Netscape Commerce and Communications Servers Administrator's Guide

Document Number 007-2909-001

Netscape Commerce and Communications Servers Administrator's Guide
Document Number 007-2909-001

Contents

List of Examples

List of Figures

List of Tables

Introduction

Welcome to the Netscape Commerce and Communications Servers for Silicon Graphics, Inc. systems. These servers let people and companies exchange information and conduct commerce over the Internet and other global networks.

The *Netscape Commerce and Communications Servers Administrator's Guide* is intended for system and network administrators who install, configure, and manage the Netscape server(s) at your site. An administrator's knowledge of IRIX is assumed. This guide documents both the Commerce and the Communications servers. The Communications server operates without security; everything else is the same for both servers.

About This Guide?

This manual explains how to configure the Netscape Commerce and Communications Servers. Refer to your software release notes for information on how to use *inst*(1M) to install the software if it is not already installed. After you configure your server, you can use this manual to help you maintain your server.

The *Netscape Servers Programmer's Guide* contains information about programming functions and *CGI* scripts for use with your server.

What Is an HTTP Server?

A HyperText Transport Protocol (HTTP) server is an application that sends documents and data from the computer host it's installed on to HTTP-compatible client applications such as Netscape Navigator. HTTP runs on any TCP/IP network, including the Internet. The Netscape Commerce Server can send the information securely by using the Secure

Sockets Layer (*SSL*) protocol. The Netscape Communications Server does not support the SSL protocol.

The following figure shows a client application requesting information from your HTTP server, which in turn sends the information back to the client. Note that with an unsecure server, anyone on the Internet could potentially intercept and view the information.

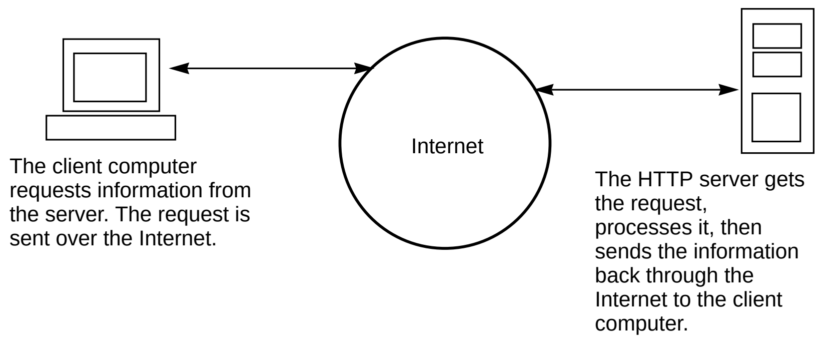

Internet
The client computer requests information from the server. The request is sent over the Internet.
The HTTP server gets the request, processes it, then sends the information back through the Internet to the client computer.

Figure i Sending Information Between Client and Server

The Netscape Servers

The following features are a partial list of what the Netscape Commerce and Communications Servers offer:

- Remote server administration. You can start, stop, configure, and manage the server from any host that has Internet access.
- Reliability. You can configure the server to restart itself after a network crash, power interruption, or other event that stops the server. However, this doesn't work with a secure server because someone must enter a password when starting or stopping the secure server.
- Flexible access control. You can configure the server to restrict access based on a hostname, the document requested, or both. You can also hide *URLs* from any user based on the user's hostname.
- Customized access logging. You can track all the users who access your server, or you can configure access logging on a file, directory, or template basis. It 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.

- Performance and extensibility. The server handles very high numbers of incoming requests with a relatively low impact on the server host.

Internet and Web Basics

If you're new to the Internet and World Wide Web (the Web or WWW), you should familiarize yourself with navigating through information. The Internet is a vast collection of computers working together to share information. There is no single company or person in charge of running the Internet. Because of this design, the information available through the Internet changes constantly and new computers are continually added to the network, which provides you with a growing source of information.

Like the Internet, the World Wide Web also contains information, but it adds multimedia access—graphics, sound, movies, and more. Because the Web is graphical (not text-based), you can access it only with a graphical application such as the Netscape Navigator.

To get information from the Internet and the Web, you need to know the address of the information. Addresses are called Uniform Resource Locators (URLs).

Understanding URLs

Internet and World Wide Web addresses are slightly different. A simple Internet address is *username@domainname* where *username* can be your name, and *domainname* is the name of your service provider (for example, jdoe@sgi.com could be an address for Jane Doe at Silicon Graphics, Inc.).

Domain names can be quite long and usually refer to subgroups or departments in an organization. Typically, a domain is classified into one of the following categories:

- com—commercial business (companies)
- edu—education (universities, primary and high schools)

- gov—government (nonmilitary)
- mil—US military
- net—network organizations
- org—miscellaneous organizations

Web addresses are hypertext links to documents where each page has a URL in the format *protocol://host/directory/file*. Each Web address begins with a protocol for the link, typically *http* (hypertext transport protocol) or *https* (secure HTTP); but also *file*, *gopher*, *ftp* (file transfer protocol), *telnet*, and *news* (Usenet news).

The second part of the URL (after the //) is the host (computer) address, a directory path for the host, and a filename (*.html* or *.htm* extensions mean the document is a hypertext markup language document that might contains jumps to other pages). For example, the Netscape home page is *http://home.netscape.com/home/welcome.html*, which means the page is a hypertext (*.html*) document in the *home* directory on the home.netscape.com host.

Understanding Hostnames and IP Addresses

All hosts on the Internet have a 32-bit Internet Protocol (IP) address that identifies them. *IP addresses* have four parts (8 bits each) separated by periods (called *dots*). IP addresses range from 0.0.0.0 to 255.255.255.255.

IP address are organized into three classes. The Internet Network Information Center (NIC) assigns these based on the number of hosts the organization or company needs (the first 8-bit number tells you the class and size of the organization):

- Class A (0-127) is used for the largest organizations—usually only Internet service providers and very large companies. You won't find many addresses in this range. This class allows over 16.5 million hosts.
- Class B (128-191) is for large companies and organizations and can have just over 65,000 hosts.
- Class C (192-223) is for smaller organizations that can have up to 254 hosts.

Note: There are classes D and E, but they have special uses not relevant to this discussion.

Because most people have a hard time remembering numbers, the Internet allows hostnames as a type of alias for the IP address. For example, the IP address 192.82.208.8 has the hostname www.sgi.com.

As described in the previous section, hostnames consist of a hostname, a domain name, and a domain identifier. The hostname www.sgi.com is a host named *www* for a domain called *sgi* in the domain group for companies (*.com*). The domain name and identifier are usually grouped together and called simply the domain.

Conventions Used in This Guide

These type conventions and symbols are used in this guide:

Bold—Literal command-line arguments (options/flags).

Italics—Executable names, filenames, URLs, glossary entries (online, these show up as underlined), IRIX commands, manual/book titles, onscreen button names, program variables, and variables to be supplied by the user in examples.

`Fixed-width type`—Error messages, prompts, and onscreen text.

`Bold fixed-width type`—User input, including keyboard keys (printing and nonprinting); literals supplied by the user in examples.

ALL CAPS—Environment variables, defined constants, boolean operators.

““ (Double quotation marks) —Onscreen menu items and references in text to document section titles.

() (Parentheses)—Following IRIX commands surround reference page (man page) section number.

[] (Brackets)—Surrounding optional syntax statement arguments.

<>—(Angle brackets) Surrounding nonprinting keyboard keys, for example, <Esc>, <Ctrl-D>.

#—IRIX shell prompt for the superuser (root).

%—IRIX shell prompt for users other than superuser

Warning:

Warnings mark important information. Make sure you read the information before continuing with a task.

Chapter 1

Installing the Server

This chapter tells you how to begin configuring the Netscape Commerce and Communications Servers for your needs.

Before You Install

Before you install the either the Netscape Commerce Server or the Netscape Communications Server, make sure you perform the tasks listed below. This makes the installation process much smoother.

- Make sure the Domain Name Service (DNS) is up and running.
- Create a *home page*.
- Create an alias for the server.
- Create an IRIX user account.
- Choose unique port numbers.

Make Sure DNS Is Up and Running

When you install the Netscape Commerce and Communications Servers, 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 host.subdomain.domain, which is translated into a dotted IP address by a Domain Name Service (DNS). For example, www.sgi.com is the host "www" in the subdomain" sgi" and domain "com".
- Internet Protocol (IP) address is a set of numbers, separated by dots, that specifies the actual location of a host 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 server can't resolve hostnames and can't connect to any remote hosts.

Create a Home Page

A home page is usually the first document users see when they access your server. The content should introduce your server, company, or organization, or even yourself. It's used as an index to the information your server provides.

If you already have a home page, you can specify it during the installation process. If you don't have a home page, you can either create one before installation, or you can use the one the installation process creates. (You can edit it after installing the server.) To create a home page, you need a text editor or an application that generates HTML documents (these are ASCII text files). Hypertext Markup Language (HTML) documents contain text and flags that a Web browser uses to display the page. The optional WebMagic Author™ from Silicon Graphics, Inc. is a WYSIWYG editing tool that enables you to create HTML files without requiring you to know HTML syntax.

If you are unfamiliar with HTML, and need to learn more, you can use the Netscape Navigator to view the HTML source code for a page.Start the navigator, go to any page on the Web, and then choose "View/Source" from the menu. You can read Appendix A to start learning HTML. You can also read other books about HTML, or you can go to *http://home/netscape.com/home/how-to-create-web-services.html* to get the information online.

Create an Alias for the Server

If your server runs on one host among many in a network, you should set up a DNS CNAME record or an alias (such as www) that points to the actual server host. Later, should the need arise, you can change the actual hostname or IP address of the server host without having to change all of your URLs.

For example, you might call the server myserver.anycompany.com and then use an alias like www.anycompany.com. The URLs to documents on your server would then use the www alias instead of myserver.

Create an IRIX User Account

You should create an IRIX user account for the server. You probably want the server to have restricted access to your system resources and run under a nonprivileged system user account.

When the server starts and runs, it runs with this IRIX user account (you'll specify this account during installation). Any child processes of the server are created with this account as the owner. The account requires read permissions for the configuration files and write permissions for the log file directory. For security reasons, the user account should not have write permissions to the configuration files. This means that in the unlikely event that someone compromises the security of the server, they can't write to the configuration files.

You can use the account with the name `nobody`, but this might not work on some systems. Some hosts have 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 make sure it is greater than zero (the default UID for user nobody is 60001).

Caution: We strongly recommended that you use a dedicated account for the server.

The administration manager can also run with a user account that has write permission on the configuration files for all installed servers. However, it is much easier to run the administration manager as root because then it can start and stop servers with port numbers less than 1024. You must make sure you *always* shut down the administration manager when you're done with it.

Choose Unique Port Numbers

You need two port numbers: one for the administration server and one for the Commerce or Communications server. The administration server is a separate daemon that lets you manage multiple servers from a single forms-based interface.

Port numbers for all network-accessible services are maintained in the file */etc/services*. The standard HTTP port number is 80, but you can install the server to any port.

Warning: If you change the default HTTP port number from 80, you must edit the /etc/init.d/netsite file (/etc/init.d/netsite_commerce for the Commerce Server) to reflect the change.

You should choose a random number for the administration server to make it more difficult for anyone to breach your server security. When you access the administration forms, use the administration server's port number.

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

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

Replacing an Existing Server

If you're running a 1.0 server, the install program upgrades your 1.0 server and puts it and the new 1.1 server in a new server root directory (you specify this directory, but we recommend you use the default of */usr/ns-home*).

Warning: If you change the default server root directory from /usr/ns-home, you must edit the /etc/init.d/netsite file (/etc/init.d/netsite_commerce for the Commerce Server) to reflect the change.

Stop running the 1.0 server while you do the upgrade. After installation, check that the upgraded server runs correctly, and then delete the old 1.0 directory structure. This lets you bind the upgraded server to the same port as the 1.0 server.

If you have existing 1.1 servers, install the new server to the same server root. This lets you use one administration manager to manage all servers.

If you're upgrading a secure Commerce server, you must manually start the upgraded server because you have to specify a password when starting it.

Note: If you aren't upgrading a server, the new server must be installed in an empty directory. We recommend the */usr/ns-home* directory.

Warning: If you change the default server root directory from /usr/ns-home, you must edit the /etc/init.d/netsite file (/etc/init.d/netsite_commerce for the Commerce Server) to reflect the change.

What the Installation Process Does

After you fill out all of the install forms and click the link called "Go For It," 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. The following directories and files are created under the server root directory:

- *start-admin* and *stop-admin* are scripts that start the server manager. The server manager lets you configure all servers installed in the server root directory.
- *admserv/* contains the server administration files, the user name and password for the administration account, and the binary files for running the server manager.
- *bin/* contains the binary files for the server, such as the actual server, the administration forms, and so on.
- `[`*type*`]-[`*port#*`]/` are the directories for each server you have installed on the host. The directory name uses the server type (either *httpd* or *https*)

and the server's port number, for example httpd-80. Each server directory has the following subdirectories and files:

  – *config/* contains the servers configuration files: *magnus.conf*, *obj.conf*, and *mime.types*. It also contains the server key and other encryption certificates.
  – *logs/* contains any error and access log files.
  – There are shell scripts to start, stop, and restart the server and a script to rotate log files.

- *userdb/* contains all user databases. This central directory lets any number of servers use the same user databases.
- *mc-icons/* contains icons for FTP listings and Gopher menus.
- *extras/* contains utilities such as a log file analyzer.
- *nsapi/* contains header files and example code for created your own functions using the Netscape *API*. See the *Netscape Servers Programmer's Guide* for more information.

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

Restarting the Server Automatically

Once installed, the Netscape server and its child processes run constantly, listening for and accepting requests. If your host 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 it is set to "on":

```
# chkconfig | grep netsite
```

2. If you see:

```
    netsite off
```

   Enter the following command:

```
# chkconfig netsite on
```

   For the Commerce Server, enter:

```
# chkconfig netsite_commerce on
```

3. Repeat step 1 until you see:

   ```
   netsite on
   ```

   (or “netsite_commerce on”).

When the system is rebooted, the server starts automatically.

Note: If the server doesn’t restart after a reboot, make sure the */etc/init.d/netsite* file (or the */etc/init.d/netsite_commerce* file for the Commerce Server) reflects the proper directory location and port for the server.

