## Contents

**About This Guide** ......................................................... 9
- Who Should Use This Guide ............................................. 9
- Using the Documentation ................................................ 10
- How This Guide Is Organized .......................................... 12
- Documentation Conventions .......................................... 13
- Product Support .......................................................... 14

**Chapter 1 Basics of Server Operation** .............................. 15
- Configuration Files ..................................................... 15
  - server.xml .............................................................. 16
  - magnus.conf ........................................................... 16
  - obj.conf ................................................................. 16
  - mime.types ............................................................. 17
  - Other Configuration Files .......................................... 17
- Directory Structure ...................................................... 17
  - All Platforms .......................................................... 17
  - UNIX and Linux Platforms ......................................... 20
  - Dynamic Reconfiguration ........................................... 21

**Chapter 2 Server Configuration Elements in server.xml** ........ 23
- The sun-web-server_6_1.dtd File ....................................... 24
  - Subelements .................................................................. 24
  - Data .............................................................................. 25
  - Attributes ..................................................................... 25
- Elements in the server.xml File ......................................... 26
- Core Server Elements ..................................................... 26
  - SERVER ........................................................................ 26
| Other Important Variables                        | 71 |
| Variable Evaluation                             | 72 |
| Sample server.xml File                          | 72 |

**Chapter 3 Syntax and Use of magnus.conf**  
- Init Functions .............................................. 78
- Server Information ......................................... 78
- Language Issues ............................................. 80
- DNS Lookup .................................................. 81
- Threads, Processes, and Connections ............... 81
- Native Thread Pools ........................................ 88
- CGI .............................................................. 89
- Error Logging and Statistic Collection .............. 91
- ACL ............................................................. 93
- Security ....................................................... 94
- Chunked Encoding ............................................ 96
- Miscellaneous Directives ............................... 97
- Deprecated Directives ..................................... 99

**Summary of Init Functions and Directives in magnus.conf**  
- Init Functions ............................................... 101
- Directives .................................................... 108

**Chapter 4 Predefined SAFs in obj.conf**  
- The bucket Parameter ......................................... 121
- AuthTrans ...................................................... 121
- basic-auth .................................................... 122
- basic-ncsa ..................................................... 124
- get-sslid ...................................................... 125
- match-browser ............................................... 126
- qos-handler ................................................... 127
- set-variable .................................................. 128
- NameTrans ...................................................... 133
- assign-name ................................................... 134
- document-root ............................................... 136
- home-page ..................................................... 137
- match-browser ............................................... 138
- ntrans-dav .................................................... 138
- ntrans-j2ee ................................................... 139
- pfx2dir ......................................................... 139
- redirect ....................................................... 141
- set-variable ................................................... 142
- strip-params .................................................. 143

| Basic-ncsa                        | 124 |
| Basic-ncsa                        | 124 |
| Get-sslid                         | 125 |
| Match-browser                     | 126 |
| Qos-handler                       | 127 |
| Set-variable                      | 128 |
| Redirect                          | 141 |
| Set-variable                      | 142 |
| Strip-params                      | 143 |
unix-home .................................................. 143
PathCheck .................................................. 144
cHECK-acl .................................................. 145
find-compressed ......................................... 146
deny-existence .......................................... 148
find-index ............................................... 149
find-links ............................................... 150
find-pathinfo .......................................... 151
get-client-cert ......................................... 151
load-config ............................................. 153
match-browser .......................................... 156
nt-URI-clean ........................................... 156
ntgicheck ............................................... 157
pcheck-dav ............................................. 157
require-auth ........................................... 158
set-variable ........................................... 159
set-virtual-index ..................................... 159
ssl-check ............................................... 160
ssl-logout ............................................. 161
unix-uri-clean ........................................ 161
ObjectType ............................................... 162
force-type ............................................. 163
match-browser ......................................... 164
set-default-type ....................................... 164
set-variable ........................................... 165
shtml-hacktype ........................................ 166
type-by-exp ............................................ 166
type-by-extension ..................................... 167
Input ..................................................... 168
insert-filter .......................................... 169
match-browser ......................................... 170
remove-filter .......................................... 170
set-variable ........................................... 171
Output .................................................... 171
insert-filter .......................................... 172
match-browser ......................................... 173
remove-filter .......................................... 173
set-variable ........................................... 174
Service ................................................... 174
add-footer ............................................. 177
add-header .............................................. 178
append-trailer ......................................... 180
imagemap ............................................... 181
<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>184</td>
</tr>
<tr>
<td>185</td>
</tr>
<tr>
<td>186</td>
</tr>
<tr>
<td>188</td>
</tr>
<tr>
<td>189</td>
</tr>
<tr>
<td>189</td>
</tr>
<tr>
<td>190</td>
</tr>
<tr>
<td>191</td>
</tr>
<tr>
<td>192</td>
</tr>
<tr>
<td>193</td>
</tr>
<tr>
<td>194</td>
</tr>
<tr>
<td>196</td>
</tr>
<tr>
<td>197</td>
</tr>
<tr>
<td>199</td>
</tr>
<tr>
<td>200</td>
</tr>
<tr>
<td>201</td>
</tr>
<tr>
<td>202</td>
</tr>
<tr>
<td>203</td>
</tr>
<tr>
<td>204</td>
</tr>
<tr>
<td>205</td>
</tr>
<tr>
<td>206</td>
</tr>
<tr>
<td>206</td>
</tr>
<tr>
<td>208</td>
</tr>
<tr>
<td>209</td>
</tr>
<tr>
<td>210</td>
</tr>
<tr>
<td>211</td>
</tr>
<tr>
<td>212</td>
</tr>
<tr>
<td>213</td>
</tr>
<tr>
<td>214</td>
</tr>
<tr>
<td>214</td>
</tr>
<tr>
<td>215</td>
</tr>
<tr>
<td>215</td>
</tr>
<tr>
<td>216</td>
</tr>
<tr>
<td>216</td>
</tr>
<tr>
<td>217</td>
</tr>
<tr>
<td>218</td>
</tr>
<tr>
<td>219</td>
</tr>
<tr>
<td>220</td>
</tr>
</tbody>
</table>

