresh Setting (see “Cache Refresh Setting” on page 25) specifies whether to make sure a document is up to date or to wait a specific interval before doing the up-to-date check.

Because FTP and Gopher protocols don’t include a method for checking that a document is up to date, having a default refresh time is the only way to make FTP and Gopher caching effective.

HTTP, on the other hand, provides an easy way to make sure a document is up to date. The proxy makes one call to the remote server that basically says, “send me this document only if it was modified since this date.” The proxy sends the content of the Last-Modified header that was stored in the proxy’s Cache Information File (CIF) as an If-Modified-Since: header.

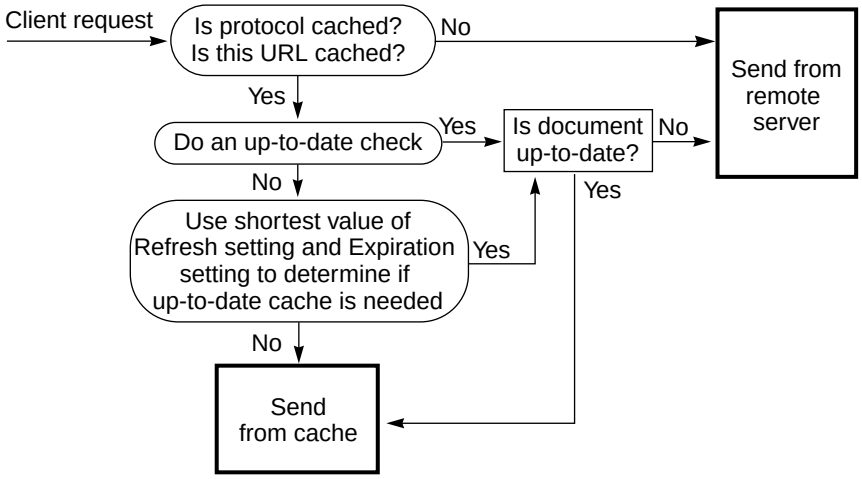

Figure 5-3 Cache Use Algorithm

Caching HTTP vs. FTP and Gopher

Internally, caching HTTP documents differs from caching FTP and Gopher documents. The HTTP protocol provides effective and efficient ways for caching that are missing from FTP and Gopher protocols.

Caching HTTP

HTTP documents have a descriptive header section that the proxy server uses to compare and evaluate the document in the proxy cache and the document on the remote server. When the proxy does an up-to-date check on an HTTP document, it sends one request to the server that tells the server to return the document if it is out of date. Often, the document hasn't changed and isn't transferred. This saves bandwidth and decreases latency.

To reduce transactions with remote servers, you can set a Cache Expiration setting so that the proxy first estimates if the HTTP document needs an up-to-date check before it actually sends the request (the proxy makes the estimate based on the HTTP document's Last-Modified date). Use an expiration setting that fits your data (you can set different expirations for different HTTP documents by creating and modifying a resource). See

"Cache Expiration Setting" on page 25 for more information about the Cache Expiration setting.

With HTTP documents, you can also use a Cache Refresh setting. This option specifies whether the proxy always does an up-to-date check (which would override an Expiration setting) or if the proxy waits a specific period of time before doing a check. Figure 5-4 shows what the proxy does if both an Expiration setting and a Refresh setting are specified. Using the Refresh setting considerably decreases latency and saves bandwidth.

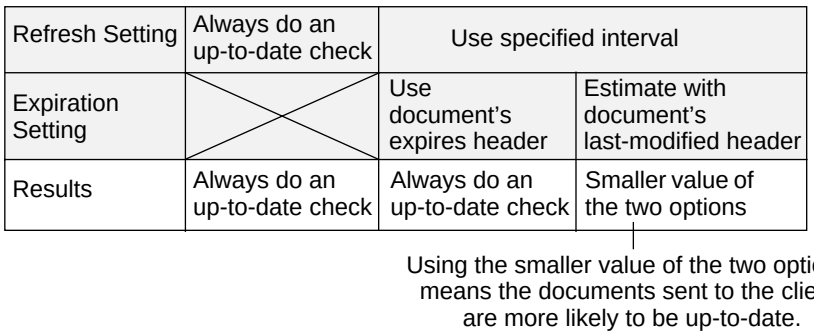

Figure 5-4 Proxy with Expiration and Refresh Setting

Caching FTP and Gopher

The only way to optimize caching for FTP and Gopher protocols is to set a Cache Refresh time; otherwise, you'll have the proxy retrieving these documents even if the documents in the cache are the latest versions.

If you set a Refresh interval, choose one that you consider safe for the documents the proxy gets. For example, if you store information that rarely changes, use a high number (several days). If the data changes constantly, you'll want the files to be checked at least every few hours. Note that during the refresh time, you risk sending an out-of-date file to the client. But if the interval is short enough (a few hours), you eliminate most of this risk while getting notably faster response times.

If your FTP and Gopher documents vary widely (some change often, others rarely), use the Resource Manager to create a resource for the different documents (for example, create a resource like *ftp://*.gif*) and then use a Refresh Interval that is appropriate for that resource.

Optimizing Cache Document Retrieval

If your proxy server is used to access a variety of document types (FTP, HTTP, Gopher), you'll want to use several cache features to optimize the way the cache works. First, you'll want to use the Caching Strategy that caches everything unless explicitly forbidden—if you have a very limited cache size, you wouldn't use this option.

The following list describes a recommended cache configuration:

- Use Cache Refresh Settings to optimize caching of FTP and Gopher documents. Create resources for documents and then use different Refresh Intervals as appropriate. For example, you could create resources such as *ftp://*.gz* and **.(Z|gz)*.
- Set a reasonable Refresh Interval (for example 4-8 hours) for HTTP documents. This reduces the number of times the proxy connects with remote servers. Even though the proxy doesn't do up-to-date checking during the refresh interval, users can force a refresh by clicking the *Reload* button in the Netscape Navigator (this makes the proxy retrieve the document from the remote server).
- Use a Cache Expiration Setting to estimate when a document is likely to be out of date (valid only for HTTP documents). Not many HTTP documents use explicit Expires headers, so it's better to estimate based on the Last-modified header (for example, use an estimation factor of 0.1).

Cache Directory Structure

When you install the proxy server, you specify a directory (usually a subdirectory of the server root) for the cache—this is where the proxy temporarily stores documents.

The installation creates the cache root directory and creates a set of subdirectories where it places documents as it retrieves them from the remote servers. Figure 5-5 shows the structure of the cache root.

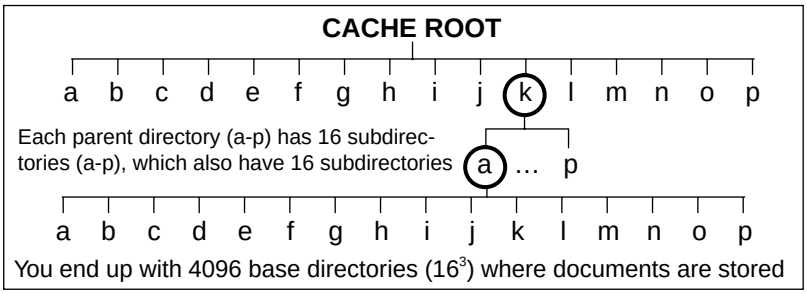

Figure 5-5 Three Subdirectories of the Cache Root Directory

The proxy stores documents in the last level of directories (there are 4096 directories at this level).

Dispersion of Files in the Cache

The proxy uses a certain algorithm to determine the directory where a document should be stored. This algorithm ensures equal dispersion of documents in the base directories, so the directories contain a small and nearly equal number of documents. This is important for several reasons:

- Directories with large numbers of documents tend to cause performance problems.
- Garbage collection is much more stable because there is a relatively consistent number of files per directory, making it easier to estimate the number of files to remove.

Filename and Directory

The proxy uses the RSA MD5 algorithm (Message Digest) to reduce a URL to 8 characters, which it then uses for the filename of the document it stores in the cache.

The MD5 algorithm reduces the URL to 128 bits (16 bytes) of binary data. The proxy uses only 48 bits (6 bytes) to calculate an 8-character filename and determine the storage directory; this is enough to cache millions of URLs.

Limiting the Number of Cache Directories

You can limit the number of top-level cache directories (*CacheRoot/a…p*). This, in effect, reduces the total number of directories the cache has for storing documents. The number of top-level cache directories must be a power of two (1, 2, 4, 8, or 16):

- 1 (a) yields 256 bottom-level directories (optimized for approximately 150 MB cache size).
- 2 (a-b) yields 512 bottom-level directories (optimized for approximately 300 MB cache size).
- 4 (a-d) yields 1024 bottom-level directories (optimized for approximately 500 MB cache size).
- 8 (a-h) yields 2048 bottom-level directories (optimized for approximately 1-2 GB cache size).
- 16 (a-p) yields 4096 bottom-level directories (optimized for 2-5 GB cache size).

To limit the number of directories, you must manually edit the init-cache function in the *magnus.conf* configuration file (see "magnus.conf File" on page 70 for more information).

Caution: The cache structure is built during installation and can't be altered later without rebuilding the entire cache from scratch. If you aren't sure what cache size to use, use the largest cache capacity or use the 2 GB default value in the installation forms (this default can hold more than 2 GB of data and can be used with 3-5 GB caches).

Moving or Splitting the Cache Directory

You can move the cache root directory to another filesystem or directory. After you move or rename the directory, you must use the Proxy Manager to tell the proxy server where the new cache location is.

The easiest way to move the directory structure is to pack the directory with *tar* and untar the file in the new directory:

```
tar cf - *|(cd [NewDirectory]; tar xvf -)
```

After you move the physical directory, use the Proxy Manager to point the proxy to the new cache root and then do a soft restart of the proxy server.

You can also split the cache directory by using symbolic links from the original cache root to a new directory. The cache root contains subdirectories *a-p*, which can be individually relocated as long as a symbolic link is created from the old directory to the new one. For example, you can have the proxy look for the cache structure in a directory called *proxycache*. This directory could contain the cache subdirectories *a* through *h*. You could then have symbolic links for directories *i* through *p* that point to a directory called *othercache*.

In the *proxycache* directory (the actual cache root), you'd type:

```
ln -s /othercache/i i
```

to create a symbolic link from *proxycache/i* to *othercache/i*. Repeat this for directories *j* through *p*.

You can only use symbolic links with the first subdirectory structure (*cacheroot/a-p*) and you must copy all subdirectories for each directory.

Using the Cache Manager

The Netscape Proxy Cache Manager lets you control caching for documents, lets you view all cached documents, and lets you see an estimated size of the current cache structure.

The Cache Manager lets you view all cached URLs and lists information about the URLs. You can explicitly expire documents in the cache (so that the next time they are accessed, the proxy does an up-to-date check and possibly refreshes the document in the cache), and you can remove one or more documents from the cache.

Accessing the Cache Manager

You can access the Cache Manager from the Administration forms or from the Resource Manager. From the Administration forms, the Cache Manager

lets you view all cached documents grouped by type and site name (for example, *http://home.sgi.com*). You can then type wildcard patterns to limit the list of sites or URLs you view.

Note: This is the same as if you used the Administration forms, accessed the Cache Manager, and typed in the resource as a wildcard pattern.

From the Resource Manager, you specify a resource first, then click the link called View the list of cached information pertinent to this resource. The Cache Manager appears with the resource you chose as a wildcard pattern, and the cached documents are listed in alphabetical order with radio buttons for expiring and removing the documents.

From the Administration Forms

When you access the Cache Manager from the Administration forms, you either view all cached sites or you use wildcard patterns to restrict what you view. When you view sites or documents, you can then expire or remove them from the cache.

When you view all cached sites, you click a button to view them either listed as bulleted items or listed with radio buttons for expiring and removing the site. The bulleted list is generated more quickly (this is good when you simply want to view information), but the radio buttons list lets you explicitly expire or remove individual sites. With the bulleted list you must first select the site, then choose the documents at that site that you want to expire or remove, so it can actually take you longer to do the same task.

With either list, you can click a link for a single site and then select specific cached URLs from that site to expire or remove.