Stopping the Automatic Server Restart

To keep the server from automatically restarting when the system is rebooted, enter:

```
# chkconfig netsite off
```

For the Commerce Server, enter:

```
# chkconfig netsite_commerce off
```

Starting the Server Manually

If you 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/netsite start
```

Stopping the Server Manually

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

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

Using the Server Manager

After you install the server files, the server should run without problems. However, you might need to change configuration information (for example, by adding security) or perform general maintenance on the server. All this is done with the Server Manager.

The Server Manager is a set of forms you use to change options and control your server. You can view the Server Manager immediately after installation (there is a link to it). You can use the Server Manager from any remote host—you don't need to use the host the server is installed on.

Starting the Administration Server

Note: The administration server uses its own configuration file.

Before you can do any server configuration, you must start the administration server.

1. Go to the server root directory and enter

   ```
   # start-admin
   ```

   This starts the administration server using the port number you specified during installation.

2. When you are done configuring your servers, stop the administration server by changing to the server root directory and entering

   ```
   # stop-admin
   ```

Starting the Server Manager

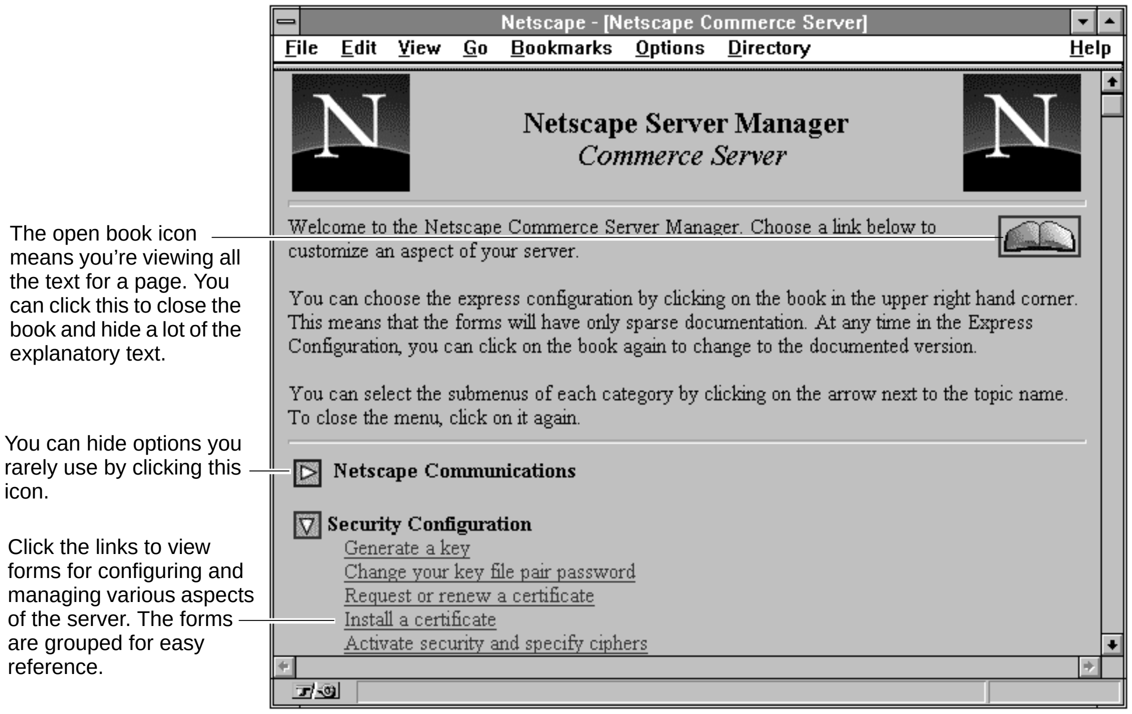
To view the Server Manager at any time, use a forms-capable browser to point to the URL:

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

Use the port number for the administration process that you specified during installation—don't use the port number for an individual server.

1. You'll be prompted for a user name and password. This is the administration username and password you specified during the installation process. The administration manager appears, listing all of the servers you have installed on the host.
2. Click the link for the server you want to configure. The Server Manager appears, as shown in Figure 1-1.

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


Netscape - [Netscape Commerce Server]
File Edit View Go Bookmarks Options Directory Help
The open book icon means you're viewing all the text for a page. You can click this to close the book and hide a lot of the explanatory text.
You can hide options you rarely use by clicking this icon.
Click the links to view forms for configuring and managing various aspects of the server. The forms are grouped for easy reference.

Netscape Server Manager
Commerce Server

Welcome to the Netscape Commerce Server Manager. Choose a link below to customize an aspect of your server.

You can choose the express configuration by clicking on the book in the upper right hand corner. This means that the forms will have only sparse documentation. At any time in the Express Configuration, you can click on the book again to change to the documented version.

You can select the submenus of each category by clicking on the arrow next to the topic name. To close the menu, click on it again.

Netscape Communications

Security Configuration
- Generate a key
- Change your key file pair password
- Request or renew a certificate
- Install a certificate
- Activate security and specify ciphers

Figure 1-1 The Server Manager—First Form

Configuring the Administration Server

You can configure the administration server to restrict access to certain people based on hostname or IP address. You can also change the port number that the administration server runs on (this is *not* the same port as any of the Commerce or Communications servers). In the Server Manager for any server (not the main administration page that lists the servers), click the link called “Configure administration server.” The following sections describe what you can configure.

Restricting Access

You will probably want to restrict which hosts are allowed to administer your server. This means only hosts that you specify can access the server manager and make changes to any of your installed servers.

You can enter a wildcard pattern of hosts to allow. You can restrict access by hostname or by IP address. Restricting by hostname is more flexible—if a host's IP address changes, you won't have to update your server. On the other hand, restricting by IP address is more reliable—if a DNS lookup fails for a connected client, hostname restriction cannot be used.

If you think you might have problems with DNS, you should use IP address restriction.

Change the Authentication User Name

You can also change the user name that the administration server uses to verify that you are the server administrator. The authentication password is the password you give the server after you enter the authentication user name.

Note: If you leave this password blank, the administration password remains unchanged—leaving it blank doesn’t remove the password.

Administrative Server Configuration

You must choose an IRIX user account to run the administration server. By using a separate server for administration, you can safely allow this administration server to run as the superuser because you run the administration server only when you need to configure a server (you don’t leave the server running constantly, as you do with other servers).

The port you choose must be specified in the URL you use to configure your server.

Troubleshooting Installation

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

- **I accidentally denied all access to the Server Manager forms.**

  Log in as root or with the server's user account. In the *admserv* directory, edit the *ns-admin.conf* file. There's a line for allowed hosts and a line for allowed addresses. Modify the lines to include your host and address, save the file, and then restart the administration manager.

- **Clients can't locate the server.**

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

- **The port is in use.**

  Most likely, you didn't shut down a server before you upgraded it. Shut down the old server, then manually start the upgraded one.

- **The server is slow and transfers take too long.**

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

Understanding Wildcard Patterns

In many parts of the server configuration, you specify wildcard patterns to represent one or more items to configure. For example, to restrict access, 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.

Table 1-1 lists wildcard patterns and their meaning.

Table 1-1 Wildcard Patterns

Pattern	Meaning
*	Matches zero or more characters.
?	Matches exactly one character, and it can be any character.
|	An OR expression. It matches either the substring `this` or the substring `that`. The substrings can contain other special characters such as `*` or `$.`
[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] (right bracket), 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.
$	Matches the end of the string.

Examples of Wildcard Usage

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

(quark|energy).sgi.com matches either quark.sgi.com or energy.sgi.com.

198.93.9[23].??? matches a numeric string starting with either 198.93.92 or 198.93.93 and ending with any 3 characters.

. matches any string with a period in it.

~sgi- matches any string except those 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|energy|neutrino).netscape.com matches any host from domain sgi.com except for hosts quark.sgi.com, energy.sgi.com and neutrino.sgi.com.

.com~.sgi.com matches any host from domain com, except for hosts from subdomain sgi.com.

Chapter 2

Security

The Netscape Commerce Server uses a protocol called Secure Sockets Layer (SSL) to provide advanced security features for secure data communications. This means the server can send and receive private information across the public Internet to SSL-enabled browsers without the data being compromised during transfer.

This chapter describes some of the concepts behind private and authenticated server communications. It discusses how to secure your server host and how to install security on the server.

Note: In contrast to the Commerce Server, the Netscape Communications Server does *not* support SSL or these advanced security features. If you need these features, contact Silicon Graphics, Inc. for information on migrating from the Communications to the Commerce Server.

Understanding Security

The Internet is a public network that interconnects millions of computers world wide. Few Internet hosts are directly connected to one another. For data to move from one computer to another, it almost always has to travel through several other connections. This is called routing. The routing of sensitive data over the Internet is problematic for two reasons:

- It's difficult to maintain privacy between two computers that aren't directly connected.
- Third parties can illegally pose as a computer in a conversation or transaction and intrude or eavesdrop on the information.

Figure 2-1 illustrates how data can be intercepted by a third party as the data travels through the Internet.

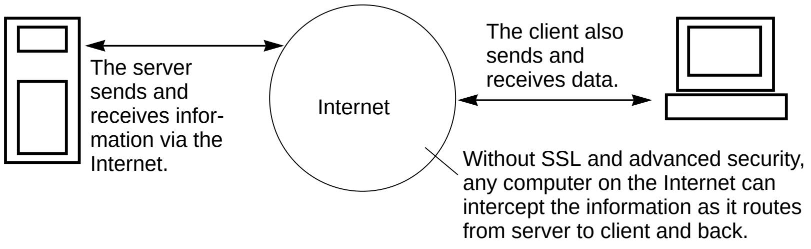

The server sends and receives infor-mation via the Internet.
Internet
The client also sends and receives data.
Without SSL and advanced security, any computer on the Internet can intercept the information as it routes from server to client and back.

Figure 2-1 Routing Between Hosts on the Internet

For the server to communicate with the client, various Internet computers must route the data. Without security, all data that passes between server and client can be intercepted by any computer along the data route. These computers can make copies of all the data, or they can pose as the client simply by answering messages sent to the client's network address.

How Security Works: SSL

The Netscape Commerce Server uses the Secure Sockets Layer (SSL) protocol for transferring data over the Internet. SSL is a cryptosystem that works at the protocol level. SSL provides the following:

- *Authentication* lets clients make sure they are communicating with the correct server. This prevents any computer from impersonating your server or attempting to appear secure when it isn't.
- *Encryption* scrambles the transferred data so that any eavesdroppers won't understand the information.
- *Data integrity* verifies that the data sent between client and server wasn't altered during transfer. That is, it can tell if anyone has added or removed data.

Note: Don't confuse SSL authentication with the server's access-control type of user authorization (see "Restricting Access" on page 10).

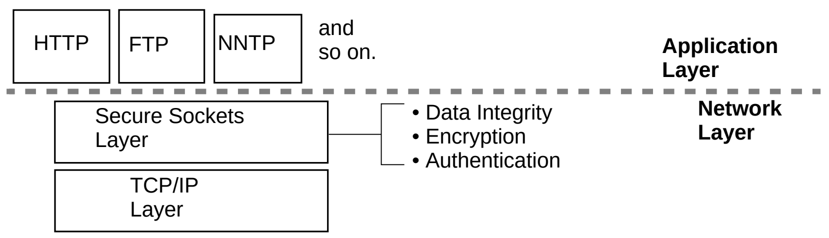


Figure 2-2 How SSL Relates to TCP/IP and Application Protocols

What Is a Certificate?

You need to acquire what is called a *digital certificate* for your server before the server can use SSL. Only companies called Certification Authorities (CAs) can issue certificates.

The signed digital certificate contains two groups of information. First is the certificate information itself, including the name of the server, its public key, the certificate's validity dates, and the name of the CA. The second piece is the digital signature. The digital signature cannot be forged. This entire message is digitally signed by a Certification Authority who is known to many servers and who can verify the relationship between a server and its public key.

To obtain a certificate, you generate a public-private keypair, safely store the private key, and then send the public key to a CA with proof of your identity. (See "Generating a Key Pair File" on page 24 for a definition of public and private keys.) The CA generates a digital signature for the server and sends back a signed digital certificate. The certificate can then be published in a directory or attached to any message being sent across the network. Any other user can then verify the authenticity of the certificate using the digital signature and the public key of the Certification Authority who signed the certificate. Once the certificate is known to be authentic, the information inside the certificate can be trusted.

Authentication

Before the server can begin a secure connection with a client, the client needs to ensure that it is connected to a secure server. To verify the identity of the server, the client and the Commerce server use authentication. After the server is authenticated, the client and server can encrypt data to each other and ensure the integrity of that data.

Warning: The client software must be SSL-enabled to do any secure transactions.

The authentication process is described as follows:

1. The client sends a request to connect to the secure server.

   Note: The server generates a private and public key before sending the request for a certificate.

2. The server sends a signed digital certificate to the client. The server uses the certificate it acquired from the CA.,

   The signed digital certificate contains two groups of information. First is the certificate information itself, including the name of the server, its public key, the certificate's validity dates, and the name of the CA. The second piece is the digital signature. The digital signature cannot be forged. It is encrypted using the CA's private key. See Figure 2-3 for a diagram of a signed digital certificate.

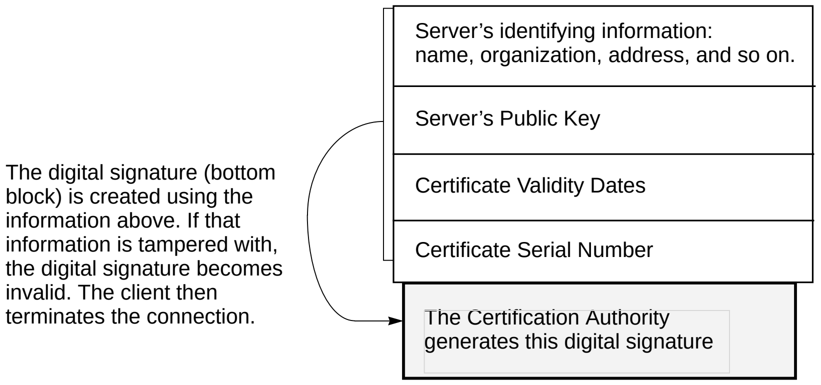

Server's identifying information: name, organization, address, and so on.
Server's Public Key
Certificate Validity Dates
Certificate Serial Number
The Certification Authority generates this digital signature
The digital signature (bottom block) is created using the information above. If that information is tampered with, the digital signature becomes invalid. The client then terminates the connection.

Figure 2-3 The Server Authentication Process and Signed Digital Certificates

3. The client authenticates the server by decrypting the digital signature and matching it with the certificate information. If the certificate was tampered with during the transaction, the digital signature won’t match. In this case, the client terminates the connection to the server. If the certificate is valid, the server is authenticated.
4. The client generates a session key and encrypts it using the server’s public key from the certificate (this way, only the server’s private key can decrypt it). The session key is later used to encrypt data and ensure data integrity. The client sends the encrypted session key to the server.
5. The server receives the session key, which it uses to encrypt and decrypt the data it can then securely send and receive from the client.

Encryption

The Netscape Commerce server delivers security by encrypting data sent between client and server. Encryption is the scrambling of information so that only someone with a specific key can decrypt the message.

After the client authenticates the server, the client generates a session key that lets the client and server encrypt and decrypt data, preventing a third party from deciphering their communications.

Figure 2-4 shows how the encrypted data can be intercepted by a third party, but the third party can’t read the data because it is encrypted. However, a third party can taint the data by removing or adding unknown data to the transaction. SSL uses data integrity to guard against this type of tampering.

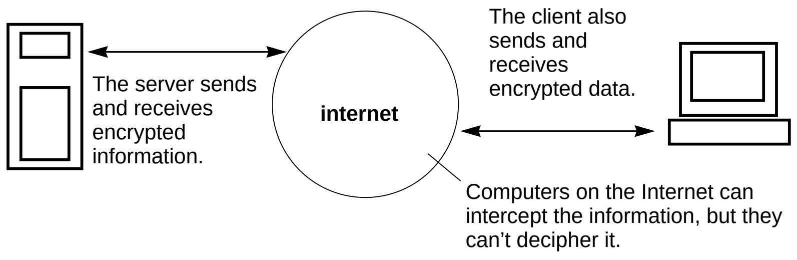

The server sends and receives encrypted information.
internet
The client also sends and receives encrypted data.
Computers on the Internet can intercept the information, but they can’t decipher it.

Figure 2-4 Encryption Scrambles Data to Thwart Intruders

Data Integrity

SSL uses Message Authentication Codes (MACs) to ensure the data transferred between client and server hasn’t been tampered. For example, someone can intercept the data being sent and add or remove bits. They can’t view the data because it’s encrypted.

If the integrity is compromised, the client or server terminates the connection.

The Certification Authority

Certification Authorities (CAs) are trusted third-party companies that can approve requests for signed digital certificates. Certificates are required if you want to use SSL and advanced security features with your server. The certificate is used to authenticate the server to client browsers before they begin a secure transaction.

Note that not everyone who requests a certificate is given one. Also, it can take anywhere from a day to two months or more to approve a certificate. You are responsible for promptly providing all the necessary information to the CA.

When you purchased the Netscape Commerce Server, you received a list of CAs. You must contact a CA to find out what information they require before they issue a certificate. Most CAs require that you prove your identity. For example, they want to verify your company name and who is authorized by the company to administer the Netscape Commerce Server and whether you have the legal right to use the information you provide.

When installing security, you provide the following to the CA:

- *Common Name* is usually the fully-qualified hostname used in DNS lookups, for example, www.sgi.com. However, some CAs might require different information, so it’s very important to contact them about this.
- *E-mail Address* is your business e-mail address. This is used for correspondence between you and the CA.

- *Organization* is the official, *legal* name of your company, educational institution, partnership, and so on. Most CAs require that you verify this information with legal documents (such as a copy of a business license).
- *Organizational Unit* is an optional field that describes an organization within your company. This can also be used to note a less formal company name (without the Inc., Corp., and so on).
- *Locality* is an optional field that usually describes the city, principality, or country for the organization.
- *State or Province* is usually required, but can be optional for some CAs. Most CAs won't accept abbreviations, but check with them to be sure.
- *Country* is a required, 2-character abbreviation of our country name (in ISO format). The country code for the United States is US.

After you contact a CA and gather the information you need, you can submit a request for a certificate (see the following section). While waiting for the CA to approve your request, you might want to work on securing the physical host your server is installed on (see "Securing the Server Host" on page 21).

Preparing to Install Server Security

Before you can enable security on your server, you need to choose a Certification Authority, request a certificate, and then install the certificate they send you. You'll also need to physically secure the host and the operating system that your Commerce server resides on.

Securing the Server Host

To have the most secure server, you need to secure the host that the server is installed on. This section gives some guidelines that will help you determine and fix some of the security risks.

Secure Private Keys

Make sure your private key is protected. This is the fundamental security risk for the server.

Warning: Store the key file in a directory that either only the root user has access to or that the server's user account has access to.

Keep the password confidential, and *never* write it down. Choose a password that is a mix of letters, numbers, and valid punctuation marks. It's also important to know if the file is stored on backup tapes or is otherwise available for someone to intercept.

Limit Availability

This is probably the simplest security measure and is often forgotten. The server host should be kept in a locked room that only authorized people have access to. This prevents anyone from hacking the server host itself.

Limit Applications

You should carefully consider all applications that run on the server host. Disable all unnecessary system daemons and services. For example, the sendmail daemon is difficult to configure securely and it can be programmed to run other possibly detrimental programs on the server host.

Carefully choose the processes started from startup scripts. Don't run *telnet* or *rlogin* from the server host. You also shouldn't have *rdist* on the server host (this can distribute files but it can also be used to update files on the server host).

Limit Ports

You should disable any ports not used on the host. Use routers or firewall configurations to prevent incoming connections to anything other than the absolute minimum set of ports. The secure server runs on port 443, so connections to any other port should fail. This means that the only way to get a shell on the host is to physically use the server's host, which should be in a restricted area already.

Limit Administration

You should never do remote server administration, especially in an unsecure network. Anyone could intercept your administration password and reconfigure the server. The administration server doesn't use the SSL security like the Commerce server.

Also, you should restrict access to the server administration forms by allowing only local hosts. See "Restricting Access" on page 10 for information on access control for the administration forms.

Change Passwords

Don't keep passwords for long periods and always use proper passwords. Your password should be long and include upper and lowercase characters, numbers, and special characters. You should never use words known in any language. A good password is one you'll remember but others won't guess. For example, `MBi12!mo` could be remembered as "My Baby is twelve months old!" A bad password would be your child's name or birthdate.

It is also very important to use different passwords for different applications. Your root password should be different from your server administration password and your keyfile password.

Know the Limits

The Netscape Commerce Server offers secure connections between the server and the client. It can't control the security of information once the client has it, nor can it control access to the host itself and its directories and files.

Being aware of these limitations helps you know what situations to avoid. For example, you might acquire credit card numbers over a secure connection, but are those numbers stored in a secure file on the server host? What happens to those numbers after the secure connection is terminated? You should be responsible for securing any information that clients send to you through a secure connection.

Installing Security

The procedure for installing security on your Netscape Commerce Server is straightforward:

1. You use a form to generate a private and public key pair.
2. You request a certificate from a Certification Authority.
3. When the certificate is transmitted back to you from the Certification Authority, you install it using another form.
4. Finally, you configure and activate security for the server.

Make sure you've read the information about Certification Authorities and have prepared all the information before you begin this process!

The following sections correspond to links on the Server Manager page.

Generating a Key Pair File

Your server needs to generate a key pair file that holds the public and private keys for your server. These keys are used during secure communications. The private key is stored in encrypted form using a password you specify.

- A public key is usually used to exchange session keys. It is also used to verify the authenticity of digital signatures. A public key can also be used to encrypt data.
- A private key is usually used to decrypt session keys that were encrypted using the matching public key. You always keep your private key private. The Commerce server keyfile password protects this key, but for additional security you shouldn't keep the key file in a directory where people have access to it. The private key is also used to create a digital signature when you first request a certificate.

To generate a key pair file,

1. Start the Server Manager and click the Security Configuration link called “Generate a key.”
2. In the form that appears, type a path (relative or absolute) where you want to store the key file. This directory should be safe from other users. For example, use a directory that only you have read and write access to.
3. Type a password for the key file. Make sure you memorize this password (and don’t write it down). The security of your server is only as good as the security of the key file and its password.

   Warning: If you must write down the password, make sure you store the written copy in a safe, safety box, or other physically secure place.

   Any time secure servers are restarted, this password is required to decrypt the key file and extract the public and private keys.

   The password must be eight characters in length. It is required that the password have at least one non-alphabetical character (a number or punctuation mark) somewhere in the middle.

   You shouldn’t administer security using a remote connection. Anyone on your network could potentially intercept your password.
4. Click the *Make These Changes* button. The server generates the key pair file and places it in the directory you specified.

You can return to the Server Manager or continue installing security by requesting a certificate.

Note: You should periodically change your key file password. Use the Security Configuration link called “Change your key file pair password” to change it. If you forget your password, you will have to regenerate your keypair file. This means you must also obtain another certificate (there are usually additional costs to do this).

Requesting a Certificate

Before you can request a certificate, you must choose a Certification Authority (CA) and contact them regarding the specific format of the information they require. See “The Certification Authority” on page 20 for more information.

To request a security certificate,

1. From the Server Manager, click the link called “Request or renew a certificate.”
2. In the form that appears, type the email address for the CA you have chosen.
3. Specify if this is a new certificate or a renewal. Certificates are generally good for 6 months to a year. Some CAs automatically send you a renewal.
4. Type the location and password for your key file.
5. Type the information for your distinguished name. The format of this information varies by CA. For a general description of these fields, see “The Certification Authority” on page 20. This usually isn’t required for a renewal.
6. Type your phone number. Be sure to include your area code and any international codes as applicable. The CA uses this to contact you regarding your request for a certificate.
7. Double-check your work to ensure accuracy. The more accurate the information, the faster your certificate is likely to be approved. Click the *Make These Changes* button when the information is correct.

The server composes an e-mail message to the CA that includes your information. The e-mail message has a digital signature created with your private key. The digital signature is used by the CA to verify that the email wasn’t tampered with during routing from your server host to the CA. In the rare event that the email is tampered with, the CA will usually contact you by phone.

You can’t continue installing security until your request for a certificate is approved and a confirmation is sent to you via email. During this time, you should read about securing the server host. See “Securing the Server Host” on page 21 for information.

Installing the Certificate

When you get an email from the CA that contains your certificate, you need to decode the certificate from the email. You can either save the email somewhere accessible to the server or copy the text of the mail and be ready to paste the text in the form. The information in the email is encrypted, but you should take every protection to prevent unauthorized access to the certificate file.

To install the certificate,

1. From the Server Manager, click the link called "Install a certificate."
2. Either type the full pathname to the saved email, or paste the email text in the space provided.
3. Specify a destination directory for the certificate. You can enter the name as a relative path or as an absolute path. The default name and location is `[`*ServerRoot*`]/https-443/ServerCert.der`.

   Remember where you put this file! It should not appear in your document root directory or any generally available directory.
4. Click the *Make These Changes* button. The server extracts the certificate from the email and saves it to the directory you specified.

You can return to the Server Manager or continue by activating security and specifying ciphers.

Activating Security and Specifying Ciphers

Most of the time, you want your server to run with security enabled. You might, at other times, want to disable security. If you temporarily disable security, make sure you enable security before processing transactions that require encryption, authentication, or data integrity. You should also make sure that any private files are removed from public access.

Ciphers define how a document is encrypted. Generally, the more bits used during encryption, the harder it is to decrypt the data.

To activate security and specify ciphers,

1. From the Server Manager, click the link called “Activate security and specify ciphers.”
2. Check the radio button to enable security. You can always return to this form and disable security.
3. Specify a new port number. The standard HTTP port is 80, and the standard *secure* HTTP port is 443. You should always use 443 with the secure server. You can use other port numbers, but it isn’t recommended. (If you do change the port number, you must edit */etc/init.d/netsite_commerce* to reflect the change.)
4. Type the path to the key file. This is used when the server needs to encrypt or decrypt messages and digital signatures.
5. Type the path for the certificate file. The server uses the certificate to let client browsers verify the servers identity (server authentication).
6. Activate any ciphers by checking them. These ciphers are used when the client and server try to agree on the encryption method they should use.

   You should check all the ciphers and let the browser and server decide the method. However, you might want to uncheck some ciphers to exclude some transactions. For example, some ciphers are only used for data transfer outside of the United States. If you uncheck these, then you’re basically restricting access to browsers that use the checked ciphers (for example, US versions of Netscape Navigator).

   The ciphers are as follows:

   - RC4 cipher with 128-bit encryption
   - RC4 cipher with 40-bit encryption
   - RC2 cipher with 128-bit encryption
   - RC2 cipher with 40-bit encryption
   - IDEA cipher with 128-bit encryption
   - DES encryption, 64-bits
   - DES encryption with EDE 3, 192 bits
7. Click the *Make These Changes* button. You’ll need to restart the server before using any secure features.

Effects of a Secure Server

Once security is enabled, you need to keep a few things in mind for your Netscape Commerce Server to be secure. This section describes what you need to know about running a secure server versus an unsecure server.

Secure URL Construction

Secure URLs are constructed using *https* instead of simply *http*. URLs that point to documents on a secure Commerce Server have the format:

`https://`*host*`.`*domain*`.`*dom*`/`*pathname*`/`*document*

Secure Server Document Root and Logging

Once security is installed and enabled on a Netscape Commerce Server, all communications between the server and SSL-enabled browsers (such as Netscape Navigator) are private and authenticated. This means that any document sent to a user with an SSL-enabled browser is *automatically encrypted*. There is no way around this.

Note: Browsers not enabled with SSL won't work with a *secure* Commerce Server because they can't decrypt the data. (They will work if the Commerce Server doesn't enable SSL security.)

The Secure Log

Once security is enabled, a new log, called "secure," is created in the normal log directory. Entries in the log look like this:

```
198.93.92.99: [02/Nov/1994:23:51:46 -0800] using keysize 40
```

The IP address is first, followed by the date and time of access, and then the key size. The key size represents a level of security. Generally, the bigger the key size, the higher the level of security. See "Activating Security and Specifying Ciphers" on page 27 for a list of supported key sizes.

Unprotected Server Document Root

If you want to send documents from a document root that does not have security features (such as the Communications Server), it's highly recommended that you operate the unsecure server on a different host from the secure Commerce Server. If your resources are limited and you must run an unprotected server on the same host as your Netscape Commerce Server, follow these guidelines.

- Port Number Assignments—make sure that the secure server and the unprotected server are assigned different port numbers. For example, use 443 for the secure server and 80 for the unsecure one.
- Use CHROOT on the document root directory—the unprotected server should have references to its document root redirected using the *chroot* command.

Changes to the magnus.conf File

With a secure server installed, you should know about the following changes to the *magnus.conf* file (the server's main configuration file). These new directives are briefly described below:

Security

The Security directive tells the server whether security is enabled or disabled.

SYNTAX

`Security` *value*

value specifies if security is on or off. `Security on` enables security; `Security off` disables security.

ServerKey

The ServerKey directive tells the server where the key file is located.

SYNTAX

`ServerKey` *keyfile*

keyfile is the server's key file, specified as a relative path from the server root or as an absolute path.

ServerCert

The ServerCert directive specifies where the certificate file is located.

SYNTAX

`ServerCert` *certfile*

certfile is the server's certificate file, specified as a relative path from the server root or as an absolute path.

Ciphers

The Ciphers directive specifies the ciphers enabled for your server.

SYNTAX