**Chapter 5 MIME Types**

- Introduction .................................................. 221
- Determining the MIME Type .................................. 222
- How the Type Affects the Response ......................... 222
What Does the Client Do with the MIME Type? ......................................................... 223
Syntax of the MIME Types File ............................................................. 223
Sample MIME Types File ................................................................. 224

Chapter 6 Other Server Configuration Files ......................................................... 227
certmap.conf ................................................................. 227
dbswitch.conf ............................................................... 229
Deployment Descriptors ............................................................. 232
generated.instance.acl ......................................................... 232
login.conf ................................................................. 233
nsfc.conf ................................................................. 233
password.conf ............................................................. 235
server.policy ............................................................. 236
*.clfilter ................................................................. 236

Appendix A Configuration Changes Between iPlanet Web Server 4.1 and Sun ONE Web
Server 6.1 ................................................................. 237
magnus.conf ............................................................. 237
obj.conf ................................................................. 240
contexts.properties ......................................................... 240
rules.properties .......................................................... 242
servlets.properties ......................................................... 242

Appendix B Configuration Changes Between iPlanet Web Server 6.0 and Sun ONE Web
Server 6.1 ................................................................. 245
magnus.conf ............................................................. 245
Init Functions .............................................................. 245
Directives ................................................................. 246
obj.conf ................................................................. 247
server.xml ............................................................. 247

Appendix C Time Formats ................................................................. 251

Appendix D Alphabetical List of Server Configuration Elements ............................. 253

Appendix E Alphabetical List of Predefined SAFs ................................................. 257

Index ................................................................. 263
About This Guide

This guide discusses the purpose and use of the configuration files for Sun™ Open Net Environment (Sun ONE) Web Server 6.1, including server.xml, magnus.conf, and mime.types, and provides comprehensive lists of the elements and directives in these configuration files.

This preface contains information about the following topics:

- Who Should Use This Guide
- Using the Documentation
- How This Guide Is Organized
- Documentation Conventions
- Product Support

Who Should Use This Guide

The intended audience for this guide is the person who administers and maintains the Sun ONE Web Server.

This guide assumes you are familiar with the following topics:

- J2EE specification
- HTTP
- HTML
- XML
- Java programming
- Java APIs as defined in servlet, JSP, and JDBC specifications
Using the Documentation

The Sun ONE Web Server manuals are available as online files in PDF and HTML formats at:

http://docs.sun.com/db/prod/s1websrv#hic

The following table lists the tasks and concepts described in the Sun ONE Web Server manuals.

Table 1  Sun ONE Web Server Documentation Roadmap

<table>
<thead>
<tr>
<th>For Information About</th>
<th>See the Following</th>
</tr>
</thead>
<tbody>
<tr>
<td>Late-breaking information about the software and documentation</td>
<td>Release Notes</td>
</tr>
<tr>
<td>Getting started with Sun ONE Web Server, including hands-on exercises that introduce server basics and features (recommended for first-time users)</td>
<td>Getting Started Guide</td>
</tr>
<tr>
<td>Performing installation and migration tasks:</td>
<td>Installation and Migration Guide</td>
</tr>
<tr>
<td>• Installing Sun ONE Web Server and its various components, supported platforms, and environments</td>
<td></td>
</tr>
<tr>
<td>• Migrating from Sun ONE Web Server 4.1 or 6.0 to Sun ONE Web Server 6.1</td>
<td></td>
</tr>
</tbody>
</table>
Using the Documentation

Performing the following administration tasks:

- Using the Administration and command-line interfaces
- Configuring server preferences
- Using server instances
- Monitoring and logging server activity
- Using certificates and public key cryptography to secure the server
- Configuring access control to secure the server
- Using Java™ 2 Platform, Enterprise Edition (J2EE™ platform) security features
- Deploying applications
- Managing virtual servers
- Defining server workload and sizing the system to meet performance needs
- Searching the contents and attributes of server documents, and creating a text search interface
- Configuring the server for content compression
- Configuring the server for web publishing and content authoring using WebDAV

Using programming technologies and APIs to do the following:

- Extend and modify Sun ONE Web Server
- Dynamically generate content in response to client requests
- Modify the content of the server

Table 1  Sun ONE Web Server Documentation Roadmap

<table>
<thead>
<tr>
<th>For Information About</th>
<th>See the Following</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performing the following administration tasks:</td>
<td>Administrator’s Guide</td>
</tr>
<tr>
<td>• Using the Administration and command-line interfaces</td>
<td></td>
</tr>
<tr>
<td>• Configuring server preferences</td>
<td></td>
</tr>
<tr>
<td>• Using server instances</td>
<td></td>
</tr>
<tr>
<td>• Monitoring and logging server activity</td>
<td></td>
</tr>
<tr>
<td>• Using certificates and public key cryptography to secure the server</td>
<td></td>
</tr>
<tr>
<td>• Configuring access control to secure the server</td>
<td></td>
</tr>
<tr>
<td>• Using Java™ 2 Platform, Enterprise Edition (J2EE™ platform) security features</td>
<td></td>
</tr>
<tr>
<td>• Deploying applications</td>
<td></td>
</tr>
<tr>
<td>• Managing virtual servers</td>
<td></td>
</tr>
<tr>
<td>• Defining server workload and sizing the system to meet performance needs</td>
<td></td>
</tr>
<tr>
<td>• Searching the contents and attributes of server documents, and creating a text search interface</td>
<td></td>
</tr>
<tr>
<td>• Configuring the server for content compression</td>
<td></td>
</tr>
<tr>
<td>• Configuring the server for web publishing and content authoring using WebDAV</td>
<td></td>
</tr>
<tr>
<td>Using programming technologies and APIs to do the following:</td>
<td>Programmer’s Guide</td>
</tr>
<tr>
<td>• Extend and modify Sun ONE Web Server</td>
<td></td>
</tr>
<tr>
<td>• Dynamically generate content in response to client requests</td>
<td></td>
</tr>
<tr>
<td>• Modify the content of the server</td>
<td></td>
</tr>
</tbody>
</table>
How This Guide Is Organized

This guide has the following chapters:

- **Chapter 1, “Basics of Server Operation”**
  This chapter introduces the major configuration files that control the Sun ONE Web Server and describes how to activate and edit them.