From the Resource Manager

When you access the Cache Manager from the Resource Manager, the Cache Manager uses the resource wildcard pattern to show only the cached documents that match the resource.

When you click the link called “View the list of cached information pertinent to this resource” the Cache Manager appears with the documents listed with

two radio buttons, so you can select the resources you want to remove or expire.

Expiring and Removing Documents in the Cache

You can expire and remove documents in several ways:

- You can select the radio buttons for each URL and then click a button to expire or remove them.
- You can select a specific URL, and then expire or remove it.
- You can simultaneously expire or remove many documents by using wildcard patterns.

Use the Cache Manager to select documents to expire or remove. See the previous section for instructions on selecting documents. When you have a list of documents, you can click the link for a specific document to get a form that lets you remove or expire that particular document.

However, if you know exactly which resources you want to expire or remove, you can use the shortcuts on the Cache Manager form—they are quick to type in, but they are slower to perform for the server because they affect the entire cache. If you choose this way, be careful when selecting the wildcard pattern.

Rebuilding the Cache Directory Structure

The Netscape Proxy Server requires a specific directory hierarchy under the cache root directory. If this hierarchy is missing or damaged, caching won't work.

Note: You can also use this utility if the cache hierarchy has been damaged (for example, parts of directories are missing).

To rebuild the cache hierarchy (if it's damaged or incomplete), click the *Special Maintenance* link from the Proxy Manager page. In the Cache Builder and Repairer section, click the button to rebuild the cache hierarchy.

While the cache is building, you can view the cache builder messages to see when cache build is complete or if errors have occurred.

Repairing the Cache URL Database

The proxy has a utility that goes through the entire cache and repairs the Cache Manager's URL database. Use this utility if your Cache Manager's URL database appears damaged when viewed through the Cache Manager (for example, if the URL database doesn't seem to contain all the URLs, if the Cache Manager claims that the cache is empty or corrupt, or if it shows garbage even after reloading the page). While running the utility, you can look at the progress report to see its current status.

To repair the cache URL database, click the Special Maintenance link from the Proxy Manager page. Click the button in the URL Database Repair section.

Running the URL repair utility can take from a few minutes to a couple of hours to complete, depending on the size of the cache and the speed and load of your host and its disks. The update takes effect on the top-level URL (site) listing only after the entire operation has completed.

This utility is rarely needed. The only way that the URL database can get out of sync is if something prevents the proxy from updating its URL database after it has completed the cache write. This could happen if the disk suddenly becomes full, if the file permissions are wrong, or if the system suddenly goes down. The URL database is located in the hosts subdirectory under the cache root directory.

This utility can re-create the entire URL database from scratch if it is accidentally deleted.

What is the Garbage Collector?

The Netscape Garbage Collector Daemon performs cache cleanup on a regular basis, ensuring that the cache directories don’t get too full or get cluttered with old documents.

The Garbage Collector Daemon runs through the cache files regularly, usually one or more times a day. It starts automatically by the same script that starts the proxy server. You can choose up to three garbage collection times a day. Once a day is sufficient if you have a large cache that doesn't fill up quickly. If you have a small cache or a busy server that caches a lot of documents, you should set the garbage collector to run three times a day.

Every 20 minutes the garbage collector checks if garbage collection is needed (the cache is about to overflow) and starts garbage collection only if it is needed (but it always runs at the times you specify). The cache directory can become full if the proxy is busy and caches a lot of files or if the cache is too small for the number of documents the proxy is configured to cache.

How Garbage Collection Works

When the garbage collector runs, it reads in information for files in the cache directory and uses it to determine which files to remove. Specifically, it looks at

- how long it took to transfer the document from the remote server
- how much time has elapsed since the file was last refreshed or had an up-to-date check

This generally means that larger files stay in the cache longer, providing quicker response times when a client requests the document. It also means that older documents are removed before newer ones.

Garbage Collector Process Priority

The garbage collector can be run as a low priority process. You can specify a priority (nice) value—the larger the nice value, the lower the priority. Zero means normal priority, 39 lowest priority.

Caution: If you have a busy CPU and you set a low priority (you use a high nice value), garbage collection might not run as scheduled. In this case, use the normal priority (zero).

If garbage collection never runs (that is, the CPU never gives the process any cycles because the process has too low of a priority), the garbage collector daemon forces a garbage collection at normal priority.

Emergency Garbage Collection

The garbage collector will run immediately if the filesystem is full. This happens if the cache root is on a shared filesystem and other applications fill up the disk space.

Using the pstats Utility

The Netscape Proxy Server has a log analyzer utility that collects data from the proxy access log file. This utility produces output that can help you determine the proxy's performance and fine-tune the number of processes the proxy needs, and that can suggest an optimal cache size.

The *pstats* utility is in *[ServerRoot]/utils*. You specify the access log name as a parameter (the log must be in the extended log file format):

```
utils/pstats logs/access
```

Interpreting the pstats Output

The output of *pstats* displays

- Two different transfer time diagrams. The first lists the percentage of requests finished in each service time category (in seconds). The second lists the percentage of the requests completed by a certain maximum service time.
- Status codes returned by the remote server and to the client. Because a client can interrupt the connection before the transfer is complete, the two numbers of status codes don't necessarily exactly match the number of bytes transferred in each direction.
- Requests and connections. This is the number of times the proxy was able to send a document from the cache instead of from the remote server. The higher this number is the more efficient the proxy is because

it doesn't have to access the remote server and use valuable network time.

- Cache performance report. This lists detailed cache information such as the number of documents retrieved from the cache and from remote servers.
- Transfer time report. This is the number of seconds the proxy takes to transfer documents.

Number of Server Processes

The transfer time report is important when determining how many processes the proxy should use (specified by Server Processes in the Configuration form or specified with the MaxProcs directive in *magnus.conf*).

Average transaction time is the actual time that one server process is busy on average with a single request. This number should be used when determining how many server processes are needed.

The other figures do not take errors into account, and give only the perceived response time to the user, not the actual time that server resources were bound to servicing the entire load of incoming requests.

Chapter 6

Security

This chapter describes how to make and keep your proxy server secure. It describes SSL and HTTPS. It also discusses how to use the Proxy Manager to implement security features in your server.

Why Do I Need Security?

Without thorough security, information transmitted over the Internet is susceptible to fraud and other misuse. Information traveling between your computer and a server uses a routing process that can extend over many computer systems. Any one of these computer systems represents an intermediary with the potential to access the flow of information between your computer and a trusted server. You need security to make sure that the intermediaries don't deceive you, eavesdrop on you, copy from you, or damage your communications.

The SSL protocol delivers server authentication, data encryption, and message integrity. SSL is layered beneath application protocols such as HTTP, SMTP, Telnet, FTP, Gopher, and NNTP, and layered above the connection protocol TCP/IP. This strategy lets SSL operate independently of the Internet application protocols.

There are two types of security: application level and network level. Application security ensures that data passed over the network can't be stolen by a third party. Network security ensures that third parties can't intrude on your network and gain access to the network files and resources. This chapter deals with application-level security.

What is SSL?

The Secure Sockets Layer (SSL) protocol for Internet security (developed by Netscape Communications to ensure private and authenticated communications) is an open platform put into the public domain for the Internet community.

SSL provides data encryption, server authentication, message integrity, and optional client authentication for a TCP/IP connection. SSL uses technology licensed from RSA Data Security Inc.

Note: SSL has been submitted for consideration as a standard security approach for World Wide Web browsers and servers on the Internet.

How Does SSL Work?

SSL uses a security *handshake* that is used to initiate the TCP/IP connection. This handshake results in the client and server agreeing on the level of security they will use. After the handshake, SSL encrypts and decrypts the byte stream of the application protocol being used (for example, HTTP, NNTP, or Telnet). This means that all information in both the HTTP request and response is fully encrypted, including

- The URL the client requests
- All submitted form contents (for example, credit card numbers)
- Any HTTP access authorization information (for example, user names and passwords)
- All data sent from the server to the client.

For more detailed information, see the SSL Protocol specification at *http://home.mcom.com/ssl.html.*

What About SSL and the Proxy Server?

When a client requests an SSL connection to a secure server through a proxy server, the proxy opens a connection to the secure server and then simply copies data in both directions without intervening in the secure transaction.

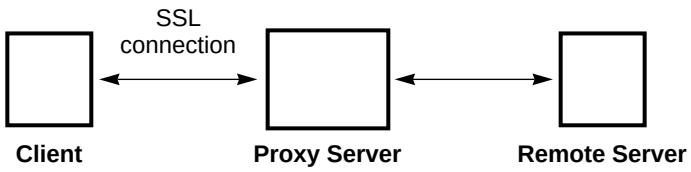

Figure 6-1 SSL Connection and Data Transfer

To use SSL proxying with HTTPS URLs, the client must support both SSL and HTTPS (such as the Netscape Navigator). HTTPS is implemented using SSL with normal HTTP. Clients without HTTPS support can still access HTTPS documents using Netscape Proxy's HTTPS proxying capability (see “What is HTTPS?” on page 65 for more information).

SSL proxying is a lower-level activity that doesn’t affect the application-level (HTTPS). SSL proxying is just as secure as SSL without proxying; the existence of the proxy in between does not in any way compromise security or reduce the functionality of SSL.

With SSL, the data stream is encrypted, so the proxy has no access to the actual transaction. Consequently, the access log cannot list the status code or the header length received from the remote server. This also prevents the proxy, or any other third party, from eavesdropping on the transactions.

Because the proxy never sees the data, it can’t verify that the protocol spoken between the client and the remote server is SSL. *This means the proxy also can’t prevent other protocols from being passed through.* You should restrict SSL connections to only well-known HTTPS ports, namely port number 443 as assigned by the Internet Assigned Numbers Authority (IANA). If there are sites that run the secure server on some other port, you can make explicit exceptions (via the Resource Manager) to allow connections to other ports on certain hosts. You would do this using the connect://* resource. At a later date, when more protocols are enhanced with SSL, you can use the standard port numbers for those protocols.

The Netscape Proxy SSL proxying capability is actually a general, SOCKS-like capability that is protocol-independent, so you can use this feature for other services, too. The Netscape Proxy handles SSL proxying for any application that has SSL support, not only the HTTPS protocol.

Configuring SSL proxying

To configure SSL Proxying

1. Jump to the Resource Manager from the Proxy Manager page.
2. Select the connect://*:443 resource from the list of existing resources (the default HTTPS port number is 443). If you don't already have this resource, type **`connect://*:443`** in the wildcard pattern text box. ("connect://…" is an internal proxy notation and doesn't exist outside of the proxy.)

   If you want to allow connections to other ports, you can use similar URL patterns.
3. Click the *Submit this form* button, and then restart the proxy.

Caution: If the proxy is misconfigured, it is possible to abuse the SSL proxy to achieve "telnet-hopping." Someone can use the proxy to make it appear that a telnet connection is coming from the proxy host, rather than the actual connecting host. This is why you have to pay extra attention to allow no more ports than absolutely necessary and to use access control on your proxy (restricting the client hosts).

SSL Proxy Protocol: Technical Details

Internally, SSL proxying uses the CONNECT method with the destination hostname and port number as a parameter followed by an empty line:

```
CONNECT energy.sgi.com:443 HTTP/1.0
```

A successful response from the proxy server is

```
HTTP/1.0 200 Connection established
Proxy-agent: Netscape-Proxy/1.1
```

followed by an empty line. Then the connection is set up between the client and the remote server, and they can transfer data in both directions until either closes the connection.

Internally, to benefit from the normal Netscape configuration mechanism based on URL patterns, the hostname and port number (energy.sgi.com:443) are automatically mapped into a URL like this:

```
connect://energy.sgi.com:443
```

This is only an internal notation used by the Netscape Proxy to make configuration easier and uniform with other URL patterns. Outside of the proxy server, connect URLs do not exist and if the Netscape Proxy receives such a URL from the network, it marks it as invalid and refuses to service the request.

What is HTTPS?

HTTPS is normal HTTP wrapped in a secure SSL layer. If you use the Netscape Navigator (or other SSL-enabled browser) when accessing the proxy server, HTTPS URLs are proxied by using the SSL proxy feature, *not* by using the HTTPS proxy feature.