```
Ciphers +rc4 +rc4export -rc2 -rc2export +idea +des +desede3
```

A + means the cipher is active, and a - means the cipher is inactive.

Additional Reading

The following resources can help you begin learning and understanding issues related to general security:

- *http://home.netscape.com/info/security-doc.html*
- *http://www.rsa.com/*
- *http://www.cis.ohio-state.edu:80/hypertext/faq/usenet/security-faq/faq.html*
- *snews://secnews.netscape.com/netscape.security*
- *http://home.netscape.com/commun/netscape_user_groups.html*
- *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. Bruce Schneier. John Wiley & Sons, Inc., 1994.

- *IRIX Advanced Site and Server Administrator's Guide*

Usenet newsgroups that regularly discuss computer security include comp.security.misc, comp.security.unix, and alt.security.

Chapter 3

Configuring the Server

This chapter describes how to configure the Netscape Commerce and Communications Servers by using the Server Manager configuration forms.

Using the Server Manager

After you install your server, you'll need to do periodic maintenance. This includes making infrequent changes to configurations, such as changing the server's name and port number, to daily tasks such as adding, changing, and removing users in user database files.

Use the Administration Manager to jump to the Server Manager for the server you want to configure. The Administration Manager lists all the servers installed on your host. It lists them according to type (Communications or Commerce) and by the port number they are listening to.

To jump to the Server Manager, click the link in the Administration Manager for the server you want to configure. You can return to the Administration Manager by clicking the link at the bottom of the Server Manager page.

Once in the Server Manager, you click links to configure parts of the server. Most links go to forms that configure the entire server. Some links go to forms that can configure files or directories the server maintains. These forms have four buttons at the top that let you specify what resource to configure, as described in Figure 3-1.

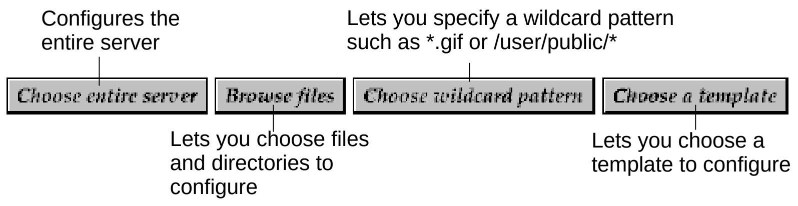

Configures the entire server
Lets you specify a wildcard pattern such as *.gif or /user/public/*
Choose entire server
Browse files
Choose wildcard pattern
Choose a template
Lets you choose files and directories to configure
Lets you choose a template to configure

Figure 3-1 Option Buttons on Server Configuration Forms

The rest of the chapter is organized by main topics on the Server Manager page.

Server Control

You can configure the server's basic functions with the forms under Server Control in the Server Manager.

System Specifics

You can configure the server's technical options, including its location, the user account it uses, and the processes the server spawns.

Changing the Server's Location

For various reasons, you might need to move the server from one directory to another. To do this, you change the location that the server references (it needs to know where the binary files are). Then you shut down the server and copy the server files and subdirectories to a new location.

Warning: If you change the default server root directory from /usr/ns-home, you must edit the /etc/init.d/netsite file (/etc/init.d/netsite_commerce for the Commerce Server) to reflect the change.

Changing the Server’s User Account

The Server User specifies an IRIX user account that the server uses (all the server’s processes are assigned to this user account).

You don’t need to specify a server user if you chose a port number greater than 1024 and aren’t running as the root user (in this case, you don’t need to be logged as root to start the server). If you don’t specify a user account, the server runs with the user account you start it with, so make sure that when you start the 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 Server User should already exist as a normal IRIX user account. After 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.

Server Processes

Whenever people use the server, the server uses background processes to service their requests. You can specify the minimum number of processes dedicated to the server. These processes are spawned when the server starts and they remain idle until needed. The maximum number sets a limit on the number of processes your server can use. The actual number of processes fluctuates between the minimum and maximum numbers.

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

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), 10-20 processes should be sufficient.

Note: 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 service. All processes must fit in the main memory simultaneously to make the server efficient.

Process Lifespan

Process lifespan specifies the number of requests that each of the child processes serves before the processes exit and are respawned (this is set to 32 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 server ensures that memory isn't wasted by "lost" processes. For example, on rare occasions, there might be resource "leaks" in operating system libraries or in the server itself (though none are currently known). By specifying a lifespan, a process answers a certain number of requests before exiting and respawning.

Domain Name Service

The server can be configured to never use Domain Name Service (DNS) lookups during normal operation. Even though DNS lookups are a useful tool, they can be expensive in terms of performance.

Warning: **Be aware of the consequences of turning off DNS on your server— hostname restrictions won't work, and hostnames won't appear in your log files.**

Stopping and Restarting the Server

You can stop, start, or restart the server from the Server Manager.

Restart performs a soft restart. This means the server reloads its configuration files and service isn't interrupted. Sometimes you'll need to do a hard stop and start, such as if you change the port number, enable security, and so on. (If you aren't sure if you need to do a hard start or soft restart, do a hard stop and restart.)

Stop shuts down the server completely, interrupting service until it is restarted.

Rotating Log Files

The server uses several log files. It logs accesses to the server and any errors that occur. There are times when you'll want to archive the log files and have the server create new log files.

When you rotate log files, the server renames the existing log files and then creates new log files with the original names. This lets you back up or archive (or simply delete) the old logs. The old log file is saved with the name of the file combined with the date the file was rotated. For example, *access* becomes `access.24-Apr`.

URL Configuration

URL configuration lets you change elements that effect the URLs to documents on your server. You can change the server name and port number and the directories to your documents, and you can map URLs for one directory to another. The general structure for URLs is

`http://`*hostname.domain.dom*`[`*:portnum*`]/`*directory*`/`*filename*

You can also map URLs to another server or directory. This is useful when you move files and directories among filesystems.

Global URL Configuration

This section configures the structure of all the URLs for your server.

Server Name

The server name is the full hostname of your server host. When clients access your server, 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 server with the name `www.sgi.com`.

If your system administrator has set up a DNS alias for your server, then you should use that alias here. If not, you should use the host's name combined with your domain to construct the full hostname.

Server Port Number

Server Port Number specifies the TCP port the server listens to. The port number you choose can affect your users—if you use a nonstandard port, then anyone accessing your server must specify a server name and port number in the URL. For example, if you use port 8080:

```
http://www.sgi.com:8080
```

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

Port numbers for all network-accessible services are maintained in the file */etc/services*. The standard HTTP port number is 80. The standard secure HTTPS port is 443.

Technically, the 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 server, you'll have to use a number above 1024.

Server Address

At times you'll want the server host to answer to two URLs. For example, you might want to answer both *http://www.a.com/* and *http://www.b.com/* from one host.

Because of limitations in the HTTP protocol, this is difficult to configure. However, there is a trick to do this that involves making your host answer to more than one IP address. This works only on certain systems.

If you have already set up your system to listen to multiple IP addresses and want to use this feature, you must tell the server which IP address it belongs to.

Document Configuration

This section configures the how the server deals with URLs.

Document Root

For a public server, you probably don't want to make all the files on your filesystem available to remote clients. The easiest way make sure this does not happen is to keep all of your server's documents in a central location, known as a document root.

Another benefit of the document root is that you can move your documents to a new directory (perhaps on a different disk) without changing any of your URLs because the paths specified in the URL are relative to the document root directory.

For example, if your document root is */usr/htmldocs*, a request such as *http://www.acme.com/products/info.html* tells the server to look for the file in */usr/htmldocs/products/info.html*. If you change the document root (that is, you move all the files and subdirectories), you only have to change the document root directory that the server uses instead of mapping all URLs to the new directory or somehow telling all your users to look in the new directory.

You can choose not to use a document root, but it isn't recommended because the server assumes / as the document root, which means users have access to all files on your server.

Directory Indexing

In your document root directory, you'll probably have several subdirectories. For example, you might create a directory called *products*, another called *people*, and so on. It's often helpful to let clients access an overview (index) of these directories.

There are two ways the server can do this:

- It first searches the directory for an index file called `index.html`, which is a file you create and maintain as an overview of the directory's contents. You can specify any file as an index file for a directory. This means you can also use CGI programs to configure pages.
- If an index file isn't found, the server generates an index file for you that lists all the files in the document root. The generated index has two formats:
  - Fancy directory indexing is fairly detailed. It includes a graphic that represents the type of file, the date the file was last modified, the file size, and a description.
  - Simple directory indexing is less detailed but takes less time to generate.

Server Home Page

When users first access your server, they usually use a URL such as *http://www.acme.com/*. When the server receives a request for this document, it returns a special document called a home page. Usually this file has general information about your server and links to other documents.

You can either specify a file in the document root as the home page, or the server will create an index file instead.

Default MIME Type

When a document is sent to a client, the server includes a section that identifies the document's type, so the client knows what to do with the document. However, sometimes the server can't determine the proper type for the document. In those cases, a default value is sent.

The default is usually "text/plain," but it should reflect the most common type of file stored in your server. Some common types are:

- text/richtext
- text/plain
- text/html
- image/tiff

- image/jpeg
- image/gif
- application/x-tar
- application/postscript
- application/x-gzip
- audio/basic
- audio/x-wav

URL Mappings

URL mappings let you map URLs to another server or directory. You specify a URL prefix to map and where to map it. When a client accesses the server with a mapped URL, the server gets the requests from the mapped server or directory.

Map a URL to a Local Directory

URL mappings are used to point to documents in directories outside of the document root directory.

Most of the time, you keep all of your documents in the document root. Sometimes, though, you want to refer to a directory outside of your document root. You can do this through directory mapping.

First, you choose the URL prefix to map. This is the URL users send to the server when they want documents in the mapped directory (this is seamless to the user). For example, a mapped URL could be *http://www.acme.com/products/index.html* where *products/* is the prefix you specify.

Next you specify the directory to map those URLs to. For example, the directory could be called */sales/tools/products*. It should be a full system path.

Finally, you might want to use a template to specify how this directory should be configured. You can choose an existing template or choose *cgi* to specify that all files in this directory are CGI programs. For more information on templates, see “Creating a Template” on page 53.

Map a URL to Another Server

Redirection is a method for the server to tell a user that a URL has changed (for example, if you move files to another directory or server). You can also use redirection to seamlessly send a person from your server to another.

To map a URL to another server, you must first specify the URL prefix you want the server to redirect. For example, if the URL you want to map is *http://www.sgi.com/info/movies*, you'd type */info/movies* as the prefix to redirect.

Choose which URL you want to redirect them to. You have two choices:

- Specify a single URL. You must specify a complete URL (hostname, directory, and filename). For example, type *http://w3.acme.com/new-files/info/movies*.
- Specify a URL prefix. Use this if the directory on the new server is the same as in the mapped URL. For example, you type only the new server name *http://w3.acme.com/*.

View, Edit, or Remove URL Mappings

You can view all URL mappings on a server, and then use links to edit and remove any URL mappings.

Each URL mapping is listed with a pair of links next to it for editing and removing the URL mapping. The "Edit" link takes you to the same page you used to add the mappings, except you can change the values in the form. The "Remove" link immediately and permanently removes the mapping from your server.

Customize Users' Public Information Directories

You can configure public information directories that let all the users on your host create home pages and other documents without your intervention.

Note: Another way to do this is to create a URL mapping to a central directory that all of your users can modify.

With this system, clients can access your server with a certain URL that the server recognizes as a public information directory. For example, suppose you choose the prefix ~ and the directory *public_html*. If a request comes in for *http://www.sgi.com/~bob/aboutbob.html*, the server recognizes that *~bob* refers to a users' public information directory. It looks up *bob* in the system's user database and find Bob's home directory. The server then looks for the request (*aboutbob.html*) in Bob's home directory.

To configure your server to use public directories, you need to choose a user URL prefix. The usual prefix used is ~ because it's a character that is rarely used. Next, you need to choose the subdirectory where the server looks for user home directories. A typical directory is *public_html*.

The server needs to know where to look for a file that lists users on your system. The server uses this file to determine valid user names and to find their home directories. You can use the system password file for this, which means the server uses standard library calls to look up users. Or, you can create another user file to use to look up users. You can specify that user file with an absolute link.

Each line in the file should have this structure (the elements in the */etc/passwd* file that aren't needed are indicated with *):

```
username:*:*:groupid:*:homedir:*
```

You also have the option of loading the entire password file on startup. If you choose this option, the server loads the password file into memory when it starts, making user lookups much faster. On the other hand, if you have a very large password file, this can use too much memory.

Finally, you can choose a configuration template that the server uses so that you can control what is allowed from public information directories. This can prevent users from creating symbolic links to information you don't want made public, and so on. See "Creating a Template" on page 53 for more information.

User Databases

User databases are lists of users who can access the server. Each user has a username and password. User databases control who has access to documents through the server.

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

Creating and Removing a User Database

To create a user database for your server:

1. Click the Server Manager link to create a user database.
2. Type a name for the database. Don't type a path, because all databases are stored in */userdb*. The database name can be up to 256 characters, but must not include any spaces.
3. Check the type of database you want to create. A DBM file stores the passwords unencrypted, so each line has the format `user:password`. An NCSA-style database encrypts the passwords.
4. Type a password for this database. The password can be up to 8 characters. Retype the password to ensure accuracy. When you click the *Make These Changes* button, the server creates the database, then lets you jump to the page where you can add users to the new database.

To remove a user database:

1. Click the Server Manager link to remove a user database.
2. In the form that appears, choose the database you want to remove.
3. Type the password for the database. You can't remove the file unless you have the password.
4. Click the *Make These Changes* button. The user database is permanently removed from the server.

Adding, Editing, and Removing Users in a Database

To add a user to a database:

1. Choose the database and type the password for the database.
2. Type a user name. This is the name the user types when authenticating with the server. It can be up to 254 characters.
3. Type a password for the user. It can be up to 8 characters. Type it twice to ensure accuracy. The user types this password when authenticating with the server.
4. Click the *Make These Changes* button. The user name and password are added to the database. You can continue adding names to the database, or you can return to the Server Manager.

If you want to change a user's name, you need to remove the user, then add them with the new name. You can change a user's password. To change user passwords in a database:

1. Click the link to edit users.
2. Choose the database that contains the user whose password you want to edit.
3. Type the password for the database file.
4. Type the user name you want to edit.
5. Type the new password.
6. Click the *Make These Changes* button. You can continue removing users or return to the Server Manager.

To remove users from a database:

1. Click the link to remove users.
2. Choose the database that contains the user name that you want to remove.
3. Type the password for the database file.
4. Type the user name you want to delete.
5. Click the *Make These Changes* button. You can continue removing users or return to the Server Manager.

Converting an NCSA or Text File to a User Database

The server stores its databases in the server root, in the directory */userdb*. The NCSA-style or text file can reside anywhere for the conversion, but the converted file is stored in */userdb*.

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

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

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 */userdb*.

Sometimes, a user may be entered twice into a database. You can choose how the convertor behaves under this circumstance: either it can overwrite the existing user, or it can leave the user unchanged and keep track of those users.

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 *Make These Changes* button. The server stores the new password.

Removing an Existing Database

You can remove a database from the */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 *Make These Changes* button. The server deletes the file from the filesystem.

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.

Restricting Through User Authorization

You can restrict access to your entire server or a particular section of it (files or directories) by using HTTP user authorization. You need a user database that provides the server with a list of users. Be sure to choose the section of the server you want to restrict! See Figure 3-1 for directions on choosing a resource.

When a user accesses a port of the server that uses user authorization, the client application asks the user for a name and a password before continuing. After the user enters this information, the server checks it, and then either allows or denies the user. If you have the Commerce server configured to use SSL, the user name and password are send encrypted.

To set up user authorization, choose the database that the server uses to look up user names and passwords. Next, type a wildcard pattern to tell the server which users from the database are allowed access. For example, if your database contained Bob, Fred, Mary, and Joe, but you only wanted Bob and Mary to have access to this section, you could use a wildcard pattern of (Bob|Mary). If you leave this entry blank, all users from the database are allowed access.

Finally, you create a realm that describes the part of the server on which you're using access control. This is a text string that helps the user know what part of the server they are trying to access. For example, if you were restricting access to a directory of product schedules, you might name the realm "Confidential product schedules."

Restricting by Hostname and IP Address

You can restrict access to pages on your server by hostname or IP address. You limit access to only the sites that you want to have access to various parts of your server.

Address restriction is an easy way to get better control over who is seeing your documents. When a request comes in for a document, the server knows the IP address that the request is coming from. Once it has this address, it uses DNS to look up the hostname that corresponds to that IP address. Then, it checks its address restriction.

The address restriction is done in two steps: first, the server tries to match the incoming hostname with the restriction hostname. If the client passes, the document is served. If the client fails the test, the server then checks its IP address against the restriction IP addresses. If it passes, the document is served. If it fails, then the server takes appropriate action.

What to Protect

First, you should choose which files or directories inside this resource you want to protect. This is simply a way to be more specific about what the server should apply restriction to.

For example, if you choose the directory */usr/html-docs/info/** as the resource you want to edit, you could specify:

- No additional specification—protects everything in this resource with hostname restrictions.
- A wildcard pattern matching only certain files—for example, **.gif*.

Note that by using this specification, you can protect many different things in the same directory with different address restrictions for all of them.

Note: To deny all hosts (for example, to hide your C source files from everyone), type a hostname restriction of **`*~*`**.

Enter a wildcard pattern of hosts to allow. You have the choice of restricting access by hostname or by IP address. Restricting by hostname is more flexible—if a host's IP address changes, you won't have to update your server. But on the other hand, restricting by IP address is more reliable—if a DNS lookup fails for a connected client, hostname restriction cannot be used.

Remember that the hostname and IP addresses should be specified with either a wildcard pattern, or a comma separated list, but not both. Also be sure your wildcard pattern is not recursive—there should only be one level of parentheses in the expression.

What Happens When a Client is Denied?

Finally, you should specify what should happen when a client is denied access. Normally, the server sends Not Found, which is the same thing that happens when a client requests a document that does not exist. However, sometimes you want to let the user know what they are missing. In those cases, you can specify a specific file that the server sends back. The filename should be an absolute path.

Restricting System Links

You can limit the use of filesystem links in your server. Filesystem links are references to files stored in other directories or filesystems. The reference makes the remote file as accessible as if it were in the current directory.

Because filesystem links are an easy way to create pointers to documents outside of the normal document root and because anyone can create these links, you might be concerned about what people might create pointers to (for example, confidential documents or system password files).

Hard links and soft links are simply two different methods of achieving this goal. You can choose to disable both of these methods, or just one.

You should choose whether or not to allow links in the resource you have selected, and what level of restriction you want.

Finally, you should choose from which directory the server should start looking for filesystem links. You can give one of two directory types here:

- A full pathname—In this case, the server treats the path you give as a prefix, and when it recognizes that prefix in a request, any directories following the prefix are checked for filesystem links.

- A partial pathname—In this case, the server looks for the partial pathname you give here as a substring of the incoming request; that is, if you give the directory *nolinks* for the "from" directory, then the server looks for a directory named *nolinks* in the incoming request If it finds the *nolinks* directory, it checks all following directories for filesystem links.

Dynamic Configuration

Web server content is seldom managed entirely by one person. Many times, different parts of a web server are written by different people. For example, each employee maintains a home page.

When these users need to configure something, it is unrealistic for you as the administrator to allow all of them access to the Netscape Server Manager. In these cases, it is useful to allow them a subset of configuration options so they can control only what they need to.

Note: If you haven't already, configure an area of your server to use these configuration files.

Normally, your server gets its configuration from two or three files that are kept in the server root and modified with the Server Manager.

With this feature, you can give users the ability to control more about their home pages in their public information directories. You can allow them to apply access control or customize error messages without allowing them to use CGI or parsed HTML. The format and capability of these dynamic configuration files is described in the following section.

When a request is made for a resource in which dynamic configuration is enabled, the server must search for the configuration files within one or more directories of that resource. This search can be an expensive operation in terms of performance, so the server lets you configure how much flexibility you need, weighing it against the efficiency cost.

You provide a base directory to the server. The server starts its search for configuration files from this filesystem directory. Alternatively, you may provide no base directory, in which case the server will attempt to infer the base directory from the URL. That is, if the requested URL is going to be serviced with a file from the document root, then it will start searching from the document root. The same applies to URL mappings, user public information directories, and CGI mappings.

You also specify the name of the configuration file to search for within the base directory.

Normally, you will want to centralize all of your configuration information for the subdirectories of the base directory into the configuration file in the base directory. This makes the server more efficient, because it does not have to waste time searching for configuration files in each of the subdirectories.

However, for convenience you will sometimes want to tell the server to search the subdirectories. That is, say that you have selected the base directory inferred from URL translation, and selected *.nsconfig* for your configuration filename. When a user requests the filesystem path */usr/ns-home/docs/icons/logo.gif*, instead of just searching for */usr/ns-home/docs/.nsconfig* you want the server to search all subdirectories:

```
/usr/ns-home/docs/.nsconfig
/usr/ns-home/docs/gfx/.nsconfig
/usr/ns-home/docs/gfx/icons/.nsconfig
```

Finally, enter a wildcard pattern of file types you want to absolutely disable in directories where dynamic configuration is enabled. To disable CGI programs and parsed HTML, use:

```
*(cgi|parsed-html)
```

Writing a Dynamic Configuration File

Dynamic configuration files consist of sets of directives that control the server. These sets are surrounded by **Files** directives that tell the server which files in the configuration file's directory the directives apply to. For example:

```
<Files PATTERN1>
... directives ...
</Files>
<Files PATTERN2>
... directives ...
</Files>
```

PATTERN1 and PATTERN2 are wildcard patterns that tell the server which filesystem pathnames to apply the directives they surround to. Any pattern given is first prefixed with the directory containing the configuration file to ensure that it is applied only to subdirectories. There can be as many files sets in the *.nsconfig* file as you need.

The file can contain blank lines. Lines beginning with # are treated as comments.

Each directive can take a variable number of parameters. The directives that can appear inside Files regions are:

- AddType exp=SHEXP type=mime-type enc=http-encoding

  AddType assigns the given type or encoding to the paths represented by the wildcard pattern SHEXP. One or both of "type" and "enc" can appear, but only one "exp".

- ErrorFile reason=error-string code=error-code path=html-file

  ErrorFile causes the HTML file described by the URL suffix "path" to be sent in place of the server's default error message when an error, described by one or both of "reason" and "code", occurs. "path" is a valid URL to the local server, but without the *http://server* prefix. The error codes are the standard HTTP error codes:

  – 401 Unauthorized

  – 403 Forbidden

  – 404 Not found

  – 500 Server error

- RequireAuth dbm=dbmfile userfile=user:passwords realm=string userpat=PATTERN userlist=user1,user2,...usern

  RequireAuth lets you ask the user for a username and a password when accessing the directory. "realm" is a unique string to tell your users which password they should use. "dbm" is a user database. "userfile" is a simple file consisting of lines in the format *user:encryptedpassword*. "userpat" and "userlist" determine which users from the given dbm or userfile are allowed access.

- RestrictAccess method=HTTP-method type=allow | deny ip=addrpattern dns=hostpattern return-code=403 | 404

RestrictAccess applies access control to the directory and restricts certain users. “method” is an optional parameter specifying a wildcard pattern of HTTP methods to protect (no method means all of them). More than one **RestrictAccess** can appear in the file. “type” determines whether the IP address wildcard pattern or hostname wildcard pattern is allowed or denied access. If the only **RestrictAccess** directives in a Files set are of type “allow”, then all hosts not specified by the patterns are denied.

Configuration Templates

Configuration templates are an easy way to apply a set of options to specific files or directories that your server maintains. For example, you can create a template that configures access logging. You can then apply that template to the files and directories you want to log. This saves you having to individually configure access logging for all the files and directories.

Creating a Template

To create a template:

1. Click the link called “Create a template.”
2. In the form that appears, type the name you want to give the template.
3. Click the *Make These Changes* button. The template is created as a named object in *obj.conf*.
4. When you return to the Server Manager, you configure the attributes for your template, then click the “Apply a template to part of your server” link to apply the template to files or documents in your server. When configuring attributes for a template, the forms list the template name at the top of the form.

Not all links in the Server Manager apply to templates. The links you can use for templates are listed here for your reference (a graphic also appears next to the link in the Server Manager page).

Applying a Template to Parts of the Server

You can apply templates to files or directories in your server. You can either specifically choose files and directories, or you can specify wildcard patterns (such as **.gif*).

To apply a template:

1. Click the "Apply a template to part of your server" link.
2. In the form that appears, click a button to choose the resources you want to apply the template to. The buttons are described as follows:
   - *Choose Entire Server* applies the template to every document the server maintains.
   - *Browse files* applies the template to specific files or directories. You can view files or directories and you can specify whether or not to view files that are symlinks to files in other directories.
   - *Choose Wildcard Patterns* lets you apply the template to files or directories that you specify with wildcard patterns. This is an easy way of specifying lots of files in different directories (such as **.gif*) or many subdirectories (such as */public/**).
   - *Choose a template* is normally used to choose templates when configuring other aspects of the server. Use the template drop-down list instead of clicking this button to apply a template.
3. Select the template you want to apply. The None template can be applied to files or directories to remove any templates previously applied to the resource.
4. Click the *Make These Changes* button. The template is applied to the resources. You'll need to restart the server for the template to take effect.

Removing a Template

When you remove a template, the template is deleted from the *obj.conf* file (it was stored as a named object). The template isn't unapplied from any resources you have applied it to. This means that before you remove a template, you should apply the None template to any files or directories first. You can also remove a template and then search and replace all instances of the template in *obj.conf*. If you don't remove these entries, you'll get a server misconfiguration error when anyone accesses the files or directories that had the template.

To remove a template:

1. Click the link called “Remove a template.”
2. In the form that appears, select the template you want to remove.
3. Click the *Make These Changes* button. The template is removed.

Error Handling

The error handling section lets you view the error log file and customize error messages sent to clients.

Viewing the Error Log File

Click the View error log link to see a list of all errors the server has encountered since the log file was started (you can rotate log files to save the current log file and then start adding entries to a new log file).

You can limit the number of errors you see by typing **?nn** at the end of the URL in the Locations box in Netscape Navigator. The question mark specifies a query, and nn represents the number of errors you want to view. After you type the query at the end of the URL, press <Enter>. The page redraws, showing you the most recent errors.

Customizing Error Responses

You can specify a custom error response that sends a detailed message to users when they encounter errors from your server. You can specify a file to send or a CGI script to run.

You may want to change the way a directory behaves when it gets an error. Instead of sending back the default file, you may want to send a custom error response instead. For example, if a user tried to repeatedly connect to a section of your server protected by access control, and failed, you could have the error file returned with information on how to get an account.

What Are the Errors?

There are several different kinds of errors that you can customize the server's response to:

- *Unauthorized*. This error occurs when a user tries to access a document in the server that is protected by access control but doesn't have permission to access the document. You might send information on how they can get access.
- *Forbidden*. This error occurs when the server doesn't have filesystem permissions to read something, or if the server is not permitted to follow symbolic links.
- *Not Found*. This error occurs when the server can't find a document, or when it has been instructed to deny the existence of a document.
- *Server Error*. This error occurs when the server is misconfigured or when a catastrophic error occurs, such as the system running out of memory or a core dump.

Setting Up the Response

Before you can set up the response, you need to either write the HTML file to send, or create the CGI program to run. After you do this, jump to the customizing error response form and select the error response you want to customize.

Type the absolute pathname to the file or script you want to return for that error code. Check if the file is a CGI program that you want to run. Do this for each of the errors you want to customize, then click the *Make These Changes* button.

To remove a customization, return to the form and delete the filename from the text box next to the error code.

Logging Configuration

You can customize access logging to any resource by specifying whether or not to log accesses, who not to record accesses from, and whether the server should spend time looking up the domain names of clients when they access a resource.

You need to decide whether or not you want to use access logging for this resource in the server. If you decide to log accesses, then the server records in common logfile format many things about the request, including the client's address, how long the transfer took, the response the server made, how many bytes were transferred, and whether or not the user was authenticated.

You also need to choose a name and location for the access log file. If you specify a partial pathname here, the server assumes the path is placed in the server root in the directory logs. The means you create a log file for each resource you want to log.

You also have the option of recording only the IP addresses of incoming requests. DNS lookups to turn an IP address into a hostname, but this can be quite costly in terms of performance.

You can also choose to have the server not log accesses from certain addresses. That way, you can tell the server not to log accesses from addresses within your own company. Type a wildcard pattern of hosts the server should not record accesses from; for example *.sgi.com doesn’t log access from people with the domain sgi.com. You type wildcard patterns for either hostnames, IP addresses, or both.

Chapter 4

Server Configuration Files

This chapter describes the configuration files the server uses. When you use the Server Manager, the forms act as an interface to the configuration files. The changes you make in the Server Manager are saved in these configuration files. You can use these files to configure the server manually.

You might need to configure the server manually 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 change information manually in the server'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.

The server configuration files are kept in the directory *httpd-80/config* (*https-443/config* for the Commerce Server) 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 server and manages the files and directories that the server can send to clients.
- *mime.types* is the file that 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*.

The magnus.conf File

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

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 occur only with the Init directive) can be continued with a \ character before the linefeed.

Warning: 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 form, you can use the backslash character.

Example 4-1 Sample magnus.conf file

```
# Sample magnus.conf file for Netscape server 1.1
# The server's home--it’s root directory
#ServerRoot /usr/ns-home/https-443
# Which port?
Port 443
# This tells the server to get its objects from obj.conf, and use
# the "default" object as the default.
LoadObjects obj.conf
RootObject default
# The logfile for errors, and the file where it should keep
# the pid of the master server process.
ErrorLog /usr/ns-home/https-443/logs/errors
```

```
PidLog /usr/ns-home/https-443/logs/pid
# Which user should the server run as? This is the Unix user
# account name.
User http
# The server's name
ServerName www.sgi.com
# Use DNS? (turn this off for performance reasons)
DNS on
# Processes - maximum and minimum to spawn
MinProcs 16
MaxProcs 32
# How many requests should a process serve before being respawned?
ProcessLife 512
# Security directives: is security on, where is my keyfile,
# which ciphers should I support (Ciphers directive is on the
# US version only)
Security on
Keyfile ServerKey.der
Certfile ServerCert.der
Ciphers +rc4, +des, +rc4export
# Initializations, such as log files and loading NSAPI libraries
Init fn=load-types mime-types=mime.types
Init fn=init-clf global=/usr/ns-home/https-443/logs/access
```

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 described below.

- ServerName defines the server hostname.
- Port defines the TCP port that the server listens to.
- User specifies the server’s IRIX user account.
- MaxProcs sets the maximum number of active processes.
- MinProcs sets the minimum number of active processes.
- 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 server's main process ID (PID).
- LoadObjects specifies a start-up 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.
- DNS specifies if the server does DNS lookups on clients who access the server.
- Security specifies the type of security the server has. This is available only for the Commerce server, and some functions are available only in the US version of the Commerce server.

ServerName

ServerName tells the server what to put in the hostname section of any URLs it sends to the client. This affects URLs that the server automatically generates; it doesn't affect the URLs for directories and files stored in the server. This name is what all clients use to access the server; they need to combine this name with the port number if the port number is anything other than 80.

This name should be the alias name if your server uses an alias. 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 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 server won't start, and you'll get a message telling you to set this value manually.

- EXAMPLES

```
ServerName server.sgi.com
ServerName www.server.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. There should be only one Port directive in *magnus.conf*.

Note: The port you choose can affect how users configure their navigators. Users must specify the port number when accessing the server if the port number is anything other than 80 or 443.

- SYNTAX

  `Port` *number*

  number is a whole number between 0 and 65535.

  DEFAULT

  If no port is specified, the server assumes 80.

- EXAMPLES

  ```
  Port 80
  Port 8080
  Port 8000
  ```

Warning: If you change the default HTTP port number from 80, you must edit the /etc/init.d/netsite file (/etc/init.d/netsite_commerce for the Commerce Server) to reflect the change.

User

User specifies the IRIX user account for the server. If the server 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 root.

The user account you specify should have read permission to the server's root and subdirectories. The user account should have write access to the `logs` directory and execute permissions to any CGI programs. The user account should not have write access to the configuration files. This ensures that in the unlikely event someone compromises the server, they won't be able to change configuration files and gain broader access to your host.

Although you can use the `nobody` user, it isn't recommended.

- SYNTAX

  `User` *name*

  name is the 8-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 http
  User server
  User nobody
  ```

MaxProcs

MaxProcs sets the maximum number of processes the server can have active. The 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 server keeps active.

Choose a number that is appropriate for the type of access you expect for the server. If this number is too small, clients will experience delays. If the number is too large, you might run out of memory and get an error such as "could not fork new process."

- 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 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 server's child processes serves before the processes exit and are 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 server ensures that memory isn't wasted by "lost" processes. For example, on rare occasions, there might be resource "leaks" in operating system libraries or in the server itself (though none are currently known). By specifying a lifespan, a process answers a certain number of requests before exiting and respawning.

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-server/logs/errors
ErrorLog SYSLOG
```

PidLog

PidLog specifies a file in which to record the process ID (PID) of the base server process. Some of the server support programs 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 (PID) is stored.

  DEFAULT

  There is no default.

- EXAMPLES

```
PidLog /var/ns-server/logs/pid
PidLog /tmp/ns-server.pid
```

LoadObjects

LoadObjects specifies one or more object configuration files to use on startup; these files tell the server where to find documents.

Note: Although you can have more than one object configuration file, the server'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-server/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 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. CGI programs must be linked statically, and any binaries (*perl* or */bin/sh*) must be copied to the Chroot directory.