- **Chapter 2, “Server Configuration Elements in server.xml”**
  This chapter discusses the server.xml file, which controls most aspects of server operation.

- **Chapter 3, “Syntax and Use of magnus.conf”**
  This chapter discusses the directives you can set in the magnus.conf file to configure the Sun ONE Web Server during initialization.

- **Chapter 4, “Predefined SAFs in obj.conf”**
  This chapter describes the predefined SAFs used in the obj.conf file.

- **Chapter 5, “MIME Types”**
  This chapter discusses the MIME types file, which maps file extensions to file types.

- **Chapter 6, “Other Server Configuration Files”**
  This chapter lists other important configuration files and provides a quick reference of their contents.
• **Appendix A, “Configuration Changes Between iPlanet Web Server 4.1 and Sun ONE Web Server 6.1”**
  This appendix describes the changes in configuration files between the 4.x and 6.1 versions of Sun ONE Web Server.

• **Appendix B, “Configuration Changes Between iPlanet Web Server 6.0 and Sun ONE Web Server 6.1”**
  This appendix describes the changes in configuration files between the 6.0 and 6.1 versions of Sun ONE Web Server.

• **Appendix C, “Time Formats”**
  This appendix describes the format strings used for dates and times in the server log.

• **Appendix D, “Alphabetical List of Server Configuration Elements”**
  This chapter provides an alphabetical list for easy lookup of elements in server.xml and directives in magnus.conf.

• **Appendix E, “Alphabetical List of Predefined SAFs”**
  This chapter provides an alphabetical list for easy lookup of directives in obj.conf.

---

**Documentation Conventions**

This section describes the types of conventions used throughout this guide:

• **File and directory paths** are given in UNIX® format (with forward slashes separating directory names). For Windows versions, the directory paths are the same, except that backslashes are used to separate directories.

• **URLs** are given in the format:
  
  http://server.domain/path/file.html

  In these URLs, server is the server name where applications are run; domain is your Internet domain name; path is the server’s directory structure; and file is an individual filename. Italic items in URLs are placeholders.

• **Font conventions** include:
  
  - The monospace font is used for sample code and code listings, API and language elements (such as function names and class names), file names, pathnames, directory names, and HTML tags.
产品支持

- **斜体**类型用于代码变量。
- **斜体**类型也用于书籍标题、强调、变量和占位符，以及在字面意义中使用的单词。
- **粗体**类型用于作为段落的前导，或者表示在字面意义中使用的单词。

- **安装根目录**由`install_dir`表示。

默认情况下，**安装根目录**在UNIX基板上位于`/opt/SUNWwbsvr/`。

在Windows上，它位于`C:\Sun\WebServer6.1`。

产品支持

如果您的系统出现问题，请通过以下方式联系客户服务：

- 网上支持网站位于：
  

- 与您的维护合同关联的电话转派号

在联系支持之前，请提供以下信息，以确保我们的支持人员能够帮助您解决这些问题：

- 问题的描述，包括问题发生的环境及其对操作的影响
- 机器类型、操作系统版本和产品版本，包括任何可能影响问题的补丁或软件
- 重现问题的方法的详细步骤
- 错误日志或核心转储
Chapter 1 Basics of Server Operation

The configuration and behavior of Sun ONE Web Server is determined by a set of configuration files. When you use the Administration interface, you change the settings in these configuration files. You can also manually edit these files.

This chapter has the following sections:

- **Configuration Files**
- **Directory Structure**
- **Dynamic Reconfiguration**

### Configuration Files

The configuration and operation of the Sun ONE Web Server is controlled by configuration files. The configuration files reside in the directory `instance_dir/config`. This directory contains various configuration files for controlling different components. The exact number and names of configuration files depends on which components have been enabled or loaded into the server.

However, this directory always contains four configuration files that are essential for the server to operate. These files are:

- **server.xml** -- contains most of the server configuration.
- **magnus.conf** -- contains global server initialization information.
- **obj.conf** -- contains instructions for handling HTTP requests from clients.
Configuration Files

- **mime.types** -- contains information for determining the content type of requested resources.

**server.xml**

This file contains most of the server configuration. A schema file, `sun-web-server_6_1.dtd`, defines its format and content.

For more information about how the server uses `sun-web-server_6_1.dtd` and `server.xml`, see Chapter 2, “Server Configuration Elements in server.xml”.

**magnus.conf**

This file sets values of variables that configure the server during initialization. The server looks at this file and executes the settings on startup. The server does not look at this file again until it is restarted.

See Chapter 3, “Syntax and Use of magnus.conf” for a list of all the variables and Init directives that can be set in `magnus.conf`.

**obj.conf**

This file contains instructions for the Sun ONE Web Server about how to handle HTTP requests from clients and service web server content such as native server plugins and CGI programs. The server looks at the configuration defined by this file every time it processes a request from a client.

This file contains a series of instructions (directives) that tell the Sun ONE Web Server what to do at each stage in the request-response process. You can modify and extend the request handling process by adding or changing the instructions in `obj.conf`.

All `obj.conf` files are located in the `server_root/config` directory. There is one `obj.conf` file for each virtual server class. Whenever this guide refers to "the `obj.conf` file," it refers to all `obj.conf` files or to the `obj.conf` file for the virtual server class being described.

By default, each active `obj.conf` file is named `vs_class-obj.conf`. Editing one of these files directly or through the Administration interface changes the configuration of a virtual server class.
The `obj.conf` file is essential to the operation of the Sun ONE Web Server. When you make changes to the server through the Administration interface, the system automatically updates `obj.conf`.

For information about how the server uses `obj.conf`, see Chapter 4, “Predefined SAFs in `obj.conf`”.

**mime.types**

This file maps file extensions to MIME types to enable the server to determine the content type of a requested resource. For example, requests for resources with `.html` extensions indicate that the client is requesting an HTML file, while requests for resources with `.gif` extensions indicate that the client is requesting an image file in GIF format.

For more information about how the server uses `mime.types`, see "MIME Types."

**Other Configuration Files**

For information about other important configuration files, see Chapter 6, “Other Server Configuration Files”.