Note: Netscape Navigator doesn't use this proxy HTTPS option because it fully supports HTTPS and SSL proxying.

Clients without native HTTPS support or without SSL proxy support can use Netscape Proxy's direct HTTPS proxying feature. HTTPS proxying is similar to proxying other protocols, such as HTTP or FTP. In the HTTPS case, the protocol spoken between the client and the proxy is always HTTP, but only the proxy establishes the secure connection to the remote server. That is, transactions between the proxy and the server are encrypted, while the transactions between the client and the proxy are sent in the clear.

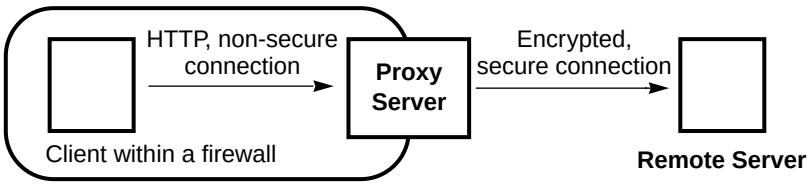

Figure 6-2 Proxy Establishes Connection

This means that in order to achieve maximum security, the network between the client and the proxy must be secure (or trusted) because documents are passed from the proxy to the client unencrypted. For example, a corporation's network behind a firewall could be considered secure, so secure documents cannot be disclosed to an outsider—the outsider has no access to the internal network, and outside of the network, the document is transferred encrypted.

Configuring HTTPS Proxying

To configure HTTPS proxying

1. Jump to the Resource Manager from the Proxy Manager page.
2. Select the https://* resource from the list of existing resources. If you don't already have this resource, type **`https://*`** in the wildcard pattern text box.
3. Click the *Submit this form* button, and then restart the proxy.

Restricting Allowed Hosts

To restrict the hosts that can use the Proxy Manager

1. Click the Admin Manager link on the Proxy Manager page. The Administrative password form appears.
2. Scroll down the page to the text boxes. Type wildcard patterns for the hostnames and IP addresses that are allowed into the proxy server. See Table 6-1 for examples of restrictions.
3. Change the administrative user name or password if you need to (see "Changing the Administrative User Name and Password" on page 67 for information).

4. Click the *Submit this form* button. The Proxy Manager updates the information. Be sure to restart the server.

Table 6-1 Example Host Restrictions

Restriction	Allows access to ...
*.sgi.com	Anyone whose host name ends with .sgi.com.
host1.sgi.com, host2.sgi.com	Allows only two hosts—the ones with host names of host1.sgi.com or host2.sgi.com.
host*.sgi.com	Anyone whose host name starts with the text "host" and ends with the domain .sgi.com.
(host1|host2).sgi.com	Either host1.sgi.com or host2.sgi.com, but doesn't allow any other hosts.
198.93.92.*	Anyone whose IP address begins with the numbers 198.93.92. For example, 198.93.92.23 is allowed, but 198.93.921.78 is not (note the placement of the last dot).

Changing the Administrative User Name and Password

To change the administrative user name and password

1. Click the Admin Manager link on the Proxy Manager page. The Administrative password form appears.
2. Scroll down the page to the user name and password text boxes.
3. Type a new user name. You can use "admin" for the user name for convenience. You can skip this step if you want to change only the password.
4. Type the current password for the administration account.
5. Type the new password. Make sure you retype the password to ensure accuracy.
6. Click the *Submit this form* button. The Proxy Manager updates the proxy server with the new user name and password. Be sure to restart the proxy server for this change to take affect.

Chapter 7

Proxy Configuration Files

This chapter describes the configuration files the proxy uses. You can use these files to manually configure the proxy server.

You might need to manually configure the proxy server for various reasons. If you accidentally lock your hosts out of the administrative forms or if you forget your administrative password, you'll have to manually change information in the proxy's configuration files.

Note: Before you can edit any of the configuration files, you must have permission to read and write the files. This probably means you need to log in as root or with the proxy's user account.

The proxy configuration files are kept in the directory *admin/config* in your server root directory. These files are described in more detail in the rest of this chapter.

- *magnus.conf* is the server's main technical configuration file. It controls aspects of the server operation not related to documents, such as hostname and port.
- *obj.conf* is the server's object configuration file. It controls access to the proxy server and manages the cache.
- *mime.types* is the file the server uses to convert filename extensions such as *.GIF* into a MIME type like *image/gif*.
- *admpw* is the administrative password file. Its format is user:password. The password is DES-encrypted just like */etc/passwd*.

If you use SOCKS, the proxy needs the SOCKS server configuration file stored in */etc/sockd.conf.* This file is described in "sockd.conf File" on page 106.

magnus.conf File

The technical configuration file, called *magnus.conf*, controls all server operations not related to handling of URLs (documents)—the *obj.conf* file handles the URLs. All of the items in the *magnus.conf* configuration file are global and apply to the entire proxy server, as opposed to affecting only one URL or set of URLs.

Every command line in the file has the format:

```
Directive Value
```

- *Directive* identifies an aspect of server operation. This string is case-insensitive.
- *Value* is a specific value you are giving the directive. Its format depends on the directive. This string is usually case-sensitive.

Comment lines begin with a # character with no leading white space.

Directive lines can contain white space at the beginning of the line and between the directive and value, but trailing white space after the value might confuse the server. Long lines (which should only occur with the Init directive) can be continued with a backslash (\) character before the linefeed.

Caution: If you are using the Administration forms, you shouldn't use continuation lines in the *magnus.conf* file. Instead, put each Init configuration entirely on a single line. If you are absolutely sure you will never use the Administration forms to edit the *magnus.conf* file, you can use the backslash character.

Example 7-1 Example magnus.conf File

```
# Sample magnus.conf file for Netscape Proxy 1.1
# Note that ServerRoot isn't a directive to the proxy itself
# because all the paths are already specified absolute by the
# administration interface. This is only for the admin scripts.
#ServerRoot /var/ns_proxy
#
# Accept connections to port 8080
Port 8080
#
# Object configuration filename (this is the default for admin
```

```
# scripts -- don't change this line)
LoadObjects obj.conf
#
# Default object name (this is the default for admin scripts.
# Don't change this line.)
RootObject default
#
# Error log name (default place for admin scripts.
# Don't change this line.)
ErrorLog /var/ns_proxy
#
# Parent process id file (default for admin scripts.
# Don't change this line.)
PidLog /u/luotonen/test/auth/logs/pid
#
# The IRIX user id that the proxy runs as
User http
#
# Server name to be used with redirects
ServerName proxy.sgi.com
#
# Maximum number of child processes active simultaneously
MaxProcs 60
#
# Load MIME type information
Init fn=load-types mime-types=mime.types
#
# Initialize the common log filesystem, and
# Opens the global access log file
Init fn=init-clf global=/var/ns_proxy
#
# Activate the proxy network library, use extended log file
# format, and set the proxy timeout to 2 minutes (120 seconds).
Init fn=init-proxy log-format="extended" timeout="120"
#
# Activate caching, set cache root to /vol1/ns-cache with
# 2GB cache size; cache all HTTP, FTP and Gopher documents;
# check that file is up to date if it’s older than 1 hour;
# and do garbage collection at 3AM with reduced priority.
# Note: Don't use continuation lines (backslashes at the end of
# lines) if you are using the administration interface—the
# scripts can't understand them. Instead, write all parameters on
# a single, long line. Using the \ character is OK if you never
# use the Administration forms.
Init fn=init-cache cache-root="/vol1/ns-cache" cache-size=2000 \
```

```
cache-protocols="http,ftp,gopher" max-uncheck=3600 \
lm-factor=0.06 gc-times="3:00" gc-nice=4
```

Directives in magnus.conf

This section defines the directives and describes their characteristics, including the directive name and description, format for the value string, default value if the directive is omitted, and how many instances of the directive should be in the file. The directives are

- **ServerName** defines the proxy hostname.
- **Port** defines the TCP port the server listens to.
- **User** specifies the proxy's IRIX user account.
- **MaxProcs** sets the maximum number of active processes.
- **MinProcs** sets the minimum number of active processes. This directive isn't available through the proxy Administration forms.
- **ProcessLife** specifies the number of requests each child process serves during its lifetime.
- **ErrorLog** specifies the directory where the server logs its errors.
- **PidLog** specifies a file to record the proxy's main process ID (pid).
- **LoadObjects** specifies a startup object configuration file.
- **RootObject** defines the default server object.
- **Chroot** lets the server be placed into a jail—for security reasons.
- **Init** (a special directive) initializes server subsystems.

ServerName

ServerName tells the server what to put in the hostname section of any URLs it sends back to the client. This affects redirections done by the administration interface. This name is what all clients use to access the server; they need to combine this name with the port number.

You can't have more than one ServerName directive in *magnus.conf*.

SYNTAX

ServerName *host*

host is a fully-qualified domain name such as myhost.sgi.com.

DEFAULT

If ServerName isn't in *magnus.conf*, the proxy server attempts to derive a hostname through system calls. If they don't return a qualified domain name (for example, it gets myhost instead of myhost.sgi.com), the proxy server won't start, and you'll get a message telling you to manually set this value.

EXAMPLES

```
ServerName proxy.sgi.com
ServerName www.proxy.anycompany.com
ServerName www.agency.gov
```

Port

Port controls which TCP port the server listens to. If you choose a port number less than 1024, the server must be started as root or superuser. The port you choose also affects how the proxy users configure their navigators (they must specify the port number when accessing the proxy server). There should be only one Port directive in *magnus.conf*.

There are no official port numbers for proxy servers, but two common numbers are 8080 and 8000. If you use the Netscape Proxy's SOCKS daemon feature, the proxy should use the standard SOCKS port (1080).

SYNTAX

Port *number*

number is a whole number between 0 and 65535.

DEFAULT

If no port is specified, the server assumes 8080.

EXAMPLES

```
Port 8000
Port 8080
Port 1080
```

User

User specifies the IRIX user account for the proxy server. If the proxy is started by the superuser or root user, the server binds to the Port you specify, and then switches its user ID to the user account specified with the User directive. This directive is ignored if the server isn't started as the superuser.

The user account you specify should have write permission to the proxy server's root and cache directories. The user account doesn't need any special privileges. Although you can use the nobody user, it isn't recommended.

SYNTAX

User *login*

login is the eight-character (or less) login name for the IRIX user account.

DEFAULT

If there is no User directive, the server runs with the user account it was started with. If the server was started as root or superuser, you'll see a warning message after startup.

EXAMPLES

```
User proxy
User sgi
User nobody
User Ari
```

MaxProcs

MaxProcs sets the maximum number of processes the server can have active. The proxy server keeps the number of active processes between the number specified in MaxProcs and the number in MinProcs. If there is no MinProcs

directive, then the MaxProcs number specifies the constant number of processes the proxy keeps active.

Choose a number that is appropriate for the type of access you expect for the proxy server. If this number is too small, clients will experience delays. If the number is too large, you might waste resources that other programs could use.

SYNTAX

MaxProcs *number*

number is a whole number between 1 and the size of your system's process table.

DEFAULT

MaxProcs 50

EXAMPLES

```
MaxProcs 20
MaxProcs 40
MaxProcs 80
```

MinProcs

MinProcs sets the minimum number of processes the server can have active. The proxy server regulates the number of processes between MinProcs and MaxProcs. If MinProcs isn't specified, a constant number of processes (specified with MaxProcs) will run.

SYNTAX

MinProcs *number*

number is a whole number between 1 and the number specified in MaxProcs.

DEFAULT

There is no default; if this directive is missing, the server uses the MaxProcs number to specify a constant number of processes.

EXAMPLES

```
MinProcs 10
MinProcs 20
```

ProcessLife

ProcessLife specifies the number of requests each of the proxy's child processes serves before the processes exit and get respawned (this is set to 64 by default). When the processes are stopped and restarted, the memory they use is freed and then reused.

By stopping and restarting a process, the proxy ensures that memory isn't wasted by "lost" processes. For example, on rare occasions, a connection might be terminated while the proxy is reading the HTTP header, but when the request fails, the memory isn't freed.

ErrorLog

ErrorLog specifies the directory where the server logs its errors. You can also use the *syslog* facility. If errors are reported to a file (instead of *syslog*), then the file and directory in which the log is kept must be writable by whatever user account the server runs as.