The user public information directory feature isn't available unless a copy of */etc/passwd* is kept in the Chroot directory and all of the users home directories are exactly mirrored within the Chroot directory.

A server using Chroot can't be restarted with the `-HUP` signal.

Logs and server configuration files should be kept outside 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-httpd
  Chroot /www
  ```

Init

Init is a special directive that initializes certain server subsystems such as access logging and user public directories. 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. Brief descriptions appear below.

  - **load-types** maps file extensions to MIME types.
  - **init-clf** initializes the Common Log subsystem.
  - **init-uhome** loads user home directory information.
  - **cindex-init** sets global characteristics for fancy indexing.

  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 "The mime.types File" on page 94 for more detailed information.

  Calling this function is crucial if you use the Administration forms.

  - PARAMETERS

  `mime-types` specifies either the full path to the global MIME types file or a filename relative to the server configuration directory. The 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 re-opened).

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

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

For example, you use init-clf to specify a name that refers to a log file, (such as `http-log=/var/ns-server/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=server-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*.

Note: You also can use AddLog to store transactions in more than one log file. See “AddLog” on page 91 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, you need to rename the file and then send the **-HUP** signal to restart the server. The server uses the inode number, but when you do a soft restart, the server first looks for the filename. 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 server-log function.

- EXAMPLES

```
Init fn=init-clf global=/var/ns-server/logs/access
Init fn=init-clf global=/tmp/server-access
```

init-uhome

The function init-uhome loads information about the system's user home directories into internal hash tables. This increases the shared memory size slightly, but saves CPU cycles for servers that have a lot of traffic to home directories.

- PARAMETERS

"pwfile" (optional) specifies the full filesystem path to a file other than */etc/passwd*. If not listed, the default (NIS or */etc/passwd*) is used.

- EXAMPLES

```
Init fn=init-uhome
Init fn=init-uhome pwfile=/etc/passwd-http
```

cindex-init

The function cindex-init configures fancy indexing. You must use fancy indexing for this function to work. See "Directory Indexing" on page 39 for more information on directory indexing.

DEFAULT

```
Init fn=cindex-init widths=22,1,1,33
```

- PARAMETERS

"opts" (optional) specifies options to activate for indexing. You specify a string of letters, one for each option to activate:

– **i** makes all icons links.

– **s** makes the server scan HTML files in the directory it's indexing in order to place their titles in the description field.

"widths" specifies the width for each column in the indexing printout. A zero width disables the column. The string should be a comma-separated list of numbers that specify the column widths according to name, last modified date, size, and description.

"ignore" specifies a wildcard pattern for filenames the server can ignore while indexing. Filenames starting with a dot are automatically ignored.

"icon-uri" specifies the prefix the server uses for icons. By default, this is */mc-icons/*. The server looks in this directory for the GIF files to use in fancy indexing.

- EXAMPLES

```
Init fn=cindex-init widths=50,1,1,0
Init fn=cindex-init widths=50,1,1,30 opts=s
Init fn=cindex-init widths=22,0,0,50 opts=is
```

The obj.conf File

The object configuration file, called *obj.conf*, uses objects to control how the server handles documents.

Objects (also referred to as resources) are settings that tell the server how to treat all documents, CGI programs, directories, imagemap files, and so on. You can define objects in two ways:

- Use wildcard patterns. Anything that matches a specified wildcard pattern belongs to the same object. This object grouping can then be used to control (in fine detail) the behavior of the 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).

- Create a template. You create a template with specific settings, and apply that template to directories or files. You then use name translation functions to tell the server which directories or files use the template.

  Using templates makes it easy to configure scattered directories. For example, if user home directories appear in several places, you use one template instead of configuring each directory separately.

Also, with *cgi-bin* directories, all files are treated as programs and are run rather than sent. If you create a CGI template and assign the template to CGI programs (using name translation), then you can have any number of CGI directories and they all use the template configuration.

The Structure of obj.conf

The *obj.conf* file must have four specific objects in it (see “Required Objects for obj.conf” on page 75 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>
<Object name=cgi>
Directives
…
</Object>
```

You use wildcard patterns (see “Understanding Wildcard Patterns” on page 11) to control what is grouped in the object, or you use a name to create a template. You then specify one or more directives to control what the server does when it encounters anything that uses the template or that matches the wildcard pattern specified with `ppath`.

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

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.

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

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 only occur with the Init directive) can be continued with a \ character before the linefeed.

A Sample Object

The following sample object applies to a directory called *user/public/*.

```
<Object ppath="/user/public/*">
PathCheck fn=deny-existence
Service fn=server-retrieve
</Object>
```

When the server receives a request for a document in this directory, it doesn't send the document. Instead, it denies the existence of any files or subdirectories and displays the "not found" error message.

The Service directive tells the server to get the documents by default.

Example 4-2 Sample obj.conf File

```
#
# Sample obj.conf file for Netscape server 1.1.
#
# This file was automatically generated by the server.
# Edit at your own risk.
# The default object. This is what the server uses if none of the other
# objects fit.
#
# This one has a CGI directory specified in /usr/local/bin/cgi,
# a directory mapping to a bigger disk in /gig-drive/sales,
# and a document root of /usr/http-docs
```

```
#
<Object name="default">
NameTrans fn="pfx2dir" from="/mc-icons" dir="/usr/ns-home/mc-icons"
NameTrans fn="pfx2dir" from="/cgi-bin" dir="/usr/local/bin/cgi" name="cgi"
NameTrans fn="pfx2dir" from="/sales" dir="/gig-drive/sales"
NameTrans fn="document-root" root="/usr/http-docs"
PathCheck fn="unix-uri-clean"
PathCheck fn="find-pathinfo"
PathCheck fn="find-index" index-names="index.html,home.html"
ObjectType fn="type-by-extension"
ObjectType fn="force-type" type="text/plain"
Service fn="imagemap" method="(GET|HEAD)" type="magnus-internal/imagemap"
Service fn="index-common" method="(GET|HEAD)" type="magnus-internal/directory"
Service fn="send-cgi" type="magnus-internal/cgi"
Service fn="send-file" method="(GET|HEAD)" type="*~magnus-internal/*"
AddLog fn="common-log"
</Object>
# All CGI directories have these configuration options set
<Object name="cgi">
ObjectType fn="force-type" type="magnus-internal/cgi"
Service fn="send-cgi"
</Object>
# This is a directory that requires authentication to enter, and uses server
# parsed HTML
<Object ppath="/usr/http-docs/private/*">
AuthTrans fn="basic-ncsa" dbm="/usr/ns-home/userdb/users" auth-type="basic"
<Client dns="*~*.sgi.com>
PathCheck fn="deny-existence"
</Client>
PathCheck realm="Confidential Documents" fn="require-auth" auth-type="basic"
Service fn="parse-html" method="(GET|HEAD)" type="magnus-internal/parsed-html"
</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 server. These functions control local file access and CGI execution.

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

The Default Object

```
<Object name="default">
NameTrans fn="pfx2dir" from="/mc-icons"
dir="/usr/home/mc-icons"
NameTrans fn="document-root" root="/usr/http-docs"
PathCheck fn="unix-uri-clean"
PathCheck fn="find-pathinfo"
ObjectType fn="type-by-extension"
ObjectType fn="force-type" type="text/plain"
Service fn="imagemap" method="(GET|HEAD)"
        type="magnus-internal/imagemap"
Service fn="index-common" method="(GET|HEAD)"
        type="magnus-internal/directory"
Service fn="send-file" method="(GET|HEAD)"
        type="*~magnus-internal/*"
</Object>
```

CGI Object

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

```
  <Object name="cgi">
  ObjectType fn="force-type" type="magnus-internal/cgi"
  Service fn="send-cgi"
  </Object>
```

How the Server Handles Objects

The Netscape server design 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, server 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 or a mirror site URL. It might also be kept intact and retrieved as is (the normal server 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 server 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 server-service routine, or local file blast, 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 described below.

- 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 different users.

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 be present. This is done in the hope that multiple translation schemes can be easily incorporated, as well as providing 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 are applied.

- **Basic NCSA**

  DESCRIPTION

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

  This function is usually used in conjunction with the PathCheck function require-server-auth.

  PARAMETERS

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

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.

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 the next one.

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. See "PathCheck" on page 82 for more information on group files.

EXAMPLES

```
AuthTrans fn=basic-ncsa auth-type=basic
  dbm=/var/ns-server/userdb/rs
AuthTrans fn=basic-ncsa auth-type=basic
  userfile=/var/ns-server/.htpasswd
  grpfile=/var/ns-server/.grpfile
```

NameTrans

NameTrans stands for Name Translation. This directive maps URLs to physical directories on your server. For example, the URL *http://www.test.com/some/file* is a virtual path that could map to the real directory called */docs/http/files/*.

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.

- **Prefix to directory**

  DESCRIPTION

  `pfx2dir` looks for a directory prefix in the path and replaces the prefix with a real directory name. Don't use trailing slashes in either the prefix or the directory.

  PARAMETERS

  from is the prefix to map.

dir is the directory that the prefix is mapped to.

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

EXAMPLES

```
NameTrans fn=pfx2dir from=/cgi-bin dir=/httpd/cgi-bin
name=cgi
NameTrans fn=pfx2dir from=/icons dir=/httpd/mc-icons
```

- **Public information directories**

  DESCRIPTION

  The unix-home function lets your internal users provide information to external users. You specify a URL prefix that corresponds to the user directories. Any request that begins with the prefix is translated to the user's home directory.

  You specify the list of users with either the */etc/passwd* file or a file with a similar structure. Each line in the file should have this structure (the elements in the passwd file that aren't needed are indicated with *):

  ```
     username:*:*:groupid:*:homedir:*
  ```

  If you want the server to scan the password file only once at startup, use the Init function init-uhome.

  PARAMETERS

  from is the URL prefix to translate.

  subdir is the subdirectory of the user's directory that contains their documents.

  pwfile is the full path and filename of the password file you want to use, if it's different from */etc/passwd* or the NIS database.

  name (optional) specifies a named object that configures this directory.

  EXAMPLES

  ```
  NameTrans fn=unix-home prefix=/~ subdir=public_html
  name=userhome
  NameTrans fn=unix-home prefix=/u subdir=public_html
  name=userhome
  ```

- **Document root**

  DESCRIPTION

The document-root function specifies the directory that contains all of your documents. This directory is prepended to the virtual path that the client sends to form the full pathname of the file or directory. For example, the client sends */home/file.html*, which is translated to *<docroot>/home/file.html*.

PARAMETERS

root specifies the document directory.

EXAMPLES

```
NameTrans fn=document-root root=/netscape/docs
```

- **Redirection**

  DESCRIPTION

  The redirect function lets you change URLs and send the updated URL to the client. When a client accesses your server with an old path, they are told to use the new URL you provide.

  PARAMETERS

  from specifies the old path.

  url specifies a complete URL to return to the client. If you use this parameter, don't use `url-prefix` (and vice-versa).

  url-prefix is the new URL to pass to the client. The "from" prefix is simply replaced by this URL prefix.

  EXAMPLES

```
NameTrans fn=redirect from=/ url-prefix=http://tmpserver
nameTrans fn=redirect from=/toopopular
   url=http://bigger/better/stronger/morepopular/new.html
```

- **Home page**

  DESCRIPTION

  The home-page function specifies the home page for your server. Whenever a user doesn't specify a path, they'll get an index file for the document root directory. If you use this function, you specify the HTML file they see instead. This file must be on the server's local filesystem.

- PARAMETERS

path is the path and name of the home page *.HTML* file.

- EXAMPLES

  ```
  NameTrans fn=home-page path=homepage.html
  NameTrans fn=home-page path=/httpd/docs/home.html
  ```

PathCheck

PathCheck directives check the full filesystem path that is returned after all of the NameTrans directives finish running. The paths are checked for things such as CGI path info and for dangerous elements such as /./and /../ and //, and then any access restriction is applied.

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

- **URI cleaning**

  DESCRIPTION

  The unix-uri-clean function denies access to any requested URL that contains /./, /../ or // (these URLs are potential security problems). If you use scripts with these elements in paths, use the find-pathinfo function (described in the next section) before unix-uri-clean.

  PARAMETERS

  None.

  EXAMPLES

  ```
  PathCheck fn=unix-uri-clean
  ```

- **Find index files for directories**

  DESCRIPTION

  The find-index function determines if the requested path is a directory. If it is, the function searches for an index file in the directory, and then changes the path to point to the index file. If no index file is found, the server generates a directory listing.

  PARAMETERS

  index-names is a comma-separated list of index filenames to look for. Use spaces only if they are part of a filename.

  EXAMPLES

```
PathCheck fn=find-index index-names=index.html,home.html
```

- **Require authorization**

  DESCRIPTION

  The require-auth function allows access to objects only if the user or group is authorized. You must use the AuthTrans directive before using the PathCheck directive with this function.

  PARAMETERS

  auth-type is the type of HTTP authorization to use. This currently can be set only to basic.

  realm is a string (enclosed in double-quotation marks) sent to the client application so users can see what object they need authorization for.

  auth-user (optional) specifies a list of users who get access. The list should be enclosed in parentheses with each user name separated by the `|` character.

  auth-group (optional) specifies a list of groups that get access. Groups are listed in the password-type file.

  EXAMPLES

```
PathCheck fn=require-auth auth-type=basic
realm="Marketing Plans"
          auth-group=mktg auth-users=(jdoe|johnd|janed)
```

- **Deny path's existence**

  DESCRIPTION

  The deny-existence function sends a `not found` message when a client tries to access a specified path. If this function appears in a `<client>` region, then it performs access control. Note that `not found` is sent instead of `forbidden`, which means the user can't tell if the path exists or not.

  PARAMETERS

  path (optional) is a wildcard pattern of the path to hide. If no paths are specified, then all paths are hidden.

  bong-msg specifies a file to send instead of the `not found` message. The file should be an HTML file specified as an absolute path.

  EXAMPLES

```
PathCheck fn=deny-existence path=*/~
<client>
PathCheck fn=deny-existence bong-msg=/svr/msg/go-away.html
</client>
```

- **Find filesystem links**

  DESCRIPTION

  The find-links function searches the current path for symbolic or hard links to other directories or filesystems. If any are found, the server returns an error. Usually you use this function in directories you don't trust (such as user home directories). This prevents someone from pointing to information that you don't want made public.

  PARAMETERS

  disable is a character string of links to disable:

  - **h** is hard links
  - **s** is soft links
  - **o** allows symbolic links from user home directories only if the user owns the target of the link.

  dir is the directory to begin checking. If you specify an absolute path, any request to that path and it's subdirectories is checked for symlinks. If you specify a partial path, any request containing that partial path is checked for symlinks. For example, if you use */user/* and a request comes in for *some/user/directory*, then that directory is checked for symlinks.

  EXAMPLES

  ```
  PathCheck fn=find-links disable=sh dir=/foreign-dir
  PathCheck fn=find-links disable=so dir=public_html
  ```

- **Find path information**

  DESCRIPTION

  The find-pathinfo function uses extra path information if it can't find a file in a specified path. Extra path information is included after the path and file in the URL.

  PARAMETERS

  None.

EXAMPLES

```
PathCheck fn=find-pathinfo
```

ObjectType

ObjectType directives determine the MIME type of the file sent to the client. This type is usually sent back to the client to let the client decide what to do. MIME attributes currently sent are `type`, `encoding`, and `language`.

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.

- **File typing by extension**

  DESCRIPTION

  The type-by-extension function uses file extensions to determine information about files. (Extensions are strings after the last period in a filenamefilename.) This matches an incoming request to extensions in the *mime.types* file. The MIME type is added to the “content-type” header sent back to the client. The type can be set to internal server types that have special results when combined with the functions you write using the NSAPI (see the *Netscape Servers Programmer's Guide* for more information.)

  PARAMETERS

  None.

  EXAMPLES