**Directory Structure**

The following section describes the directory structure created when you first install Sun ONE Web Server 6.1. The information is organized in two parts:

- All Platforms
- UNIX and Linux Platforms

**All Platforms**

For all platforms, the following directories are created under the server root directory:

- alias contains the key and certificate files for all Sun ONE servers (for example, `https-admserv-server_id-cert8.db` and `secmod.db`).
• **bin** contains the binary files for the server, such as the actual server, the Administration Server forms, and so on. In addition, this directory includes the https/install folder that contains files needed for migrating server settings and default configuration files needed for backward compatibility.

• **docs** is the server's default primary document directory, where your server's content files are usually kept. If you are migrating settings from an existing server, this directory doesn't appear until you finish the migration process.

• **extras** contains the log analyzer and log analysis tools.
  - The flexanlg directory contains a command-line log analyzer. This log analyzer analyzes files in flexlog format.
  - The log_anly directory contains the log analysis tool that runs through the Server Manager. This log analyzer analyzes files in common log format only.

• **httpacl** contains the files that store access control configuration information in the generated.server-id.acl and genwork.server-id.acl files. The file generated.server-id.acl contains changes you make using the Server Manager access control forms after saving your changes; genwork.server-id.acl contains your changes before you save your changes.

• **https-admserv** contains the directories for the Administration Server. This directory has the following subdirectories and files:
  - For UNIX/Linux platforms, this directory contains shell scripts to start, stop, and restart the server and a script to rotate log files.
  - ClassCache contains classes and Java files, generated as result of the compilation of JavaServer pages.
  - conf_bk contains backup copies of the administration server's configuration files.
  - config contains the server's configuration files.
  - logs contains any error or access log files.
  - SessionData contains session database data from MMapSessionManager.
  - startsvr.bat is the script that starts the Server Manager on Windows machines. The Server Manager lets you configure all servers installed in the server root directory.
  - stopsvr.bat is the script that stops the Server Manager on Windows machines.
**https-server_id** are the directories for each server you have installed on the machine. Each server directory has the following subdirectories and files:

- ClassCache contains classes and Java files, generated as result of the compilation of JavaServer pages.
- conf_bk contains backup copies of the server’s configuration files.
- config contains the server instance configuration files.
- logs contains the server instance log files.
- reconfig is the script used to reconfigure the server dynamically. If you make non-global changes to the server, you can use this script to reconfigure the server without stopping and starting it. Note that changes to ACL files and magnus.conf require you to stop and restart the server.
- restart is the script that restarts the server.
- rotate rotates server log files without affecting users who may be connected to the server.
- search contains the following directories: admin and collections
- SessionData contains session database data from MMapSessionManager.
- startsvr.bat is the script that starts the Server Manager. The Server Manager lets you configure all servers installed in the server root directory.
- stopsvr.bat is the script that stops the Server Manager.

**manual** contains the online manuals for the product.

**plugins** contains directories for Java, search, and other plugins. This directory has the following subdirectories:

- .htaccess contains server plugin for .htaccess access control and htconvert, an .nsconfig to .htaccess converter.
- digest contains the Digest Authentication Plugin for Sun ONE Directory Server 5.0, as well as information about the plugin.
- samples contains samples and example components, plugins and technologies supported by the Sun ONE Web Server servlet engine. This includes binaries, all code, and a build environment.
- servlets contains information about and examples of web-apps applications.
- include contains various include files.
Directory Structure

- **lib** contains shared libraries.
- **nsacl** contains information for your server’s access control lists.
- **loadbal** contains the required files for the Resonate load-balancer integration plugin.
- **nsapi** contains header files and example code for creating your own functions using NSAPI. For more information, see the Sun ONE documentation web site at: http://docs.iplanet.com/docs/manuals/enterprise.html.
- **search** contains information for your server’s search plugins.
- **snmp** contains information for your server’s SNMP plugins.
- **setup** contains the various Sun ONE Web Server setup files, including setup.log and uninstall.inf.
- **userdb** contains user databases and related information.
- **LICENSE.txt** is the license file.
- **README.txt** is the readme file that contains a link to the Sun ONE Web Server Release Notes.

**UNIX and Linux Platforms**

In addition to the files and directories described in “All Platforms,” the following files are created at the server-root directory for UNIX and Linux platforms:

- **startconsole** launches a browser to the Administration Server page.

The following files are created under the server-root/https-admserv directory for UNIX and Linux platforms:

- **ClassCache** contains classes and Java files, generated as result of the compilation of JavaServer pages.
- **conf_bk** contains backup copies of the server’s configuration files.
- **config** contains the Administration Server configuration files.
- **logs** contains the Administration Server log files.
- **SessionData** contains session database data from MMapSessionManager.
- **restart** is the script that restarts the Server Manager.
Dynamic Reconfiguration

Dynamic reconfiguration allows you to make configuration changes to a live web server without having to stop and restart the web server for the changes to take effect. You can dynamically change all configuration settings and attributes in server.xml and its associated files without restarting the server.

To access the dynamic reconfiguration screen and install a new configuration dynamically, click the Apply link found in the upper right corner of the Server Manager, Class Manager, and Virtual Server Manager pages, then click the Load Configuration Files button on the Apply Changes page. If there are errors in installing the new configuration, the previous configuration is restored.
Chapter 2 Server Configuration Elements in server.xml

The server.xml file contains most of the server configuration. The encoding is UTF-8 to maintain compatibility with regular UNIX text editors. The server.xml file is located in the instance_dir/config directory. A schema file, sun-web-server_6_1.dtd, determines the format and content of the server.xml file.

This chapter describes server.xml and sun-server_1_0.dtd in the following sections:

- The sun-web-server_6_1.dtd File
- Elements in the server.xml File
- Core Server Elements
- Listener Elements
- WebDAV Elements
- Search Elements
- Web Application Elements
- Java Configuration Elements
- Resource Elements
- LOG
- User Database Selection
- The Sun ONE LDAP Schema
Variables

Sample server.xml File

The sun-web-server_6_1.dtd File

The sun-web-server_6_1.dtd file defines the structure of the server.xml file, including the elements it can contain and the subelements and attributes these elements can have. The sun-web-server_6_1.dtd file is located in the install_dir/bin/https/dtds directory.