SYNTAX

ErrorLog *logfile*

logfile can be either a full path and filename or the keyword SYSLOG (it must be in all capital letters).

DEFAULT

There is no default error log.

EXAMPLES

```
ErrorLog /var/ns_proxy/logs/errors
ErrorLog SYSLOG
```

PidLog

PidLog specifies a file in which to record the process ID (PID) of the base proxy server process. Some of the server support programs (including the forms-based Administration Manager) assume that the PID log is in the server root, in *logs/pid*.

To shut down your server, kill the base server process listed in the PID log file by using a -TERM signal. To tell your server to reread its configuration files and reopen its log files, use *kill* with the -HUP signal.

If the PidLog file isn't writable by the user account that the server uses, the server does not log its process ID anywhere.

SYNTAX

PidLog *file*

file is the full pathname and filename where the process ID is stored.

DEFAULT

There is no default.

EXAMPLES

```
PidLog /var/ns_proxy/logs/pid
PidLog /tmp/ns-proxy.pid
```

LoadObjects

LoadObjects specifies one or more object configuration files to use on startup; these files tell the server the kinds of URLs to proxy and cache.

Although you can have more than one object configuration file, the proxy's Administration forms work only with one file and assume that it is in the

server root in *admin/config/obj.conf*. If you use the Administration forms (or plan to), don't put the *obj.conf* file in any other directory and don't rename it.

SYNTAX

LoadObjects *filename*

filename is either the full pathname or a relative pathname. Relative pathnames are resolved from the directory specified with the **-d** command line flag. If no **-d** flag was given, the server looks in the current directory.

DEFAULT

There is no default.

EXAMPLES

```
LoadObjects obj.conf
LoadObjects /var/ns_proxy/admin/config/local-objs.conf
```

RootObject

RootObject tells the server which object loaded from an object file is the server default. The default object is expected to have all of the name translation directives for the server; any server behavior that is configured in the default object affects the entire server.

If you specify an object that doesn't exist, the server doesn't report an error until a client tries to retrieve a document. The Administration forms assume the default to be the default named object. Don't deviate from this convention if you use (or plan to use) the Administration forms.

SYNTAX

RootObject *name*

name is the name of an object defined in one of the object files loaded with a LoadObjects directive.

DEFAULT

There is no default.

EXAMPLES

```
RootObject default
RootObject server1
```

Chroot

Chroot lets the IRIX system administrator place the proxy server into a jail where it has access only to files in a given directory. This is useful if the server's security is ever compromised. For example, if an intruder somehow obtains shell access on the server host, the intruder could affect only a very limited set of files on the server host.

The server must be started as the superuser to use the Chroot directive. A server using Chroot can't be restarted with the -HUP signal, and it can't be configured with the Netscape Proxy Manager forms.

Logs and server configuration files should be kept outside the Chroot directory, but the cache root must be under the Chroot directory.

Note: All paths in *magnus.conf* must be absolute; paths in *obj.conf* must be relative to the Chroot directory.

SYNTAX

Chroot *directory*

directory is the full pathname to the directory used as the server's root directory.

DEFAULT

There is no default.

EXAMPLES

```
Chroot /d/ns_proxy
Chroot /www
```

Init

Init is a special directive that initializes certain proxy subsystems such as the networking library, caching module, and access logging. The functions referenced with the Init directive load data for specific subsystems once on server startup and keep that data internally until the server is shut down. You can specify zero or more Init directives.

SYNTAX

Init fn=*function-name [parm1=value1]…[parmN=valueN]*

function-name identifies the server initialization function to call. These functions shouldn't be called more than once.

parm=value pairs are values for function-specific parameters. The number of parameters depends on the function you use. The order of parameters doesn't matter.

Init functions are described in detail in the following sections. Briefly, they are

- **init-proxy** initializes the networking code used by the proxy.
- **init-cache** enables and initializes caching.
- **init-sockd** enables and initializes the SOCKD feature.
- **load-types** maps file extensions to MIME types.
- **init-clf** initializes the Common Log subsystem.

init-proxy

The function init-proxy initializes the networking code used by the proxy. Calling this function in *magnus.conf* is crucial (even though it is called automatically, you shouldn't rely on this feature).

PARAMETERS

timeout (optional) is the number of seconds after which the proxy should time out if it doesn't receive any new data from the remote server. This is *not*

the maximum time allowed for the entire document transfer—it is the time between successive data packets. The timeout is set to 60 seconds by default.

log-format (optional) is either common or extended. It sets the log format used for proxy accesses. The default value is common.

suppress (optional) is a wildcard pattern for request header lines that should be filtered out from the request and not be forwarded to the remote server.

EXAMPLES

```
Init fn=init-proxy log-format=extended timeout=45
Init fn=init-proxy log-format=common timeout=30
Init fn=init-proxy log-format=common timeout=30
suppress="(from|user-agent)"
```

init-cache

The function init-cache enables and initializes the caching system. It should be called only once if caching is enabled—calling this function is crucial if you want to use caching.

If you are using the Administration forms, you shouldn't use continuation lines in the *magnus.conf* file. Instead, put each Init configuration entirely on a single line. If you are absolutely sure you will never use the Administration forms to edit the *magnus.conf* file, you can use the backslash character to continue to the next line.

PARAMETERS

cache-root specifies the directory under which the cache resides. This directory must contain the Cache hierarchy, built either at installation time or through the Administration interface.

cache-size is the maximum allowed size of the cache in megabytes. When this size is reached, emergency garbage collection occurs.

cache-protocols is a comma-delimited list of protocols that can be cached (HTTP, FTP, and Gopher). Caching for a given protocol must be enabled this way before any document with that protocol can be cached.

gc-times is a comma-delimited list of 24-hour-clock times (for example, 20:30) that specifies when to run garbage collection.

lock-timeout (optional) is the time in seconds after which a hung cache lock can be broken. The default value is 1800 (30 minutes). See "Cache Lock Timeout" on page 23 for more information on how this can happen.

top-dirs (optional) is the number of top-level cache directories (a, b, c, …) under the cache-root directory. It must be a power of two (1, 2, 4, 8, or 16) and the value must reflect the number of actual directories in the cache root (using a conflicting value causes badly degraded cache performance). The default value is 16.

Note: To explicitly not cache a document, you create an object for that document, and then set caching parameters for that object. See "ObjectType" on page 98 for more information.

cache-mode (optional) has either the value all or nothing. "All" caches all documents unless you explicitly specify not to. Nothing caches no documents unless you explicitly specify that the given document should be cached. The default value is all.

max-uncheck is the number of seconds allowed between successive up-to-date checks. If the value is zero (default), an up-to-date check is always performed.

lm-factor is a floating point number used to estimate document expiration based on the last-modified date of the document. If the value is zero (default), the last-modified estimation is turned off and only explicit Expires HTTP headers are used. This value has no effect if max-uncheck is set to zero.

gc-nice is an integer from 0 to 39 that specifies the *nice value* of the garbage collector process. Zero (default) means normal priority; 39 means the lowest priority.

connect-mode specifies network connectivity and can be set to

- normal (default) connects to remote servers when necessary.
- fast-demo connects only if the item isn't found in the cache.

- never returns an error if the document is not found in the cache, but no connection to a remote server is ever made.

cover-errors, if present and greater than zero, returns a document from the cache if the remote server is down and an up-to-date check cannot be made. The value specified is the maximum number of seconds since the last up-to-date check; if more time has elapsed, an error is returned. Using this feature involves a risk of getting stale data from the cache while the remote server is down. Setting this value to zero, or not specifying it (default) causes an error to be returned if the remote server is unavailable.

EXAMPLES

```
Init fn=init-cache cache-root=/var/mc-cache cache-size=500 \
     cache-protocols="http,ftp,gopher" gc-times="3:00,15:00"
Init fn=init-cache cache-root=/var/mc-cache cache-size=500 \
     lock-timeout=1200 top-dirs=16 \
     cache-protocols="http,ftp,gopher"\
     cache-mode=all max-uncheck=86400 lm-factor=0.02 \
      gc-times="3:00" gc-nice=4 \
connect-mode=normal cover-errors=3600
```

init-sockd

The init-sockd function enables and initializes the Netscape Proxy SOCKD feature. It makes the proxy behave like a SOCKS daemon in addition to its normal tasks.

The SOCKD configuration is done through the normal SOCKD configuration file, */etc/sockd.conf*. The same care must be taken as when configuring any other SOCKS daemon.

PARAMETERS

sockd-conf (optional) specifies the SOCKD configuration file to use. The default is */etc/sockd.conf*. The format follows the standard SOCKD configuration file format.

ident-check (optional) specifies the type of remote identity checking. These are the possible values:

- strict means an identity check is required, and if the remote host is not running identd, the request is denied.
- loose means an identity check is required, but if the remote host is not running identd, access is granted anyway.
- none means an identity check is not performed.

log-type (optional) specifies where the proxy SOCKD module stores the access information. The directive can be

- common uses the common log format and the same log file as other accesses. This is the default.
- common-separate uses the common log format, but puts the entries in a separate log file (the log file name is given in the log-name parameter).
- syslog uses the traditional SOCKD format and reports to the *syslog* facility.
- syslog-separate uses the SOCKD *syslog* format, but instead of *syslog*, it uses a separate log file as specified by the log-name parameter.

log-name specifies the absolute pathname for a separate log file used for logging SOCKS accesses. This name is required when the log-type is either common-separate or syslog-separate.

EXAMPLES

```
Init fn=init-sockd sockd-conf=/etc/sockd.conf log-type=syslog
Init fn=init-sockd log-type=common-separate
                   log-name=/d/proxy/logs/sockd.log
```

load-types

The function load-types scans a file that tells it how to map filename extensions to MIME types. MIME types are essential for network navigation software like Netscape Navigator to tell the difference between file types. For example, they are used to tell an HTML file from a GIF file. See “mime.types File” on page 104 for more detailed information.

Calling this function is crucial if you use the Administration forms or the FTP proxying capability.

PARAMETERS

mime-types specifies either the full path to the global MIME types file or a filename relative to the server configuration directory. The proxy server comes with a default file called *mime.types*.

local-types is an optional parameter to a file with the same format as the global MIME types file, but it is used to maintain types that are applicable only to your server.

EXAMPLES

```
Init fn=load-types mime-types=mime.types
Init fn=load-types mime-types=/tp/mime.types \
        local-types=local.types
```

init-clf

The function init-clf initializes the Common Log subsystem. It opens the log files whose names are given as parameters. The log files stay open until the server is shut down or until the base server process is sent the -HUP signal (at which time the logs are closed and reopened).

You use this function to specify which log files the proxy uses to record transactions. Then, you use the AddLog directive in the *obj.conf* file to specify the log file where the proxy stores the transaction record.

Note: Initializing this function is required if you are using the common log or extended log features.

For example, you use init-clf to specify a name that refers to a log file (such as http-log=/var/ns_proxy/loghttp); then you use the name with the AddLog function in *obj.conf* to add a log entry to the file (such as AddLog fn=proxy-log name=http-log). If you ever change the path or filename of the log file, you do it only once—in *magnus.conf*—instead of multiple times in *obj.conf*.

You also can use AddLog to store transactions in more than one log file. See "AddLog" on page 102 for more information about the AddLog function.

If you move, remove, or change the log file without shutting down or restarting the server, client accesses might not be recorded. To save or back up a log file, rename the file and then send the -HUP signal to restart the proxy. The proxy uses the inode number, but when you do a soft restart, the proxy first looks for the filename, and if it doesn't find the log file, it creates a new one (the renamed original log file is left for you to use).

PARAMETERS

At least one log file should be given. The parm part of the *parm=value* pair should be a unique name for the log file. You will use this name later on, as a parameter to the proxy-log function.

EXAMPLES

```
Init fn=init-clf global=/var/ns_proxy/logs/access
Init fn=init-clf global=/tmp/proxy-access
```

obj.conf File

The object configuration file, called *obj.conf*, uses objects to control how the server performs access control, configures the cache, and routes URLs.

Objects (also referred to as resources) are settings that tell the proxy how to treat all URLs. URLs matching a specified wildcard pattern belong to the same object (or resource). This object grouping can then be used to control (in fine detail) the behavior of the proxy server.