  ```
  ObjectType fn=type-by-extension
  ```

- **File typing by wildcard pattern**

  DESCRIPTION

  The type-by-exp function matches the current path with a wildcard expression. If the two match, the type parameter information is applied to the file. This is the same as type by extension, except you use wildcard patterns for the files or directories specified in the URLs.

  PARAMETERS

exp is the wildcard pattern of files or directories that the information is applied to.

type (optional) is the type to assign to any matching files.

enc (optional) is the encoding given to matching files (the "content-encoding" header).

lang (optional) is the language assigned to matching paths.

EXAMPLES

```
ObjectType fn=type-by-exp exp=*.test type=application/html
```

Forcing file types

DESCRIPTION

The force-type function assigns a type to objects. This is used to specify a default object type.

PARAMETERS

type is the type to assign to matching files.

enc (optional) is the encoding given to matching files.

lang (optional) is the language assigned to matching paths.

EXAMPLES

```
ObjectType fn=force-type type=text/plain
ObjectType fn=force-type lang=en_US
```

- **Server-parsed HTML**

  DESCRIPTION

  The `shtml-hacktype` function provides backward compatibility with server-side includes.

  Server-side includes require a different MIME type than HTML. This means that your parsed documents must have different file extensions than nonparsed documents. If this is a problem, this function can be used as a solution. Another is to have the server parse all HTML, but this can have detrimental performance effects. You can also check for the execute bit on the file. If it's there, the file is parsed; otherwise, it isn't. None of these solutions is recommended. See the *Netscape Server Programmer's Guide* for more information on server-parsed HTML.

  PARAMETERS

exec-hack (optional) checks if the execute bit is enables for the file. If you don't specify this parameter, all files are marked as parsed.

EXAMPLES

```
ObjectType fn=shtml-hacktype exec-hack=true
```

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.

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

type (not with `server-retrieve`) specifies a wildcard pattern of MIME types to apply the directive to. The 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.

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.

query specifies a wildcard pattern of search queries that must be present for the directive to run.

If an object contains more than one Service function, the first is one is used and all subsequent ones are ignored.

- **Send a plain file**

  DESCRIPTION

  The send-file function sends the contents of a plain text file to the client. If this function finds any extra path information, it doesn't send the text file to the client.

  PARAMETERS

  None.

  EXAMPLES

```
Service type=*~magnus-internal/* method=(GET|HEAD)
fn=send-file
```

- **Send an error message**

  DESCRIPTION

  The send-error function sends an HTML file to the client regardless of the path the client requested. This is used mainly for error messages.

  PARAMETERS

  path specifies the full filesystem path of the HTML file.

  EXAMPLES

  ```
  Service fn=send-error path=/popular/service/we-moved.html
  Service fn=send-error path=/http/errors/no-post.html
  ```

- **Append a trailer to HTML documents**

  DESCRIPTION

  The append-trailer function appends text to the end of every HTML document from the object. This is mainly used for author information and copyright text. The date the file was last modified can automatically be included.

  If there is extra path information, the request is flagged as "not found" and the document isn't sent to the client.

  PARAMETERS

  trailer is the text you want to append to all HTML documents. The text can contain HTML tags and can be approximately 700 characters long. The string `:LASTMOD:` is substituted with the date the file was last modified; you must also specify a time format with `timefmt`.

  timefmt is a time string in the *strftime*(3C) function format.

  EXAMPLES

  ```
  Service fn=Service type=text/html fn=append-trailer
          trailer="<hr><img scr=/logo,gif> Copyright 1995"
  Service fn=Service type=text/html fn=append-trailer
  timefmt="%D"
          trailer="<hr>File last updated on: :LASTMOD:"
  ```

- **Run a CGI program**

  DESCRIPTION

The send-cgi function runs a file as a CGI program and sends the results to the client.

PARAMETERS

None.

EXAMPLES

```
Service fn=send-cgi
Service type=magnus_internal/cgi fn=send-cgi
```

- **Set a default query handler**

  DESCRIPTION

  The query-handler function runs a CGI program instead of referencing the path requested. This is used mainly to support the obsolete ISINDEX tag. If possible, use a FORM instead.

  PARAMETERS

  path is the full path and filename of the CGI program to run.

  EXAMPLES

  ```
  Service query=* fn=query-handler path=/http/cgi/do-grep
  Service query=* fn=query-handler path=/http/cgi/proc-info
  ```

- **Use an imagemap**

  DESCRIPTION

  The imagemap function includes imagemap files.

  PARAMETERS

  None.

  EXAMPLES

  ```
  Service type=magnus_internal/imagemap method=(GET|HEAD)
  fn=imagemap
  ```

- **Simple directory indexing**

  DESCRIPTION

  The index-simple function scans a directory and produces an HTML file of a bulleted list of files in the directory. Each file appears as a link.

If this function encounters a subdirectory, the link redirects the user to the subdirectory. In directories with subdirectories, use fancy indexing as described in the following section.

PARAMETERS

None.

EXAMPLES

```
Service type=magnus-internal/directory fn=index-simple
```

Fancy directory indexing

DESCRIPTION

The index-common function scans a directory and produces an HTML file of a bulleted list of files in the directory. Each file appears as a link. This function produces a format common to the CERN and NCSA HTTP servers. It includes more information than simple indexing and references icon files.

PARAMETERS

header is a file to prepend to the indexing that introduces the contents of the directory. If you specify a filename for this parameter, the server looks for the filename as an *.HTML* file, and then incorporates the file in the directory list as HTML; otherwise, the file is included as plain text.

readme is a file (HTML or plain text) to append to the indexing. This gives more information about the contents of the directory.

EXAMPLES

```
Service type=magnus_internal/directory method=(GET|HEAD)
fn=index-common header=hdr.html readme=end-text.txt
```

- **Parsed HTML (server-side includes)**

  DESCRIPTION

  The parse-html function parses an HTML document, scanning for embedded server directives. These server directives provide certain information only the server has, or they include the contents of other files. Parsing lots of HTML documents can reduce server performance.

  PARAMETERS

  opts are parsing options. The `no-exec` option is the only currently available option—it disables the `exec` directive.

EXAMPLES

```
Service type=magnus_internal/parsed-html
method=(GET|HEAD)         fn=parse-html
```

AddLog

After the request is finished and the server has stopped talking to the client, the 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 Format**

  DESCRIPTION

  server-log is an AddLog function that records request-specific data in the common log format (used by most HTTP servers). There is a log analyzer in the */extras* directory. There are also a number of free statistics generators for the common format.

  The log format is specified by the `init-server` 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 hostname 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=server-log
  # Log non-local accesses to another log file
  <Client ip=*~198.93.9[2345].*>
  AddLog fn=server-log name=nonlocal
  </Client>
  ```

- **Record the client software name**

DESCRIPTION

The record-useragent function records the IP address of each client, followed by their User-Agent HTTP header. This tells the server what version of Netscape Navigator (or what other browser) the client used for this transaction.

PARAMETERS

name (optional) gives the name of the log file where the log is recorded—it must have been specified with the init-clf function. If no name is listed, the log is recorded in global.

EXAMPLES

```
AddLog fn=record-useragent name=browsers-used
```

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 conditions occur, the server can send a short HTML page to the client telling them in very general terms about the error.

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

Table 4-1 lists the errors returned by the server. Each error shows the 3-digit HTTP code and the error with a short description.

Table 4-1 Server Error Codes

Error Code	Meaning
`401 Unauthorized`	(For Administration forms only). The server requires HTTP user authorization to allow access to the Administration forms, and either the client 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 allows you to tell people nicely that they don't have that access to your server.
`500 Server Error`	Server errors mean that an error has occurred within the server that prevents it from finishing the request. Server errors happen mainly because of misconfiguration or because host resources such as swap space are exhausted.

You can call most Service functions from Error directives.

PARAMETERS

reason gives one of the above strings (the text in bold, such as unauthorized or forbidden).

code sends the three-digit number, such as 401 or 407.

EXAMPLES

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

The 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 graphics file. The *mime.types* file contains the global file extensions for all 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 server transfers a file with one of these extensions, it uses the MIME type you specified in type.
- **icon** is the name of the icon that the browser displays. Netscape Navigator keeps these images internally. If you use a browser that doesn't have these icons, the server delivers them. Figure 4-1 illustrates the internal icons for MIME types.

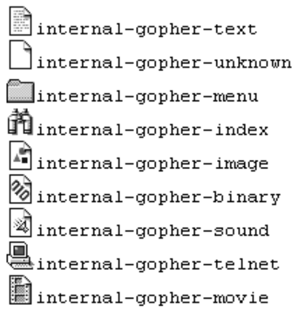


Figure 4-1 Internal Icons for MIME Types

Example 4-3 provides a sample *mime.types* file.

Example 4-3 Sample 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 server. Most
# of the MIME types are already compiled in the server. 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
# server will provide them.
type=application/oda        exts=oda
type=application/pdf        exts=pdf
type=application/x-mif      exts=mif
type=application/x-dvi      exts=dvi
type=application/x-hdf      exts=hdf
type=application/x-netcdf   exts=nc,cdf
type=application/x-texinfo  exts=texinfo,texi  icon=internal-gopher-text
type=application/zip        exts=zip
type=application/x-tar      exts=tar
type=application/x-macbinary  exts=bin
type=application/x-stuffit   exts=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
```

The 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*.

You can create multiple Administration users by creating an NCSA-type of user database, and then rename that file to replace *admpw*.

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 Server Manager, you don't need to enter a new password—but you should immediately go to the Administration Manager and set a new one.

Warning: **Because you can do this, it is very important to keep secure the server's IRIX account and to ensure that only that server account and the root account have full (write) access to the server root directory. This way, only someone running as root or with the server's user account can enter the <ServerRoot>/admserv directory and edit the file.**

Appendix A

Writing HTML Documents

This appendix describes the basic elements of HTML documents when they are stored on your computer as ASCII text. If you use a WYSIWYG Web authoring tool such as WebMagic™ Author, you do not need to know HTML. If you do not have such a tool, or want to learn HTML anyway, the information contained in this appendix can help you understand the actual ASCII construction of the HTML documents. Choose "Help/How to Create Web Services" on your Netscape Navigator to jump to a page containing more information about HTML. In addition, there are many other books and HTML documents that define the language.

What is HTML?

HyperText Markup Language defines how documents are written for the HyperText Transport Protocol (HTTP)—the protocol for the World Wide Web. HTML documents contain plain text and marker tags.

Tags describe the type of text embedded in them. HTML is not a WYSIWYG language. That is, you don't specify the format (font, size, position) of the text. Instead, you use tags to describe the organization and typeface of the text. For example, you designate what is a title, what is body text, and what is hypertext (or links to other pages) by using tags.

Because there are many different Web browsers for various platforms, HTML documents look different depending on the browser you use. For this reason, HTML provides structure for your text instead of layout.

The structure of HTML documents is hierarchical. You begin with tags for the entire document, then move to heading tags and paragraph tags.

Tools for Writing HTML Documents

Because HTML documents are plain text, you can use any text editor from the IRIX *vi* editor to complex word processors. There are also many good document conversion tools that let you convert documents to HTML. For example, you can convert a PostScript file to HTML.

This appendix describes plain HTML and assumes you're using a simple text editor.

Viewing HTML Source Text

Netscape Navigator lets you view the HTML source of any page you view with it. While viewing a page, choose "View/Source" from the menu. The navigator then displays the HTML tags and text.

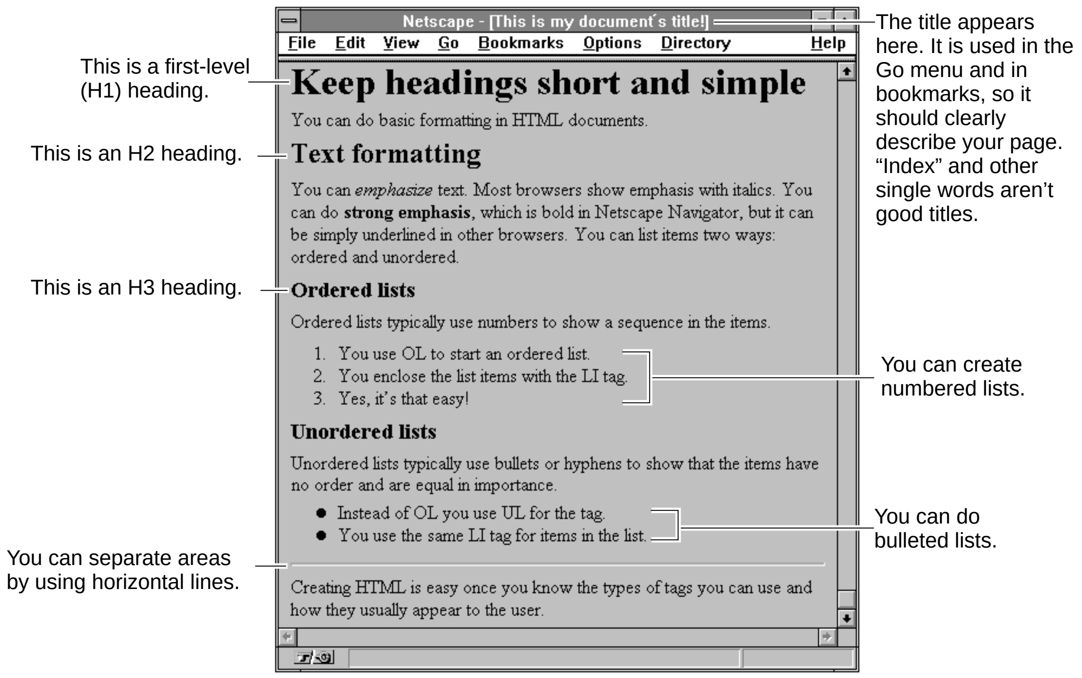
While you're learning to create HTML documents, you might find it helpful to view the source of another file to see how the author marked the text.

HTML Tags

When you write HTML documents, you use tags to mark the beginning and ending of text elements. Tags are enclosed in angle brackets. To specify a header, you type `<H1>` at the beginning of the header text and `</H1>` at the end (note that ending tags always begin with the forward slash). For example, to mark a heading, you would type

```
<H1>This is a heading</H1>
```

Figure A-1 shows a basic HTML page in Netscape Navigator.


Netscape - [This is my document's title!]
File Edit View Go Bookmarks Options Directory Help
This is a first-level (H1) heading.
This is an H2 heading.
This is an H3 heading.
You can separate areas by using horizontal lines.
The title appears here. It is used in the Go menu and in bookmarks, so it should clearly describe your page. "Index" and other single words aren't good titles.
You can create numbered lists.
You can do bulleted lists.

Keep headings short and simple

You can do basic formatting in HTML documents.

Text formatting

You can *emphasize* text. Most browsers show emphasis with italics. You can do **strong emphasis**, which is bold in Netscape Navigator, but it can be simply underlined in other browsers. You can list items two ways: ordered and unordered.

Ordered lists

Ordered lists typically use numbers to show a sequence in the items.

1. You use OL to start an ordered list.
2. You enclose the list items with the LI tag.
3. Yes, it's that easy!

Unordered lists

Unordered lists typically use bullets or hyphens to show that the items have no order and are equal in importance.

- Instead of OL you use UL for the tag.
- You use the same LI tag for items in the list.

Creating HTML is easy once you know the types of tags you can use and how they usually appear to the user.

Figure A-1 A Simple Web Page Using Various HTML Tags

Below, in Example A-1 is the HTML source for the web page shown in Figure A-1.

Example A-1 The HTML Source For Figure A-1

```
<HTML>
<HEAD>
<TITLE> This is my document’s title!</TITLE>
</HEAD>

<BODY>
<H1>Keep headings short and simple</H1>

You can do basic formatting in HTML documents.