Each element defined in a DTD file (which may be present in the corresponding XML file) can contain the following:

- Subelements
- Data
- Attributes

Subelements

Elements can contain subelements. For example, the following file fragment defines the VSCLASS element.

```xml
<!ELEMENT VSCLASS (VARS?, VS*, QOSPARAMS?)>
```

The ELEMENT tag specifies that a VSCLASS element can contain VARS, VS, and QOSPARAMS elements in that order.

The following table shows how optional suffix characters of subelements determine the requirement rules, or number of allowed occurrences, for the subelements.

<table>
<thead>
<tr>
<th>Subelement Suffix</th>
<th>Requirement Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>element*</td>
<td>Can contain zero or more of this subelement.</td>
</tr>
<tr>
<td>element?</td>
<td>Can contain zero or one of this subelement.</td>
</tr>
<tr>
<td>element+</td>
<td>Must contain one or more of this subelement.</td>
</tr>
<tr>
<td>element (no suffix)</td>
<td>Must contain only one of this subelement.</td>
</tr>
</tbody>
</table>
If an element cannot contain other elements, you see EMPTY or (#PCDATA) instead of a list of element names in parentheses.

**Data**

Some elements contain character data instead of subelements. These elements have definitions of the following format:

```
<!ELEMENT element-name (#PCDATA)>
```

For example:

```
<!ELEMENT DESCRIPTION (#PCDATA)>
```

In the server.xml file, white space is treated as part of the data in a data element. Therefore, there should be no extra white space before or after the data delimited by a data element. For example:

```
<DESCRIPTION>myserver</DESCRIPTION>
```

**Attributes**

Elements that have ATTLIST tags contain attributes (name-value pairs). For example:

```
<!ATTLIST JDBCRESOURCE jndiname CDATA #REQUIRED
poolname CDATA #REQUIRED
enabled %boolean; "true">
```

A JDBCRESOURCE element can contain jndiname, poolname, and enabled attributes.

The #REQUIRED label means that a value must be supplied. The #IMPLIED label means that the attribute is optional, and that Sun ONE Web Server generates a default value. Wherever possible, explicit defaults for optional attributes (such as "true") are listed.

Attribute declarations specify the type of the attribute. For example, CDATA means character data, and %boolean is a predefined enumeration.
Elements in the server.xml File

This section describes the XML elements in the server.xml file. Elements are grouped as follows:

- Core Server Elements
- Listener Elements
- WebDAV Elements
- Search Elements
- Web Application Elements
- Java Configuration Elements
- Resource Elements

**NOTE**

Subelements must be defined in the order in which they are listed under each Subelements heading unless otherwise noted.

For an alphabetical listing of elements in server.xml, see “Alphabetical List of Server Configuration Elements.”

Core Server Elements

General elements are as follows:

- SERVER
- PROPERTY
- DESCRIPTION
- VARS

**SERVER**

Defines a server. This is the root element; there can only be one server element in a server.xml file.
Subelements

The following table describes subelements for the `SERVER` element.

<table>
<thead>
<tr>
<th>Table 2-2</th>
<th>SERVER subelements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
<td><strong>Required</strong></td>
</tr>
<tr>
<td>VARS</td>
<td>zero or one</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>zero or more</td>
</tr>
<tr>
<td>LS</td>
<td>one or more</td>
</tr>
<tr>
<td>MIME</td>
<td>zero or more</td>
</tr>
<tr>
<td>ACLFILE</td>
<td>zero or more</td>
</tr>
<tr>
<td>VSCLASS</td>
<td>one or more</td>
</tr>
<tr>
<td>QOSPARAMS</td>
<td>zero or one</td>
</tr>
<tr>
<td>JAVA</td>
<td>zero or one</td>
</tr>
<tr>
<td>LOG</td>
<td>zero or one</td>
</tr>
</tbody>
</table>

Attributes

The following table describes attributes for the `SERVER` element.

<table>
<thead>
<tr>
<th>Table 2-3</th>
<th>SERVER attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attribute</strong></td>
<td><strong>Default</strong></td>
</tr>
<tr>
<td>qosactive</td>
<td>no</td>
</tr>
<tr>
<td>qosmetricsinterval</td>
<td>30</td>
</tr>
<tr>
<td>qosrecomputeinterval</td>
<td>100</td>
</tr>
</tbody>
</table>
PROPERTY

Specifies a property, or a variable that is defined in server.xml and referenced in obj.conf. For information about variables, see "Variables."

For a list of variables commonly defined in server.xml, see "Variables Used in the Interface."

A property adds configuration information to its parent element that is one or both of the following:

- Optional with respect to Sun ONE Web Server
- Needed by a system or object that Sun ONE Web Server doesn’t have knowledge of, such as an LDAP server or a Java class

For example, an AUTHREALM element can include PROPERTY subelements:

```xml
<AUTHREALM name="file"
    classname="com.iplanet.ias.security.auth.realm.file.FileRealm">
    <PROPERTY name="file" value="instance_dir/config/keyfile"/>
    <PROPERTY name="jaas-context" value="fileRealm"/>
</AUTHREALM>
```

Which properties an AUTHREALM element uses depends on the value of the AUTHREALM element's name attribute. The file realm uses file and jaas-context properties. Other realms use different properties.

Subelements

The following table describes subelements for the PROPERTY element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIPTION</td>
<td>zero or one</td>
<td>Contains a text description of the property.</td>
</tr>
</tbody>
</table>

Attributes

The following table describes attributes for the PROPERTY element.
### Listener Elements

The Listener elements are as follows:

- **LS**
- **SSLPARAMS**

### Table 2-5  PROPERTY attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>none</td>
<td>Specifies the name of the property or variable.</td>
</tr>
<tr>
<td>value</td>
<td>none</td>
<td>Specifies the value of the property or variable.</td>
</tr>
</tbody>
</table>

### DESCRIPTION

Contains a text description of the parent element.

**Subelements**

- None

**Attributes**

- None

### VARS

Defines variables that can be given values in `server.xml` and referenced in `obj.conf`. For more information, see “Variables” on page 70.

**Subelements**

- None

**Attributes**

- None
Listener Elements

- MIME
- ACLFILE
- VSCLASS
- VS
- QOSPARAMS
- USERDB

**LS**

Defines an HTTP listen socket.