Using this object grouping scheme, you can specify single resources with their complete URL, whole "directories" with the path followed by /*, and various other groups such as *.html. You can then configure the settings you want to use for that object (for example, caching or denying access based on the server's hostname or a string within a URL).

Note: All pathnames must be relative if you use Chroot. See "Chroot" on page 79 for more information.

Structure of obj.conf

The *obj.conf* file must have four specific objects in it (see "Required Objects for obj.conf" on page 90 for a description of the objects). You can add other objects to this file. To specify an object, use the following format:

```
<Object ppath=wildcardpattern>
Directives
…
  <Client dns=wildcardpattern>
    Directives
   …
  </Client>
</Object>
```

You use wildcard patterns to control which URLs are grouped in the object. You then specify one or more directives to control what the proxy server does when it encounters any URL matching the wildcard pattern specified with ppath.

You can also set options for specific client hosts. This is a powerful feature because, unlike other proxy servers where a host either can or cannot access a URL, you make the proxy act differently for a client depending on the URL the client sends to the proxy. Although you don't need any <Client> sections in an object section, you can specify more than one—so the proxy acts differently based both on who requests something and what the request is.

Directive Syntax

Each directive line (regardless of where it appears) has the format:

`Directive` *fn=function [parameter1=value1]…[parameterN=valueN]*

Directive identifies an aspect of server operation. This string is case-insensitive and must appear at the beginning of a line. *Function* is a function and parameters given to the directive. Its format depends on the directive.

Directive lines cannot contain white space at the beginning of the line and between the directive and value, but trailing white space after the value might confuse the server. Long lines (which should occur only with the Init

directive) can be continued with a \ character before the linefeed. Comment lines begin with a # character with no leading white space.

Sample Object

The following sample object applies to all HTTP URLs (the wildcard is http://*).

When the proxy receives a request for an HTTP document, it scans the URL for the characters *play* (PathCheck); if it finds that string in the URL, it doesn't retrieve the document from the remote server (it denies service to the client).

```
<Object ppath="http://*">
PathCheckfn=deny-service
path="*play*"
ObjectTypefn=cache-setting
max-uncheck=14400
lm-factor=0.1
Servicefn=proxy-retrieve
</Object>
```

This object also caches all HTTP documents and refreshes the documents if they are older than four hours or if they need refreshing as determined by the date they were last modified. The Service directive tells the proxy to retrieve the HTTP documents by default.

Example 7-2 Example obj.conf File

```
#
# Sample obj.conf file for Netscape Proxy 1.1.
#
######################################################################
# The default object
######################################################################
# The default object is a special named object that applies to all URLs
<Object name="default">
# Authenticate the proxy user
AuthTrans fn="proxy-auth" auth-type="basic"
dbm="/var/ns_proxy/admin/userdb/pw-file"
#
# Make FTP URLs that are of file:// form to work.
NameTrans fn="map" from="file:" to="ftp:"
#
```

```
# Mappings for the admin interface and icons -- these aren't actually
# proxy configuration directives but features of a stripped down
# HTTP server that the Netscape Proxy contains for this purpose
NameTrans fn=pfx2dir from=/admin/bin dir=/var/ns_proxy/admin/bin  name=cgi
NameTrans fn=pfx2dir from=/admin     dir=/var/ns_proxy/admin/html name=file
NameTrans fn=pfx2dir from=/ns-icons  dir=/var/ns_proxy/ns-icons   name=file
#
# Allow proxy accesses only from inside the sgi.com domain.
# That is, anything outside of that domain is denied service to the proxy.
<Client dns="*~*.sgi.com" ip="*~(198.93.*.*|198.95.*.*)>
PathCheck fn=deny-service
</Client>
#
# Require all proxy users to authenticate to the proxy;
# grant access to any successfully authenticated user.
PathCheck fn=require-proxy-auth realm="Firewall proxy server" auth-type=basic auth-user=*
#
# Default action is to deny access (the remaining objects explicitly allow it).
Service fn=deny-service
#
# Log the access
AddLog fn=proxy-log
#
# End of default object
</Object>
##############################################################################
# ppath objects
##############################################################################
# Allow FTP to be proxied through this proxy;
# retrieve a new copy if older than 10 hours.
<Object ppath="ftp://*">
ObjectType fn=cache-setting max-uncheck=36000
Service fn=proxy-retrieve
</Object>
#
# Allow HTTP to be proxied; use the max-uncheck setting in the
# cache-init function (1 hour).
<Object ppath="http://*">
Service fn=proxy-retrieve
</Object>
#
# Allow HTTPS URLs to be proxied (secure documents don't get cached)
<Object ppath="https://*">
Service fn="proxy-retrieve"
</Object>
```

```
#
# Allow Gopher to be proxied; use the max-uncheck setting in the
# cache-init function (1 hour).
<Object ppath="gopher://*">
Service fn=proxy-retrieve
</Object>
#
# Allow SSL proxying, i.e. CONNECT requests to remote 443 ports
<Object ppath="connect://*:443">
Service fn=connect method=CONNECT
</Object>
#
# The following three objects are the standard settings to make the
# admin interface and the icons work, and admin interface protected
# by the admin password. Do not change!
<Object name=cgi>
PathCheck fn=unix-uri-clean
PathCheck fn=find-pathinfo
ObjectType fn=force-type type=magnus-internal/cgi
Service fn=send-cgi
</Object>

<Object name=file>
PathCheck fn=unix-uri-clean
PathCheck fn=find-index index-names=index.html
ObjectType fn=type-by-extension
ObjectType fn=force-type type=text/plain
Service fn=send-file
</Object>

<Object ppath="/var/ns_proxy/admin/*">
AuthTrans fn=basic-ncsa auth-type=basic userfile=/var/ns_proxy/admin/config/admpw
<Client dns="*~(helium|carbon|uranium).sgi.com">
PathCheck fn=deny-service
</Client>
PathCheck fn=require-auth auth-type=basic realm="Netscape Proxy Administration"
</Object>
```

Required Objects for obj.conf

There are certain objects that must be in the *obj.conf* file to make the Administration forms work for your proxy. If you are familiar with the

normal Netscape Server software (a regular HTTP server), you might notice that these functions control local file access and CGI execution.

Note: If you don't use the Administration forms, these objects don't necessarily need to be in *obj.conf*.

However, on a proxy server the local access interface is a simplified version, and it cannot be used for any purpose other than the Administration forms. Special care is taken inside the proxy software to guarantee that it cannot be used, accidentally or otherwise, as a normal HTTP server.

The following sections describe the objects that must be in *obj.conf*.

The Default Object

```
<Object name="default">
NameTrans fn="map" from="file:" to="ftp:"
NameTrans fn="pfx2dir" from="/admin/bin"
dir="ServerRoot/admin/bin"  name="cgi"
NameTrans fn="pfx2dir" from="/admin"
dir="ServerRoot/admin/html" name="file"
NameTrans fn="pfx2dir" from="/mc-icons"
dir="ServerRoot/mc-icons"   name="file"
Service fn="deny-service"
AddLog fn="proxy-log"
</Object>
```

Note: ServerRoot stands for the proxy installation directory, for example */var/ns_proxy*.

The first NameTrans directive takes care of URLS that use file: by changing them to ftp: URLs. If you have any mappings to mirror sites, put them after this mapping.

The next three NameTrans directives map admin and icon URLs into their respective directories. These are the only legal uses for the pfx2dir function, which doesn't belong to the actual proxy configuration (you'll get errors if you try to use it anywhere else).

The deny-service function ensures that by default, access isn't granted. (Access isn't granted by default even if you forgot this function, but the error

message would be less descriptive, and it would be classified as a misconfiguration.)

The proxy-log function takes care of proxy access logging, whether it is in common or extended log format. This function can have the additional iponly=1 parameter, which inhibits reverse DNS lookups and logs only the IP address of the requesting client.

CGI Object

This object controls the admin form handler scripts and should read exactly as follows:

```
<Object name="cgi">
PathCheck fn="unix-uri-clean"
PathCheck fn="find-pathinfo"
ObjectType fn="force-type" type="magnus-internal/cgi"
Service fn="send-cgi"
</Object>
```

Local Access Object

This object controls HTML Administration forms and internal images. It should read exactly as follows:

```
<Object name="file">
PathCheck fn="unix-uri-clean"
PathCheck fn="find-index" index-names="index.html"
ObjectType fn="type-by-extension"
ObjectType fn="force-type" type="text/plain"
Service fn="send-file"
</Object>
```

Admin Interface Protection

The Administration forms are protected by default; the following object configures the protection. (ServerRoot stands for the proxy installation directory.)

```
<Object ppath="/var/ns_proxy/admin/*">
AuthTrans fn="basic-ncsa"
userfile="/var/ns_proxy/admin/config/admpw"
auth-type="basic"
```

```
PathCheck fn="find-index" index-names="index.html"
<Client ip="*~127.0.0.1" dns="*~*.your.domain">
PathCheck fn="deny-service"
</Client>
PathCheck fn="require-auth" realm="Netscape Proxy
Administration"
auth-type="basic"
</Object>
```

What Are Named Objects?

Named objects are objects identified by <Object name=…> in the object configuration file.

To control the behavior of the entire server, you would modify the setting for the default object. This object must contain all of the name-translation directives for the server, and it should contain any global configuration changes.

How the Proxy Server Handles Objects

The Netscape design (both the HTTP Server and the Proxy) breaks down the process of responding to an information request. Each step in the process is done once for all objects, then another step is done for all objects, and so on. The process steps are as follows:

1. Authorization translation. Translate any authorization information given by the client into a user and group. If necessary, decode the message to get the actual request. Also, proxy authorization is available.
2. Name translation. Before anything else is done, a URL can be translated into a filesystem-dependent name (an administration URL), a redirection URL, mirror site URL, or it might be kept intact and retrieved as-is (the normal proxy case).
3. Path checks. Perform various tests on the resulting path. Largely used to make sure that it's safe for the given client to retrieve the document (only for local access).

4. Object types. Determine the MIME type information for the given document. MIME types can be registered document types such as *text/html* and *image/gif*, or they can be internal document identification types. Internal types always begin with *magnus-internal/*, and are used to select a server function to use to decode the document. (Only used for local access; the proxy system calls these routines automatically when necessary.)
5. Service. Select an internal server function that should be used to send the result back to the client. This function can be the normal proxy-service routine or a CGI execution for admin forms.
6. Log. Select a function to record information about the transaction that just finished.

These steps map directly to six of the configuration directives allowed for each object. There is a seventh configuration directive (send-error) that controls how the server responds to the client when it encounters an error.

Directives in obj.conf

This section defines the directives and describes their characteristics, including the directive name and description, format for the function string, default value if the directive is omitted, and how many instances of the directive can be in the file. The directives are

- **AuthTrans** protects server resources from specific users.
- **NameTrans** maps URLs to mirror sites and the local filesystem.
- **PathCheck** checks URLs after NameTrans.
- **ObjectType** tags additional information to requests.
- **Service** sends data and completes the requests.
- **AddLog** adds log entries to any log files.
- **Error** sends customized error messages to clients.

AuthTrans

AuthTrans stands for Authorization Translation. Server resources can be protected so that accessing them requires the client to provide certain

information about the person using the client program. This authorization information is encoded to prevent clients from authorizing themselves as a different user.

The server analyzes the authorization of client users into two steps. First, it translates authorization information sent by the client, and then it requires that such authorization information is present. This is done in the hope that multiple translation schemes can be easily incorporated, as well as to provide the flexibility to have resources that record authorization information but do not require it.

If there is more than one AuthTrans directive in an object, all functions will be applied.

Proxy authorization

DESCRIPTION

proxy-auth translates authorization information provided through the basic proxy authorization scheme. This scheme is similar to the HTTP authorization scheme, but doesn't interfere with it, so using proxy authorization doesn't prevent the ability to authenticate to the remote server.

This function is usually used with the PathCheck function require-proxy-auth.

PARAMETERS

"auth-type" specifies the type of authorization to be used. This should always be "basic".

"userfile" specifies the full pathname of the user database in the NCSA-style *httpd* user file format. This format consists of *name*:*password* lines where *password* is encrypted. If you use this parameter, don't use dbm.