<H2>Text formatting</H2>
<P>You can <EM>emphasize</EM> text. Most browsers show emphasis with italics.
```

```
<P>You can do <STRONG>strong emphasis</STRONG>, which is bold in Netscape Navigator, but it
can be simply underlined in other browsers.
You can list items two ways: ordered and unordered.

<H3>Ordered lists</H3>
Ordered lists typically use numbers to show a sequence in the items.
<OL>
   <LI>You use OL to start an ordered list.</LI>
   <LI>You enclose the list items with the LI tag.</LI>
   <LI>Yes, it’s that easy!</LI>
</OL>

<H3>Unordered lists</H3>
Unordered lists typically use bullets or hyphens to show that the items have no order and
are equal in importance.
<UL>
   <LI>Instead of OL you use UL for the tag.</LI>
   <LI>You use the same LI for items in the list.</LI>
</UL>
<HR>
Creating HTML is easy once you know the types of tags you can use and how they usually
appear to the user.
</BODY>
</HTML>
```

Types of Tags

There are many different types of tags. Some are used for headings, others for body text. You might want to create an HTML document that uses many tags—then use the Navigator to see how the tags look. The following table describes the most common HTML tags.

Table A-1 HTML Tags in Hierarchical Order

Tag name	Description
HTML	Defines the file as a hypertext markup language document.
HEAD	Defines the heading for the document. This usually includes the TITLE tag.

Table A-1 (continued) HTML Tags in Hierarchical Order

Tag name	Description
TITLE	Defines the title of the document. This text is used to reference the page in history lists (such as the Go menu in Netscape Navigator).
BODY	Defines where the body text of the file begins.
H1…H6	Defines six levels of headings.
P	Defines a paragraph.
OL	Defines an ordered (numbered) list. Use LI to define items in the list.
UL	Defines an unordered (bulleted) list. Use LI to define items in the list.
LI	Defines individual items in a list. Each list item appears preceded with either a number or a bullet (or hyphen in some browsers).
EM	Emphasizes characters, usually with italic type.
STRONG	Strongly emphasizes characters, usually with bold type.
CODE	Displays text in a fixed-width font, usually Courier.
B	Displays text in bold type.
I	Displays text in italic type.
TT	Displays text in a typewriter-like font. This is often the same as CODE.
IMG	Inserts an image (graphic) in the document.
HR	Separates the page with a horizontal rule line.
BR	Inserts a line break.
A	Defines attributes with links to other pages (HREF=) or sections of the current page (NAME=).

Tag Syntax

Tags are case-insensitive. <HEAD> is the same as <head> and <Head>, but most authors use all capital letters to make tags stand out from the text.

Tags can appear on the same line as the embedded text or they can appear on separate lines. The line

```
<H1> HTML is fun to write! </H1>
```

is the same as

```
<H1>
HTML is fun to write!
</H1>
```

Special Characters

There are some special characters in HTML. Because the Web is multiplatform, only a reduced set of keyboard characters is available. You can use any standard, lower ASCII character. These usually include all the characters on your keyboard (unless you have a special, non-English keyboard).

You can't use upper ASCII characters directly. For example, in the word processor you use to write your HTML, you might use a sequence of commands to type é. However, on another computer platform, this character might translate to a different character or no character at all. To use these special characters, you need to type *character* references or *entity* references.

- Character references have the format `&#`*nnn*`;` where *nnn* is a number that references the character.
- Entity references have the format `&`*nnn*`;` where *nnn* is a text string that references the character.

HTML also has reserved characters. For example, what if you want to use an angle bracket in text? How does a browser know the bracket doesn't mean the start or end of a tag? Table A-2 shows a list of reserved characters.

Table A-2 Reserved Characters

Character	Decimal	Entity
"	&#34;	&quot;
&	&#38;	&amp;
<	&#60;	&lt;
>	&#62;	&rt;

Table A-3 lists character references and entity references.

Table A-3 Special Characters in HTML

Character	Decimal	Entity
ª	&#170;	
«	&#171;	
¬	&#172;	
-	&#173;	
®	&#174;	
¯	&#175;	
°	&#176;	
	&#177;	
	&#178;	
	&#179;	
´	&#180;	
	&#181;	
¶	&#182;	

Table A-3 (continued) Special Characters in HTML

Character	Decimal	Entity
·	&#183;	
¸	&#184;	
	&#185;	
º	&#186;	
»	&#187;	
	&#188;	
	&#189;	
	&#190;	
¿	&#191;	
À	&#192;	&Agrave;
Á	&#193;	&Aacute;
Â	&#194;	&Acirc;
Ã	&#195;	&Atilde;
Ä	&#196;	&Auml;
Å	&#197;	&Aring;
Æ	&#198;	&AElig;
Ç	&#199;	&Ccedil;
È	&#200;	&Egrave;
É	&#201;	&Eacute;
Ê	&#202;	&Ecirc;
Ë	&#203;	&Euml;
Ì	&#204;	&Igrave;
Í	&#205;	&Iacute;
Î	&#206;	&Icirc;

Table A-3 (continued) Special Characters in HTML

Character	Decimal	Entity
Ï	&#207;	&Iuml;
	&#208;	&ETH;
Ñ	&#209;	&Ntilde;
Ò	&#210;	&Ograve;
Ó	&#211;	&Oacute;
Ô	&#212;	&Ocirc;
Õ	&#213;	&Otilde;
Ö	&#214;	&Ouml;
	&#215;	
Ø	&#216;	&Oslash;
Ù	&#217;	&Ugrave;
Ú	&#218;	&Uacute;
Û	&#219;	&Ucirc;
Ü	&#220;	&Uuml;
	&#221;	&Yacute;
	&#222;	&THORN;
ß	&#223;	&szlig;
à	&#224;	&agrave;
á	&#225;	&aacute;
â	&#226;	&acirc;
ã	&#227;	&atilde;
ä	&#228;	&auml;
å	&#229;	&aring;
æ	&#230;	&aelig;

Table A-3 (continued) Special Characters in HTML

Character	Decimal	Entity
ç	&#231;	&ccedil;
è	&#232;	&egrave;
é	&#233;	&eacute;
ê	&#234;	&ecirc;
ë	&#235;	&euml;
ì	&#236;	&igrave;
í	&#237;	&iacute;
î	&#238;	&icirc;
ï	&#239;	&iuml;
	&#240;	&eth;
ñ	&#241;	&ntilde;
ò	&#242;	&ograve;
ó	&#243;	&oacute;
ô	&#244;	&ocirc;
õ	&#245;	&otilde;
ö	&#246;	&ouml;
	&#247;	
ø	&#248;	&oslash;
ù	&#249;	&ugrave;
ú	&#250;	&uacute;
û	&#251;	&ucirc;
ü	&#252;	&uuml;
	&#253;	&yacute;

Table A-3 (continued) Special Characters in HTML

Character	Decimal	Entity
	&#254;	&thorn;
ÿ	&#255;	&yuml;

Adding Images to HTML Documents

There are two ways to include images (graphics) in an HTML document: inline and external. You'll usually use inline images, which appear directly in the HTML page. External images are downloaded when a user clicks a link to the image.

Because not all browsers can view various types of image files, your images should be *.GIF* files. There are lots of shareware products that create GIFs or translate one type of image (for example, BMP) to GIF.

To include an image in your HTML document, use the <IMG> tag.

```
<IMG SRC="some.gif">
```

The previous line includes the file *some.gif* in your HTML document. This assumes that the file is in the same directory as your HTML document. If the file is in another directory, use either the relative or absolute path.

You can include images on separate lines, or you can include them in text in headings, body paragraphs, and even lists.

Elements of <IMG>

The image tag has several attributes that control the graphic. The first is SRC. This defines the source for the graphic—the GIF image file.

You can control where the image is positioned relative to the text of the line it appears in by using the ALIGN attribute. You can set ALIGN to top, middle, or bottom. This positions the top, middle, or bottom of the image with the baseline of the text. If you don't specify alignment, it defaults to bottom.

.

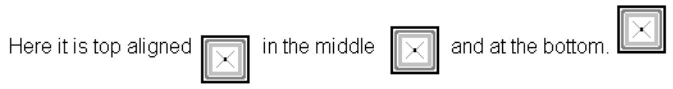


Figure A-2 Three Ways to Align Images

Note: Text does not wrap around an image.

Some browsers can't display images. You can include a text string that describes the image by using the ALT attribute.

The following example displays an image whose middle is aligned with the text baseline. The example includes the descriptive text for browsers that can't display images:

```
<IMG SRC="icon.gif" ALIGN=middle ALT="This is my icon">
```

Linking Images to Other Pages

You can use graphics as links to other pages by embedding the image tag in a link. The following example adds a circle graphic and links it to the HTML document called `circles`.

```
<A HREF="circles.html"><IMG SRC="circle.gif"></A>
```

You can combine graphics and text in one link. This means that you can click either the graphic or the text to jump to the corresponding page:

```
<A HREF="icons.html"><IMG SRC="myicon.gif">My icon is
cool</A>
```

What Are Imagemaps?

An image map is a graphic that has clickable regions that link to different pages. For example, you can have an image with a square and a circle where a click in the square takes you to one page and a click in the circle takes you to a different page.

.

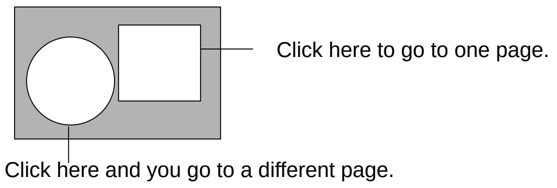

Click here to go to one page.
Click here and you go to a different page.

Figure A-3 Different Areas of an Image Map

To create an imagemap, you need a graphic file and a map file. The map file contains coordinates that define the clickable regions in the graphic.

Specifying Regions

You create an ASCII text file with the *.map* extension that contains the coordinates for the areas you want to link. Coordinates are specified from the upper left corner of the image. There are several good imaging applications that will give you the coordinates for a point in an image.

Each line in the map file specifies a clickable region. Lines have the format

method URL coordinate1 coordinate2 ...

method defines the shape the coordinates specify. Methods can be:

- `point URL` *x,y* specifies a clickable point in the image. This is useful if you click an undefined area because the click is then sent to the closest point to the clicked area.
- `circle URL` *x,y x,y* specifies a circle. Circles need two coordinates—the circle center and any point on the circle's edge.
- `rect URL` *x,y x,y* specifies a rectangle by its upper left and lower right corners.
- `poly URL` *x,y x,y*`...` specifies a polygon of up to 100 sides. Each *x,y* pair is the point where two sides of the polygon meet. The last *x,y* pair is connected to the first to enclose the polygon.
- `default URL` defines the URL to jump to when someone clicks in an area not specified by any regions. If you use a point in the map file, then the default is never used.

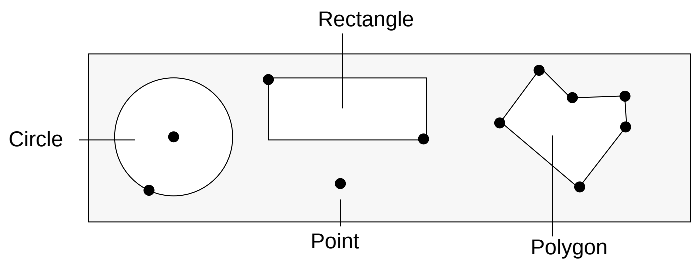


Figure A-4 Defining Regions in an Image

Coordinates are measured from the top left corner of the image.

Example A-2 Map File Example

```
# sample map file image
# This is the top left circle
circle http://www.sgi.com/funstuff 37,39 32,62
# This is the rectangle in the middle
rect http://www.sgi.com/fabulous 75,7 150,39
# This is the point
point http://www.sgi.com/homepage 125,62
# This is that weird polygon
poly http://w3.sgi.com/ 175,35 190,5 200,10 220,9 219,37 203,62
```

Glossary

API

The Netscape Application Programming Interface (NSAPI) is a set of ANSI C functions and header files that help you create functions to use with the directives in server configuration files. The Netscape Commerce and Communications Servers use this API to build the regular functions for the directives used in both *magnus.conf* and *obj.conf*. (These regular functions are described in Chapter 4.)

cache

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

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.

CGI

Common Gateway Interface—an interface for external programs to "talk" to the HTTP server. Programs that are written to use CGI are called CGI programs or CGI scripts. CGI programs do things such as handle forms or perform output parsing not normally done by the server.

common logfile format

The common logfile 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 hosts on a network to associate standard IP addresses (such as 198.93.93.10) with hostnames (such as www.netscape.com). Machines 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. Machines 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 host called `realthing.`*yourdomain*`.`*domain,* where the server currently exists.

document root

A directory on the server host that contains the files, images, and data you want to present to users accessing the server.

EMACS

A text editor that can also be used to read email and news.

expires header

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

fancy indexing

Fancy indexing provides more information than simple indexing. Fancy indexing displays a list of contents by name with file size, last modification date, and an icon reflecting file type. Because of this, fancy indexes might take longer than simple indexes for the client to load.

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 (*.GIF* or *.HTML*).

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.

hard restart

Terminating the process, and starting it up again.

home page

A document that exists on the server and acts as a catalog or entry point for the server's contents. The location of this document is defined within the server's configuration files.

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 host 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 *httpd*.

HTTPS

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

imagemap

A process that makes areas of an image active, letting users navigate and obtain information by clicking the different regions of the image with a mouse. Imagemap can also refer to a CGI program called "imagemap," which is used to handle imagemap functionality in other *httpd* implementations.

IP address

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

ISINDEX

Documents can often use a network navigator's capabilities to accept a search string and send it to the server to access a searchable index without using forms. In order to use ISINDEX, you must create a query handler.

ISMAP

An extension to the IMG SRC tag used in an HTML document to tell the server that the named image is an imagemap.

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, that is unique with high probability, and 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

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

NIS

Network Information Service. This is a system of programs and data files that local area networks use to collect, collate, and share specific information about hosts, 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 IRIX hosts that store IRIX user login names, passwords, and user ID numbers. It is also known as */etc/passwd*, because of where it is kept.

public information directories

Directories not inside the document root that are in an IRIX user's home directory, or directories that are under the user's control.

RAM

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

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.

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 hosts. The root user has complete access privileges to all files on the host.

ScriptAlias

A method that NCSA *httpd* used for some of its configuration, including directory remapping and CGI activation.

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 host dedicated to holding the server program, configuration, maintenance, and information files.

simple index

The opposite of fancy indexing—this type of directory listing displays only the names of the files without any graphical elements.

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 HUP signal (signal number one) to the process. 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.

strftime

A function that converts a date and a time to a string. It's used by the server when appending trailers. `strftime` has a special format language for the date and time that the server can use in a trailer to illustrate a file's last modified date.

superuser

The most privileged user available on IRIX hosts (also called root). The superuser has complete access privileges to all files on the host.

sym-links

Abbreviation for symbolic links, which is a type of redirection used by the

IRIX operating system. Sym-links let you create a pointer from one part of your filesystem to an existing file or directory on another part of the filesystem.

telnet

A protocol where two hosts 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(1)

A program on IRIX systems that shows the current state of system resource usage. (See also the *gr_top*(1), *gr_osview*(1), and *ps*(1) reference pages.)

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 IRIX user on a host.

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*://*host*[:*port*]/*document*

A sample URL is *http://www.sgi.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. You can browse this database 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