**NOTE**

When you create a secure listen socket through the Server Manager, security is automatically turned on globally in `magnus.conf`. When you create a secure listen socket manually in `server.xml`, security must be turned on by editing `magnus.conf`.

The `CONNECTIONGROUP` element from the schema file for `server.xml` in version 6.0 of Web Server is no longer supported. Its attributes and the subelement `SSLPARAMS` are added to the `LS` element in Sun ONE Web Server 6.1.

### Subelements

The following table describes subelements for the `LS` element.

<table>
<thead>
<tr>
<th>Table 2-6 LS subelements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
</tr>
<tr>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>SSLPARAMS</td>
</tr>
</tbody>
</table>

### Attributes

The following table describes attributes for the `LS` element.
### Table 2-7  LS attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>none</td>
<td>(optional) The socket family type. A socket family type cannot begin with a number.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When you create a secure listen socket in the server.xml file, security must be turned on in magnus.conf. When you create a secure listen socket in the Server Manager, security is automatically turned on globally in magnus.conf.</td>
</tr>
<tr>
<td>ip</td>
<td>any</td>
<td>Specifies the IP address of the listen socket. Can be in dotted-pair or IPv6 notation. Can also be any for INADDR_ANY.</td>
</tr>
<tr>
<td>port</td>
<td>none</td>
<td>Port number to create the listen socket on. Legal values are 1 - 65535. On UNIX, creating sockets that listen on ports 1 - 1024 requires superuser privileges. Configuring an SSL listen socket to listen on port 443 is recommended. Two different IP addresses can’t use the same port.</td>
</tr>
<tr>
<td>security</td>
<td>false</td>
<td>(optional) Determines whether the listen socket runs SSL. Legal values are on, off, yes, no, 1, 0, true, false. You can turn SSL2 or SSL3 on or off and set ciphers using an SSLPARAMS subelement for this listen socket.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Security setting in the magnus.conf file globally enables or disables SSL by making certificates available to the server instance. Therefore, Security in magnus.conf must be on or security in server.xml does not work. For more information, see Chapter 3, “Syntax and Use of magnus.conf.”</td>
</tr>
<tr>
<td>acceptorthreads</td>
<td>1</td>
<td>(optional) Number of acceptor threads for the listener. The recommended value is the number of processors in the machine. Legal values are 1 - 1024.</td>
</tr>
</tbody>
</table>
**SSLPARAMS**

Defines SSL (Secure Socket Layer) parameters.

Subelements

none

Attributes

The following table describes attributes for the `SSLPARAMS` element.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>family</td>
<td>none</td>
<td>(optional) The socket family type. Legal values are <code>inet</code>, <code>inet6</code>, and <code>nca</code>. Use the value <code>inet6</code> for IPv6 listen sockets. When using the value of <code>inet6</code>, IPv4 addresses will be prefixed with <code>::ffff:</code> in the log file. Specify <code>nca</code> to make use of the Solaris Network Cache and Accelerator.</td>
</tr>
<tr>
<td>blocking</td>
<td>false</td>
<td>(optional) Determines whether the listen socket and the accepted socket are put in to blocking mode. Use of blocking mode may improve benchmark scores. Legal values are on, off, yes, no, 1, 0, true, false.</td>
</tr>
<tr>
<td>defaultvs</td>
<td>none</td>
<td>The id attribute of the default virtual server for this particular listen socket.</td>
</tr>
<tr>
<td>servername</td>
<td>none</td>
<td>Tells the server what to put in the host name section of any URLs it sends to the client. This affects URLs the server automatically generates; it doesn’t affect the URLs for directories and files stored in the server. This name should be the alias name if your server uses an alias. If you append a colon and port number, that port will be used in URLs the server sends to the client.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>servercertnickname</td>
<td>Server-Cert</td>
<td>The nickname of the server certificate in the certificate database or the PKCS#11 token. In the certificate, the name format is tokenname:nickname. Including the tokenname: part of the name in this attribute is optional.</td>
</tr>
<tr>
<td>ssl2</td>
<td>false</td>
<td>(optional) Determines whether SSL2 is enabled. Legal values are on, off, yes, no, 1, 0, true, and false. If both SSL2 and SSL3 are enabled for a virtual server, the server tries SSL3 encryption first. If that fails, the server tries SSL2 encryption.</td>
</tr>
<tr>
<td>ssl2ciphers</td>
<td>none</td>
<td>(optional) A space-separated list of the SSL2 ciphers used, with the prefix + to enable or - to disable, for example +rc4. Allowed values are rc4, rc4export, rc2, rc2export, idea, des, desede3.</td>
</tr>
<tr>
<td>ssl3</td>
<td>true</td>
<td>(optional) Determines whether SSL3 is enabled. Legal values are on, off, yes, no, 1, 0, true and false. If both SSL2 and SSL3 are enabled for a virtual server, the server tries SSL3 encryption first. If that fails, the server tries SSL2 encryption.</td>
</tr>
<tr>
<td>ssl3tlsciphers</td>
<td>none</td>
<td>(optional) A space-separated list of the SSL3 ciphers used, with the prefix + to enable or - to disable, for example +rsa_des_sha. Allowed SSL3 values are rsa_rc4_128_md5, rsa_3des_sha, rsa_des_sha, rsa_rc4_40_md5, rsa_rc2_40_md5, rsa_null_md5. Allowed TLS values are rsa_des_56_sha, rsa_rc4_56_sha.</td>
</tr>
<tr>
<td>tls</td>
<td>true</td>
<td>(optional) Determines whether TLS is enabled. Legal values are on, off, yes, no, 1, 0, true, and false.</td>
</tr>
</tbody>
</table>

Table 2-8  SSLPARAMS attributes
MIME

Defines MIME types.

The most common way that the server determines the MIME type of a requested resource is by invoking the `type-by-extension` directive in the `ObjectType` section of the `obj.conf` file. The `type-by-extension` function does not work if no `mime` element has been defined in the `SERVER` element.

Attributes

The following table describes attributes for the `MIME` element.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>none</td>
<td>Internal name for the MIME types listing. Used in a <code>VS</code> element to define the MIME types used by the virtual server. The MIME types name cannot begin with a number.</td>
</tr>
<tr>
<td>file</td>
<td>none</td>
<td>The name of a MIME types file. For more information, see “MIME Types” on page 221.</td>
</tr>
</tbody>
</table>
ACLFILE

References one or more ACL files.