"dbm" specifies the full path and base filename of the user database in the server's native format. The native format is a system DBM file, which is a hashed file format allowing instantaneous access to billions of users. If you use this parameter, don't use userfile as well.

"grpfile" (optional) specifies the NCSA-style httpd group file to be used. Each line of a group file consists of group:user1 user2…userN where each user is separated by spaces.

EXAMPLES

```
AuthTrans fn=proxy-auth auth-type=basic
  dbm=/var/ns_proxy/userdb/rs
AuthTrans fn=proxy-auth auth-type=basic
  userfile=/var/ns_proxy/.htpasswd
grpfile=/var/ns_proxy/.grpfile
```

NameTrans

NameTrans stands for Name Translation. This directive maps URLs to mirror sites and to the local filesystem (for the Administration forms).

NameTrans directives should appear in the root object (the "default" object), although you can put them elsewhere. If there is more than one NameTrans directive in an object, the server applies functions until one succeeds, and then modifies the URL to either a mirror site URL or to a full filesystem path.

Map URLs to Mirror Sites

DESCRIPTION

map looks for a certain URL prefix in the URL the client is requesting. If it finds the prefix, it replaces the prefix with the mirror site prefix.

Don't use trailing slashes with the prefixes you specify—they cause `Not Found` errors.

PARAMETERS

"from" is the prefix to map.

"to" is the mirror site prefix that the from prefix is mapped to.

"name" (optional) gives a named object from which to derive the configuration for this mirror site.

EXAMPLES

```
# Map site http://home.sgi.com/ to mirror site
http://mirror.dom
NameTrans fn=map from="http://home.sgi.com"
                  to="http://mirror.dom"
```

PathCheck

PathCheck directives check the URL that is returned after all of the NameTrans directives finish running. Local file paths (with the Administration forms) are checked for things such as CGI path info and for dangerous elements such as ../ and //, and then any access restriction is applied.

If there is more than one PathCheck directive in an object, all of the directives will be applied in the order they appear.

Deny Proxy Access

DESCRIPTION

deny-service is a PathCheck function that sends a `Proxy Denies Access` error when a client tries to access a specific path. If this directive appears in a Client region, then it performs access control on the specified clients.

The proxy specifically denies clients instead of specifically allowing them access to documents (for example, you don't configure the proxy to allow a list of clients). The "default" object is used when a client doesn't match any client region in objects, and because the "default" object uses the deny-service function, no one is allowed access by default.

PARAMETERS

"path" is a wildcard expression of the path to check. Not specifying this parameter is equivalent to specifying *. URLs matching the expression are denied access to the proxy server.

EXAMPLES

```
<Object ppath="http://sgi/*">
# Deny servicing proxy requests for fun GIFs
```

```
PathCheck fn=deny-service path=*fun*.gif
# Make sure nobody except Netscape employees can use the
object
# inside of which this is placed.
<Client dns=*~*.sgi.com>
PathCheck fn=deny-service
</Client>
</Object>
```

ObjectType

ObjectType directives tag additional information to the requests, such as caching information and if another proxy should be used.

If there is more than one ObjectType directive in an object, all of the directives are applied in the order they appear. If a directive sets an attribute and a later directive tries to set that attribute to something else, the first setting is used and the subsequent ones ignored.

Use Another Proxy for Retrieving This Resource

DESCRIPTION

set-proxy-server is an ObjectType function that directs the Netscape Proxy to connect to another proxy for retrieving the current resource.

PARAMETERS

“hostname” is the name of the host that the other proxy is running on.

“port” is the port number of the remote proxy.

EXAMPLES

```
ObjectType fn=set-proxy-server hostname=proxy.sgi.com
port=8080
```

Use a SOCKS Server When Retrieving This Resource

DESCRIPTION

set-socks-server is an ObjectType function that tells the Netscape Proxy to make the connection to the remote server using a SOCKS daemon when retrieving the current resource.

PARAMETERS

“hostname” is the name of the host that the SOCKS daemon runs on.

“port” is the port number of the SOCKS daemon.

EXAMPLES

```
ObjectType fn=set-socks-server hostname=socks.sgi.com port=1080
```

Specify Caching Parameters

DESCRIPTION

cache-setting is an ObjectType function that sets parameters used for cache control. Defaults for these settings are provided through the init-cache function.

PARAMETERS

max-uncheck (optional) is the maximum time (in seconds) allowed between consecutive up-to-date checks. If set to zero, a check is made every time the document is accessed.

lm-factor (optional) is a floating point number specifying the factor used in expiration-time estimation (how long a document might be up to date based on the time it was last-modified). The time elapsed since the last modification is multiplied by this factor, and the result gives the estimated time for the document to remain unchanged. Specifying a value of zero turns this off, and the caching system uses only explicit expiration information (rarely available). This value has no effect if max-uncheck is set to zero.

cache-mode can be either all or nothing. This overrides the default cache-mode set in the init-cache function. So, if the default cache-mode is on (all documents are cached unless explicitly inhibited), using the value

"nothing" inhibits caching for the specified object. Similarly, to cache a resource (object) on a proxy that isn't caching by default, use the value all.

EXAMPLES

```
# Force check every time
ObjectType fn=cache-setting max-uncheck=0
# Check every 30 minutes, or sooner if changed less than
# 6 hours ago (factor 0.1; last change 1 hour ago would
# give 6-minute maximum check interval).
ObjectType fn=cache-setting max-uncheck=1800 lm-factor=0.1
# Disable caching of the current resource
ObjectType fn=cache-setting cache-mode=nothing
```

Maximum Length of Cached Queries

DESCRIPTION

cache-queries is an ObjectType function that specifies the maximum length of a query in order for it to be cached (if it has either a Last-modified or an Expires header). The proxy caches the query results, not the query.

PARAMETERS

"maxlen" specifies the maximum length (in characters). The default value is 16. If zero, queries aren't cached.

EXAMPLES

```
# Cache queries not longer than 8 characters
ObjectType fn=cache-queries maxlen=8
# Don't cache queries at all
ObjectType fn=cache-queries maxlen=0
```

Service

Once the other directives have done all the necessary checks and translations, the Service class of functions sends the data (first receiving it from a remote server when necessary) and completes the request. Most of the time, the Service function connects to a remote server making the request for the client and then passing the results back to the client.

Service directives support the following optional parameters to help determine whether the directive is used or not:

- method specifies a wildcard expression of HTTP methods that the client must be using to have the directive applied. Valid HTTP methods are GET, HEAD, and POST. Multiple values are enclosed in parentheses and separated by the pipe (|) symbol.
- type (not with proxy-retrieve) specifies a wildcard expression of MIME types to apply the directive to. The proxy server defines several MIME types internally that are used only to select a Service function that translates the internal type into a form presentable to the client.

If there is more than one Service directive in an object, the first applicable directive is used and the rest are ignored.

Proxy Retrieval

DESCRIPTION

proxy-retrieve retrieves a document from a remote server and returns it to the client. It also manages caching if it is enabled.

By default (in the absence of the method parameter) all methods (GET, HEAD, and POST) are allowed.

PARAMETERS

None.

EXAMPLES

```
# Normal proxy retrieve
Service fn=proxy-retrieve
# Proxy retrieve with POST method disabled
Service fn=proxy-retrieve method=(GET|HEAD)
```

Denying Service

DESCRIPTION

deny-service is the only function that belongs to two classes: PathCheck and Service. It prevents access to the requested resource.

PARAMETERS

"path" (optional) specifies a wildcard expression of the path to check. Not specifying this parameter is equivalent to specifying *. URLs that match the expression deny access to the client.

EXAMPLES

```
# Deny all requests for the current resource
Service fn=deny-service
# Deny POST requests
Service method=POST fn=deny-service
# Deny POST requests for the given site
Service method=POST path=http://some.site/* fn=deny-service
```

AddLog

After the request is finished and the proxy server has stopped talking to the client, the proxy server logs the transaction. The server records information about every access clients make, and it records information about the client making the request.

If there is more than one AddLog directive in an object, all are used.

Log in the Common or Extended Format

DESCRIPTION

proxy-log is an AddLog function that records request-specific data in either the extended or common log format (used by most HTTP servers). There are a number of free statistics generators for the common format, but the extended format gives more detailed information about the bytes transferred and the time elapsed (use *pstats* to interpret the log file; see "Interpreting the pstats Output" on page 58).

The log format is specified by the init-proxy function call.

PARAMETERS

"name" (optional) gives the name of a log file, which must have been given as a parameter to the init-clf Init function. If no name is given, global is assumed.

iponly (optional) instructs the server to skip looking up the host name of the remote client and records the IP address instead. The value of iponly has no significance, as long as it exists; the Administration forms set iponly="1".

EXAMPLES

```
# Log all accesses to the central log file
AddLog fn=proxy-log
# Log non-local accesses to another log file
<Client ip=*~198.93.9[2345].*>
AddLog fn=proxy-log name=nonlocal
</Client>
```

Error

At any time during a request, conditions can occur that stop the server from fulfilling a request and then return an error to the client. When these happen, the server can send a short HTML page to the client describing the error in very general terms.

In order to make error handling more friendly, the proxy server lets you intercept certain errors and send a file of your choosing instead of the server's default error message.

The following is a list of errors that are returned by the server. Each line has the 3-digit HTTP code that designates the error, followed by a short description of the error.

- **401 Unauthorized** (for Administration forms only). The server requires HTTP user authorization to allow access to the Administration forms, and the client either provided none or its HTTP authorization was insufficient.
- **403 Forbidden**. The server tried to access a file or directory, and found that the user it was running as didn't have sufficient permission to access the file.
- **404 Not Found**. The client asked for a filesystem path that doesn't exist, or the server was configured to tell the client that it doesn't exist. If you use access control, changing the response to this error lets you tell people nicely that they don't have that access to your proxy.
- **407 Proxy Authorization Required**. The proxy requires proxy authorization, and the client either didn't provide any, or it was

insufficient. Also, the client software might not support proxy authorization. The Netscape Navigator version 1.1 supports this authorization.

- **500 Server Error**. Server errors mean that an error has occurred within the server that prevents it from finishing the request. Server errors mainly happen because of misconfiguration or host resources such as swap space being exhausted.

EXAMPLES

```
Error fn=send-error code=401 path=/var/ns_proxy/errors/401.html
```

mime.types File

The *mime.types* file tells the server how to convert files with certain extensions (such as *.GIF*) into a Multipurpose Internet Mail Extensions type (such as *image/gif*). MIME files are compact files and transfer quickly. Also, MIME is needed by browsers (like the Netscape Navigator); without MIME they couldn't tell the difference between an HTML page and a graphic file.

The *mime.types* file contains the global file extensions for all proxy servers. The first line in the file identifies the file format and must read:

```
#--Netscape Communications Corporation MIME Information
```

Other non-comment lines have the format

type=*type/subtype* exts=*[file extensions]* icon=*icon*

- type/subtype
- exts are the file extensions associated with this type. When the proxy transfers a file with one of these extensions, it uses the MIME type you specified in type.
- icon is the name of the icon the browser displays. Netscape Navigator keeps these images internally. If you use a browser that doesn't have these icons, the proxy delivers them.

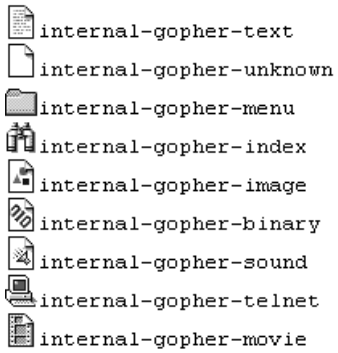

Figure 7-1 Internal Icons for MIME Types

Example 7-3 Example mime.types File

```
#--Netscape Communications Corporation MIME Information
# Don’t delete the above line. It identifies this file’s type.
#
# This is a simple MIME types file for the Netscape Proxy. Most
# of the MIME types are already compiled in the proxy. Types that
# are part of the Administration forms (HTML and GIF) must appear
# here, or they won’t be known to the part of the server that
# manages the Administration interface calls.
#
# Icons (internal-gopher-…) are references to Netscape's
# internal icons. If a client doesn't support these icons, the
# proxy will provide them.
type=application/odaexts=oda
type=application/pdfexts=pdf
type=application/x-mifexts=mif
type=application/x-dviexts=dvi
type=application/x-hdfexts=hdf
type=application/x-netcdfexts=nc,cdf
type=application/x-texinfoexts=texinfo,texi
icon=internal-gopher-text
type=application/zipexts=zip
type=application/x-tarexts=tar
type=application/x-macbinaryexts=bin
type=application/x-stuffitexts=sit
type=image/gif       exts=gif          icon=internal-gopher-image
type=image/jpeg   exts=jpeg,jpg,jpe    icon=internal-gopher-image
type=image/x-xwindowdump   exts=xwd    icon=internal-gopher-image
```

```
type=text/html   exts=htm,html,shtml    icon=internal-gopher-text
type=text/plain  exts=txt               icon=internal-gopher-text
type=text/richtext   exts=rtx           icon=internal-gopher-text
type=text/tab-separated-values exts=tsv icon=internal-gopher-text
type=text/x-setext      exts=etx        icon=internal-gopher-text
type=application/x-tar enc=x-gzip exts=tgz
enc=x-gzip                      exts=gz
enc=x-compress                  exts=z
```

admpw File

The *admpw* file contains the Administration password. If you forget your password, there is no way to find out what it was. You must encrypt a new one and replace the old version with it. The file has the format:

```
user:password
```

If you forget your Administration password, you can edit the *admpw* file and delete the password section (everything after the semicolon). When you go to the Proxy Manager, you don't need to enter a new password—but you should immediately go to the Administration Manager and set a new one.

Caution: Because you can delete the password, it is very important to keep secure the proxy's IRIX id account and to ensure that only that proxy account and the root account have full (write) access to the server root directory. This way, only someone running as root or with the proxy's user account can enter the admin/config directory and edit the file.

sockd.conf File

Although SOCKD doesn't know if a host is internal to its network, it does knows which host is the requestor and which is the destination (it uses this for access control). This means SOCKD provides access from external hosts into your internal networks in addition to the normal internal-to-external proxy functionality. Use caution with the external-to-internal functionality. If you don't need external to internal access, you should specifically deny such connections. For example, if 198.95 is your internal network, use deny 0.0.0.0 0.0.0.0 198.95.0.0 255.255.0.0 as the first line in *sockd.conf* to protect your inside hosts from external access attempts.

Note: The SOCKS daemon is a generic firewall daemon that controls access through the firewall on a point-to-point basis. By default, the SOCKS daemon features are disabled. The Netscape Proxy Server supports SOCKS version 4.

Structure of sockd.conf

The proxy uses the file */etc/sockd.conf* to control access to the SOCKS proxy server SOCKD and its services. Each line in this file can be up to 1023 characters long and has the following format:

```
action [*=userlist] src_addr src_mask [dst_addr dst_mask]
[op dst_port] [ : shell_cmd ]
```

Note: Use spaces or tabs to separate the fields. Text in square brackets is optional. Comment lines begin with #, except for lines that start with #NO_IDENTD: or #BAD_ID:.

Each line defines what the proxy does when it gets a request that matches the line. The line begins with either permit or deny—this defines how the server behaves when it gets a matching request.

When SOCKD receives a request, it checks the request against the lines in */etc/sockd.conf*. When it finds a line that matches the request, the request is either permitted or denied based on the first word in the line. The daemon ignores the remaining lines in the file. If there are no matching lines, the request is denied.

Caution: Because the daemon uses only the first line that matches a request, the order of the lines is important. For example, the following two lines in this order deny access to a user named chris at the specified domain:

```
deny *=chris 198.95.251.30 0.0.0.0
permit 198.95.251.30 0.0.0.0
```

If the two lines switch position, then requests by user chris are granted because the user belongs to the domain listed in the first line.

The rest of this section describes the other elements you can place in a line in the *sockd.conf* file.

userlist

The optional userlist specifies one or more user IDs separated by commas—spaces and tabs aren't allowed. If you don't specify a user list, the other contents of the line applies to all users accessing the proxy.

Unlike other SOCKD applications, this version of the proxy doesn't support filenames in the userlist.

src_addr and dst_addr

You can specify lines that apply to requests based on the source or destination IP address. src_addr and dst_addr specify dotted IP addresses for hosts, networks, or subnets.

The src_mask and dst_mask fields are masks for the corresponding IP addresses. Bits in these masks that are set to 0 indicate the bit positions to be ignored during comparisons of IP addresses. So, specifying 255.255.255.255 in the mask demands an exact match with the specified IP address field, whereas 0.0.0.0 in the mask causes a match no matter what IP address is specified. (This is the same way netmasks are usually interpreted, and is the opposite of the interpretation in previous versions of SOCKD.) If the dst_addr dst_mask pair is omitted, the line applies to all destination hosts.

op dst_port

You can specify lines that apply to requests based on destination ports. If you don't specify a destination port, the line applies to all port numbers. The op section specifies an operator for the destination port number. The operator must be one of the following:

- eq for equals
- neq for not equals
- lt for less than
- gt for greater than
- le for less than or equal to
- ge for greater than or equal to

The destination port (dst_port) can be a port number or a service name specified in */etc/services* (for example, you could use telnet instead of port number 23).

shell_cmd

The shell_cmd specifies a string that runs when the host matches a line in *sockd.conf*. The proxy replaces the following items before the string is sent to the shell:

Table 7-1 String Substitutions in shell_cmd

Item	Replaced by
%A	The client's domain name if known, or its IP address
%a	The client's IP address
%c	*connect* or *bind*, the command SOCKD runs
%p	The process ID of *sockd*
%S	The service name if known, otherwise the destination port number
%s	The destination port number
%U	The user ID reported by *identd*
%u	The user ID reported by the client program
%Z	The destination host's domain name if known, otherwise its IP address
%z	The destination host's IP address
%%	A single %

You can use several shell commands together. For example,

```
/usr/ucb/finger @%A | /usr/ucb/mail -s 'SOCKS: rejected %u@%A to %Z (%S)' root root@%A
```

fingers the client host and pipes the result into an e-mail message for superusers at the server host and the client host with an appropriate Subject line. Most often this feature is used with a deny line, but it can be used with permit also.

Although there is an implied deny all at the end of the configuration file, you can supply one explicitly to take some specific action when requests are so rejected. For example, the following would appear in one continuous line:

```
deny 0.0.0.0 0.0.0.0 : /usr/ucb/finger @%A | /usr/ucb/mail -s 'SOCKS: rejected %u@%A to %Z
(%S)' root root@%A
```

#NO_IDENTD and #BAD_ID

You can also specify commands to run when SOCKD can't connect to a client's *identd* or when the user IDs reported by the client programs and the client's *identd* don't match. These special entries must start with #NO_IDENTD: and #BAD_ID:, followed by the shell commands to run. For example:

```
#NO_IDENTD: /usr/ucb/mail -s 'Please run identd on host %A' root@%A
#BAD_ID: finger @%A | /usr/ucb/mail -s '%U pretends to be %u on %A' root root@%A
```

Appendix A

Proxy Error Log Messages

This appendix defines the errors the proxy sends to the error log file. The errors are categorized:

- **catastrophe** is a fatal error, a software crash, or other serious error that causes the client to receive no service, partial service, or totally invalid service.
- **failure** means something failed, but the proxy handled the error.
- **inform** is an informational log entry.
- **misconfig** means something was misconfigured either in *magnus.conf* or *obj.conf*.
- **warning** flags something that could be misconfigured.
- **security** is information or a warning that indicates if there's reason to believe that someone is trying to intrude through the proxy.

List of Errors

```
CATASTROPHE

cache file size not in sync with cache info
```

The system suddenly went down or the filesystem became full during the cache write, or the cache file has otherwise been truncated. Normally the proxy notices any abnormal conditions, but if an outside agent causes cache files to become corrupt, the proxy will issue this message. The corrupt cache file will be removed, and a new one created during the next request.

```
CATASTROPHE

filesystem is full
```

The cache filesystem has become full. The proxy will halt any cache writes, and an attempt is made to signal the garbage collector to activate immediately. Cache writes are resumed after the condition no longer persists. Consider allocating more space to the cache system.

```
CATASTROPHE

cannot open file .../.cache-size for writing—the cache may
overflow if the condition persists
```

The proxy failed to write the current cache size to the file that contains it. This could be a temporary condition, but if it persists the proxy will not be able to keep track of its cache size. This can cause the cache to overflow on high impact systems. It's possible the write permissions aren't correct for the user account the proxy uses.

```
CATASTROPHE

cannot read header section from the cache file
```

The cache file is truncated, or permissions are such that the cache file cannot be read. Care should be taken that the cache hierarchy is entirely readable and writable by the proxy user.

```
CATASTROPHE

caught SIGSEGV or SIGBUS, trying to dump core in admin/config
```

The proxy encountered an internal software error. Contact Netscape Communications for help with this error. If you rarely encounter this error and it doesn't affect the proxy service, you can ignore this message.

```
CATASTROPHE

failed to write cache status file
```

The write to the cache data directory ([*CacheRoot*]*/.mc-data*) failed. The condition might be temporary (for example, it was caused by a full filesystem), but if it persists, the proxy might stop caching until the situation is fixed. It's possible the permissions for the proxy's user account aren't specified correctly.

```
CATASTROPHE

filesystem permission problem in subdir .mc-data under Cache
Root
```

The filesystem permissions are wrong under the [*CacheRoot*]*/.mc-data*. Care should be taken that the entire CacheRoot directory and recursively all its subdirectories are readable and writable by the proxy user.

```
FAILURE

SOCKS request read failed
```

Client disconnected before the SOCKS request was entirely received by the proxy.

```
FAILURE

accept failed on the bound socket (...)
```

The SOCKS daemon failed to establish the connection from a remote server, requested by the client (SOCKS BIND request).

```
FAILURE

no port number specified for host…
bad port number specified in …
```

Proxy received an invalid SSL proxying request (CONNECT method with number or a bad port number).

```
FAILURE

cache write aborted
```

Cache write was aborted because the remote server failed to send the entire document, the client disconnected, or some other error condition occurred.

```
FAILURE

called with no hostname or address (or corrupt) [SSL proxy]
```

Proxy received an invalid SSL proxying request.

```
FAILURE

can’t create socket (...)
can’t bind (...)
can’t connect (...)
can’t get peer name (...)
can’t get socket name (...)
can’t make ... connection non-blocking
can’t make client socket non-blocking (...)
can’t make connection to ... non-blocking
can’t make identd connection non-blocking (...)
connect failed (...)
connect to ... failed (...)
timed out sending ident request
```

The SSL proxying module or the SOCKS daemon couldn’t successfully execute the system call in question, as part of establishing a connection to either a remote host or a remote identity daemon (*identd*).

```
FAILURE

can’t connect to identd at ... -- access denied
```

Remote host is not running the identity daemon, and strict identity check is enabled.

```
FAILURE

can’t connect to identd at ... -- error ignored
```

Remote host is not running the identity daemon, but loose identity check allows the request to be serviced.

```
FAILURE

can’t find username from ident daemon response (...)
can’t parse response from ident daemon (...)
```

The SOCKS daemon received a bad response from the remote identity daemon.

```
FAILURE

can’t locate SOCKS host
```

The SSL proxy module is unable to locate the SOCKS host it has been configured to contact to establish outbound connections.

```
FAILURE

can’t locate host ...
```

The SSL proxy module is unable to locate the remote host.

```
FAILURE

can’t open SOCKD log ...
```

Can’t open the separate SOCKS daemon log file.

```
FAILURE

cannot create lock file ... (...)
```

The proxy failed to create a lock file; this might happen if the system resources are exhausted or the machine load is so high that the process holding the lock cannot get it. In the short term this error is harmless, and the proxy will automatically recover from it. However, if the condition persists for long periods of time it might cause cache overflow or other abnormal behavior. Check the permissions for the proxy’s user account and the files under the cache root directory.

```
FAILURE

cannot open ... for writing -- caching disabled as    long as
error persists
cannot open cache output file ...
cannot open file .../.cache-size for appending - the cache may
overflow if the condition persists
cannot open gc pid file -- cannot signal gc
cannot remove file ... -- may cause disk full detection to fail
```