Subelements

The following table describes subelements for the ACLFILE element.

| Table 2-10 ACLFILE subelements |
|-------------------|------------------|------------------|
| Element           | Required         | Description      |
| DESCRIPTION       | zero or one      | Contains a text description of the ACLFILE element. |

Attributes

The following table describes attributes for the ACLFILE element.

| Table 2-11 ACLFILE attributes |
|-------------------------------|------------------|------------------|
| Attribute        | Default | Description                  |
| id              | none    | Internal name for the ACL file listing. Used in a VS element to define the ACL file used by the virtual server. An ACL file listing name cannot begin with a number. |
| file            | none    | A space-separated list of ACL files. Each ACL file must have a unique name. For information about the format of an ACL file, see the Sun ONE Web Server 6.1 Administrator’s Guide. The name of the default ACL file is generated.https-server_id.acl, and the file resides in the server_root/server_id/httpacl directory. To use this file, you must reference it in server.xml. |

VSCLASS

Defines a virtual server class.
Subelements
The following table describes subelements for the VSCLASS element.

**Table 2-12** VSCLASS subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIPTION</td>
<td>zero or one</td>
<td>Contains a text description of the VSCLASS.</td>
</tr>
<tr>
<td>VARS</td>
<td>zero or one</td>
<td>Specifies a property of the VSCLASS.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>zero or more</td>
<td>Specifies a property of the VSCLASS.</td>
</tr>
<tr>
<td>VS</td>
<td>zero or more</td>
<td>Defines a virtual server.</td>
</tr>
<tr>
<td>QOSPARAMS</td>
<td>zero or one</td>
<td>Defines quality of service parameters.</td>
</tr>
</tbody>
</table>

Attributes
The following table describes attributes for the VSCLASS element.

**Table 2-13** VSCLASS attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>none</td>
<td>Virtual server class ID. This is a unique ID that allows lookup of a specific virtual server class. A virtual server class ID cannot begin with a number.</td>
</tr>
<tr>
<td>objectfile</td>
<td>obj.conf</td>
<td>The obj.conf file for this class of virtual servers. Cannot be overridden in a VS element.</td>
</tr>
</tbody>
</table>
VS

Defines a virtual server. A virtual server, also called a virtual host, is a virtual web server that serves content targeted for a specific URL. Multiple virtual servers may serve content using the same or different host names, port numbers, or IP addresses. The HTTP service can direct incoming web requests to different virtual servers based on the URL.

Subelements

The following table describes subelements for the vs element.

Table 2-13  VSCLASS attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rootobject</td>
<td>default</td>
<td>(optional) Tells the server which object loaded from an obj.conf file is the default. The default object is expected to have all the name translation (NameTrans) directives for the virtual 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 Server Manager assumes the default to be the object named default. Don’t deviate from this convention if you use (or plan to use) the Server Manager.</td>
</tr>
<tr>
<td>acceptlanguage</td>
<td>false</td>
<td>(optional) If true, the server parses the Accept-Language header and sends an appropriate language version based on which language the client can accept. You should set this value to on only if the server supports multiple languages. Can be overridden in a VS element. Legal values are on, off, yes, no, 1, 0, true, and false.</td>
</tr>
</tbody>
</table>
Listener Elements

**Table 2-14  VS subelements**

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIPTION</td>
<td>zero or one</td>
<td>Contains a text description of this element.</td>
</tr>
<tr>
<td>VARS</td>
<td>zero or one</td>
<td>Specifies a property or a variable of the VS.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>zero or more</td>
<td>Specifies a property or a variable of the VS.</td>
</tr>
<tr>
<td>QOSPARAMS</td>
<td>zero or one</td>
<td>Defines quality of service parameters.</td>
</tr>
<tr>
<td>USERDB</td>
<td>zero or more</td>
<td>Defines the user database for the virtual server.</td>
</tr>
<tr>
<td>DAV</td>
<td>zero or one</td>
<td>Defines the WebDAV configuration for the virtual server.</td>
</tr>
<tr>
<td>SEARCH</td>
<td>zero or one</td>
<td>Defines the search configuration for the virtual server.</td>
</tr>
<tr>
<td>WEBAPP</td>
<td>zero or more</td>
<td>Specifies a web application.</td>
</tr>
</tbody>
</table>

**Attributes**

The following table describes attributes for the `vs` element.

**Table 2-15  VS attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>none</td>
<td>Virtual server ID. This is a unique ID that allows lookup of a specific virtual server. Can also be referred to as the variable <code>$id</code> in an <code>obj.conf</code> file. A virtual server ID cannot begin with a number.</td>
</tr>
<tr>
<td>connections</td>
<td>none</td>
<td>(optional) A space-separated list of LS ids that specify the connection(s) the virtual server uses. Required only for a VS that is not the default vs of a listen socket.</td>
</tr>
<tr>
<td>urlhosts</td>
<td>none</td>
<td>A space-separated list of values allowed in the Host request header to select the current virtual server. Each VS that is configured to the same listen socket must have a unique urlhosts value for that group.</td>
</tr>
</tbody>
</table>
### Table 2-15  VS attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objectfile</td>
<td>objectfile of the enclosing VSCLASS</td>
<td>(optional) The file name of the obj.conf file for this virtual server.</td>
</tr>
<tr>
<td>rootobject</td>
<td>default</td>
<td>(optional) Tells the server which object loaded from an obj.conf file is the default.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tells the server which object loaded from an obj.conf file is the default. The default object is expected to have all the name translation (NameTrans) directives for the virtual 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.</td>
</tr>
<tr>
<td>mime</td>
<td>none</td>
<td>The id of the MIME element used by the virtual server.</td>
</tr>
<tr>
<td>aclids</td>
<td>none</td>
<td>(optional) One or more id attributes of ACLFILE elements, separated by commas. Specifies the ACL file(s) used by the virtual server.</td>
</tr>
<tr>
<td>errorlog</td>
<td>none</td>
<td>(optional) Specifies a log file for virtual-server-specific error messages. See the LOG description for details about logs.</td>
</tr>
<tr>
<td>acceptlanguage</td>
<td>off</td>
<td>(optional) If true, the server parses the Accept-Language header and sends an appropriate language version based on which language the client can accept. You should set this value to on only if the server supports multiple languages. Legal values are on, off, yes, no, 1, 0, true, false.</td>
</tr>
<tr>
<td>state</td>
<td>on</td>
<td>(optional) Determines whether a virtual-server is active (on) or inactive (off, disabled). The default is on (active). When inactive, a virtual server does not service requests. If a virtual server is disabled, only the global server administrator can turn it on.</td>
</tr>
</tbody>
</table>
**QOSPARAMS**

Defines quality of service parameters of an `SERVER, VSCLASS, or VS` element.