The filesystem permissions under the cache root or the server root are wrong so that the proxy cannot open the cache file for writing. On a heavily loaded system this can also be caused by a temporary failure to do disk I/O, in which case this error might be ignored unless the condition persists. In the long-term this error may cause malfunction of the caching subsystem and the garbage collector.

```
FAILURE

cannot signal gc pid ...; running start-proxy to respawn gc
```

After the filesystem full condition, the garbage collector couldn't be signaled to start garbage collection. The proxy will automatically attempt to spawn a new garbage collector process.

```
FAILURE

connection timed out after ... seconds idle
```

The SSL proxy or SOCKS connection has been idle too long.

```
FAILURE

content-length mismatch; too many bytes received
```

The proxy received an incorrect amount of data while it was writing to the cache file. This is due to erroneous behavior on the remote server side, and causes the cache file to be discarded.

```
FAILURE

disconnected by client/server/timeout/internal error
condition with ... bytes in/outbound data undelivered
```

SOCKS or SSL proxying connection was terminated prematurely by one of the parties before all the pending data was transferred.

```
FAILURE

failed to send ident request (sent ... bytes out of ...)
```

The SOCKS daemon was unable to send the identity check request to the remote identity daemon.

```
FAILURE
```

```
internal netlib timeout; process terminated
```

Proxy retrieval lasted too long. This is an internal timeout that cleans up processes that suddenly get blocked due to an unexpected error in the network or one of the proxy subsystems.

```
FAILURE
```

```
method without URI
```

The client sent an invalid proxy request.

```
FAILURE
```

```
proxy retrieve failed: ...
```

A generic message when the retrieval failed due to a mistyped URL, a non-existing hostname, unreachable host or network, disabled or overloaded server, or other unexpected network error.

```
FAILURE
```

```
proxy timeout; closing connection
```

The proxy didn't receive any data from the remote server within the proxy timeout period.

```
FAILURE
```

```
remote closed the connection prematurely (timeout?)
```

The remote server closed the connection before all the data was received, causing the cache write to abort and discard the cache file.

```
FAILURE
```

```
select over the two connections failed (...)
```

Unexpected error in SOCKD or SSL proxying module while passing data through the proxy.

```
FAILURE

unknown SOCKS command (...)
```

The proxy received a bad or unsupported SOCKS command.

```
FAILURE

while scanning proxy HTTP headers, ...
```

An error occurred while reading the request headers from the client.

```
FAILURE

wrong SOCKS protocol version (...)
```

The proxy received a SOCKS request that is not supported in the current version (Netscape Proxy 1.1 supports SOCKS version 4).

```
INFORM

... already locked
```

The cache file is already locked—another process is already writing the cache file. This is merely informative, not an error.

```
INFORM

SOCKD logging shutting down
SOCKD logs to a separate file…in extended common log file format
SOCKD logs to a separate file…in traditional SOCKD SYSLOG format
cache-size sync in progress; abandoning scheduled sync
cache-size sync lock timed out; breaking it
removing timed-out lock file…
signalled gc to start immediately
```

These errors are informative and self-explanatory.

```
SECURITY

SOCKS request when SOCKD features disabled [sockd]
```

Attempt to access the proxy as a SOCKS daemon while the SOCKD feature on the proxy was disabled.

```
SECURITY

attempt to access the proxy as a normal HTTP server, URL: ...
```

Attempt to access the proxy as a normal server; this is usually simply a mistyped admin URL, but might also reflect somebody trying to intrude on the local filesystem of the proxy using evil URLs, such as the ones containing /../. Netscape guards against any such attempts and accessing the local filesystem is impossible, except for the admin interface.

```
SECURITY

denying service of ...
```

Service was denied by configuration.

```
SECURITY

incoming connection came from a wrong host: ..., should have
come from ...
```

With SOCKS BIND command, a wrong host connected to the listener.

```
SECURITY

no matching rule -- refused by default
```

SOCKS request was denied because it wasn't explicitly allowed.

```
SECURITY

refused by deny rule
```

SOCKS request was denied by an explicit rule.

```
SECURITY

username mismatch: real ..., not ...
```

SOCKS identity check failed. The remote user claimed to be someone other than who the remote identity daemon reported.

```
WARNING

terminated, shutting down
```

Proxy was shut down by the TERM signal.

Glossary

cache

A copy of original data stored locally so that it doesn't have to be retrieved again from a remote server when requested.

cache build

The creation of the Netscape cache hierarchy.

cache directory hierarchy

Netscape Proxy's directory structure for storing cache files.

cache refresh

Replacing a cached document with a fresh copy.

cache repair

A process to repair a cache damaged by a software failure, system crash, disk breakdown, or full filesystem.

cache root

A directory on the proxy server that contains all cached files. The proxy controls which documents are copied to the cache root, and the garbage collector daemon purges this directory structure to control the amount of data stored.

cache up-to-date check

This checks that the copy in the cache is still valid, and if not, refreshes it.

CERN

The European Laboratory for Particle Physics (CERN) invented the World Wide Web to share information among research groups. This is where the CERN proxy prototype was produced.

common log file format

The common log file format is the format used by the server for entering information into the access logs. The format is the same among all of the major servers, Netscape Commerce and Communications servers, CERN *httpd*, and NCSA *httpd*.

DNS

Domain Name System. The system used by systems on a network to associate standard IP addresses (such as 198.93.93.10) with hostnames (such as www.netscape.com). Systems normally get this translated information from a DNS server, or look it up in tables maintained on their systems.

DNS alias

A DNS alias is a hostname that the DNS server knows points to a different host—specifically a DNS CNAME record. Systems always have one real name, but they can have one or more aliases. For example, www.[*yourdomain*].[*domain*] might be an alias that points to a real system called realthing.[*yourdomain*].[*domain*] where the server currently exists.

EMACS

A text editor that can also be used to read e-mail and news.

expires header

The expiration time of the returned document, specified by the remote server.

extended log file format

Similar to the common log file format, but it contains more information.

file extension

The last section of a filename that typically defines the type of file (for example, GIF and HTML). For example, in the filename *index.html* the file extension is *html*.

file type

The format of a given file. For example, a graphics file doesn't have the same file type as a text file. File types are usually identified by the file extension (for esample, *.GIF* or *.HTML*).

garbage collector

A process to periodically clean up the cache—for example it removes old files to make room for new ones.

garbage collector daemon

A process that monitors the cache size and spawns the garbage collector when necessary.

GIF

A cross-platform image format originally created by CompuServe. The acronym stands for Graphics Interchange Format. GIF files are usually much smaller in size than other graphic file types (BMP, TIFF). GIF is one of the most common interchange formats. GIF images are readily viewable on UNIX, Microsoft Windows, and Apple Macintosh systems.

hard restart

Terminating the process, and starting it up again.

hostname

A name for a host of the form host.domain.dom, which is translated into an IP address. For example, www.netscape.com is the system www in the subdomain netscape and com domain.

HTML

Hypertext Markup Language is a formatting language used for documents on the World Wide Web. HTML files are plain text files with formatting codes that tell browsers such as the Netscape Navigator how to display text, position graphics and form items, and display links to other pages.

HTTP

Hypertext Transfer Protocol is the method for exchanging information between HTTP servers and clients.

HTTPD

An abbreviation for the HTTP daemon, a program that serves information using the HTTP protocol. The Netscape Communications Server is often called an *httpd*.

HTTPS

A secure version of HTTP, implemented using the secure sockets layer, SSL.

IP address

Internet Protocol address—a set of numbers, separated by dots, that specifies the actual location of a host on the Internet.

last-modified header

The last modification time of the document file, returned in the HTTP response from the server.

MD5

A message digest algorithm by RSA Data Security, Inc., which can be used to produce a short digest of data of any size, which is unique with high probability, and for which it is mathematically extremely hard to produce a piece of data that will produce the same message digest.

MD5 signature

A message digest produced by the MD5 algorithm.

MIME

Multi-Purpose Internet Mail Extensions. This is an emerging standard for multimedia e-mail and messaging.

NIS

Network Information Service—a system of programs and data files that IRIX systems use to collect, collate, and share specific information about systems, users, filesystems, and network parameters throughout a network of computers.

NCSA

The National Center for Supercomputing Applications is a research organization at the University of Illinois at Urbana-Champaign.

Password file

A file on UNIX systems that stores UNIX user login names, passwords, and user ID numbers. It is also known as */etc/passwd*, because of where it is kept. The proxy also has its own password files for user authentication; these are *not* connected with UNIX users.

proxy server

A proxy server stands between the originating and receiving hosts, performing specified monitoring and control operations for an application protocol.

RAM

Random Access Memory. The physical semiconductor-based memory in a computer.

rc.local

A file that describes programs that are run when the system starts. It is also called */etc/rc.local* because of its location.

realm

A term used in HTTP and proxy access authorization that helps the user identify what part of the system is asking for an HTTP or proxy user name and password. For example, the realm in the Netscape Proxy Server Manager is “Netscape Proxy Administration.”

redirection

A system by which clients accessing a particular URL are sent to a different location, either on the same server or on a different server. This is useful if a resource has moved and you want the clients to use the new location transparently. It’s also used to maintain the integrity of relative links when directories are accessed without a trailing slash.

resource

Any document (URL), directory, or program that the server can access and send to a client that asks for it.

root

The most privileged user available on IRIX systems. The root user has complete access privileges to all files on the system.

server daemon

The server daemon is a process that, once running, listens for and accepts requests from clients.

server root

A directory on the server system dedicated to holding the server program, configuration, maintenance, and information files.

SOCKS

Firewall software that establishes a connection from inside a firewall to the outside when direct connection would otherwise be prevented by the firewall software or hardware (for example, the router configuration).

soft restart

Causes the server to internally restart, that is, reread its configuration files, by sending the process the HUP signal (signal number one). The process itself does not die, as it does in hard restart.

SSL

Secure Sockets Layer. A software library establishing a secure connection between two parties (client and server) used to implement HTTPS, the secure version of HTTP.

superuser

The most privileged user available on UNIX systems (also called root). The superuser has complete access privileges to all files on the system.

telnet

A protocol where two systems on the network are connected to each other and support terminal emulation for remote login.

timeout

A specified time after which the server should give up trying to finish a service routine that appears hung.

top

A program that shows the current state of system resource usage. See also gr_top(1).

top-level domain authority

The highest category of hostname classification, usually signifying either the type of organization the domain is (.com is a company, .edu is an educational institution) or the country of its origin (.us is the United States, .jp is Japan, .au is Australia, .fi is Finland).

UID

A unique number associated with each UNIX user on a system.

URL

Uniform Resource Locator—the addressing system used by the server and the client to request documents. It is often called a location. The format of a URL is [*protocol*]://[*system:port*]/[*document*]. A sample URL is *http://www.netscape.com/index.html*.

URL database

A database in the Netscape cache that contains all the URLs found in the cache and links them to the cache files. This database can be browsed using the Cache Manager.

URL database repair

A process that repairs and updates a URL database that has been damaged by a software failure, a system crash, a disk breakdown, or a full filesystem.

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

We'd Like to Hear From You

As a user of Silicon Graphics documentation, your comments are important to us. They help us to better understand your needs and to improve the quality of our documentation.

Any information that you provide will be useful. Here is a list of suggested topics to comment on:

- General impression of the document
- Omission of material that you expected to find
- Technical errors
- Relevance of the material to the job you had to do
- Quality of the printing and binding

Please include the title and part number of the document you are commenting on. The part number for this document is 007-2911-001.

Thank you!

Three Ways to Reach Us

The **postcard** opposite this page has space for your comments. Write your comments on the postage-paid card for your country, then detach and mail it. If your country is not listed, either use the international card and apply the necessary postage or use electronic mail or FAX for your reply.

If **electronic mail** is available to you, write your comments in an e-mail message and mail it to either of these addresses:

- If you are on the Internet, use this address: techpubs@sgi.com
- For UUCP mail, use this address through any backbone site: *[your_site]*!sgi!techpubs

You can forward your comments (or annotated copies of manual pages) to Technical Publications at this **FAX** number:

415 965-0964