**Subelements**

none

**Attributes**

The following table describes attributes for the `QOSPARAMS` element.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxbps</td>
<td>none</td>
<td>(required if enforcebandwidth is yes) The maximum bandwidth limit for the server, vsclass, or vs in bytes per second.</td>
</tr>
<tr>
<td>enforcebandwidth</td>
<td>false</td>
<td>(optional) Specifies whether the bandwidth limit should be enforced or not. Allowed values are yes, no, true, false, on, off, 1, 0.</td>
</tr>
<tr>
<td>maxconn</td>
<td>none</td>
<td>(required if enforceconnections is yes) The maximum number of concurrent connections for the <code>SERVER, VSCLASS, or VS</code>.</td>
</tr>
<tr>
<td>enforceconnections</td>
<td>false</td>
<td>(optional) Specifies whether the connection limit should be enforced or not. Allowed values are yes, no, true, false, on, off, 1, 0.</td>
</tr>
</tbody>
</table>

**USERDB**

Defines the user database used by the `VS` element.

**Subelements**

The following table describes subelements for the `USERDB` element.
Attributes
The following table describes attributes for the `USERDB` element.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>none</td>
<td>The user database name in the virtual server’s ACL file. A user database name cannot begin with a number.</td>
</tr>
<tr>
<td>database</td>
<td>none</td>
<td>The user database name in the <code>dbswitch.conf</code> file.</td>
</tr>
<tr>
<td>basedn</td>
<td>none</td>
<td>(optional) Overrides the base DN lookup in the <code>dbswitch.conf</code> file. However, the <code>basedn</code> value is still relative to the base DN value from the <code>dbswitch.conf</code> entry.</td>
</tr>
<tr>
<td>certmaps</td>
<td>none</td>
<td>(optional) Specifies which certificate mapped to LDAP entry mappings (defined in <code>certmap.conf</code>) to use. If not present, all mappings are used. All lookups based on mappings in <code>certmap.conf</code> are relative to the final base DN of the VS.</td>
</tr>
</tbody>
</table>

WebDAV Elements
The WebDAV elements are as follows:

- DAV
- DAVCOLLECTION
DAV

Defines the WebDAV (Web-based Distributed Authoring and Versioning) configuration for the `VS` element.

Subelements

The following table describes subelements for the `DAV` element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROPERTY</td>
<td>zero or more</td>
<td>Specifies a property or a variable.</td>
</tr>
<tr>
<td>DAVCOLLECTION</td>
<td>zero or more</td>
<td>Collections for which DAV is enabled.</td>
</tr>
</tbody>
</table>

Attributes

The following table describes attributes for the `DAV` element.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lockdb</td>
<td>server-instance/lock-db/vs</td>
<td>(optional) Specifies the directory where the locking database will be maintained.</td>
</tr>
<tr>
<td>lockdbupdateinterval</td>
<td>0</td>
<td>(optional) specifies the frequency with which the memory representation of the lock database should be synced up to the disk copy of the lock database. The interval is specified in seconds. A value of zero disables the memory representation of the lock database.</td>
</tr>
<tr>
<td>minlocktimeout</td>
<td>none</td>
<td>(optional) Minimum lifetime of a lock in seconds, -1 implies never expires. A value of 0 sets minlocktimeout to infinity.</td>
</tr>
</tbody>
</table>
DAVCOLLECTION

Defines a DAV-enabled collection of documents rooted at a URI; the source of the documents are accessed via a separate URI space.

The DAVCOLLECTION element defines WebDAV functionality for a URI space. The attributes specified on a collection override any virtual server attribute values.

Subelements

The following table describes subelements for the DAVCOLLECTION element.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>propdbupdateinterval</td>
<td>0</td>
<td>(optional) specifies the frequency with which the memory representation of the property database should be synced up to the disk copy of the database. The interval is specified in seconds. A value of zero disables the memory representation of the property database.</td>
</tr>
<tr>
<td>maxpropdbsize</td>
<td>8192</td>
<td>(optional) specifies an upper limit on the total size of the memory representation of the property databases in the collection. When this size is reached, any additional databases accessed in this collection will not have a memory representation.</td>
</tr>
<tr>
<td>maxxmlrequestbodysize</td>
<td>8192</td>
<td>(optional) Maximum size of the XML request body. Needed to prevent potential Denial of Service (DOS) attacks.</td>
</tr>
<tr>
<td>maxpropdepth</td>
<td>1</td>
<td>(optional) The depth of the PROPFIND request. If the request is to a collection, then the depth of the subdirectories included in the response is specified by this attribute. Legal values are 0, 1, and infinity.</td>
</tr>
<tr>
<td>enabled</td>
<td>true</td>
<td>(optional) Specifies if DAV functionality is enabled for a virtual server. Legal values are yes, no, true, false, on, off, 1, 0.</td>
</tr>
</tbody>
</table>
Table 2-21  DAVCOLLECTION subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIPTION</td>
<td>zero or one</td>
<td>Contains a text description of this element.</td>
</tr>
</tbody>
</table>

Attributes

The following table describes attributes for the DAVCOLLECTION element.

Table 2-22  DAVCOLLECTION attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>none</td>
<td>(required) Specifies the URI by which the output content is accessed.</td>
</tr>
<tr>
<td>sourceuri</td>
<td>none</td>
<td>(optional) Specifies the URI by which the source content of the documents are accessed.</td>
</tr>
<tr>
<td>lockdb</td>
<td>lockdb value specified in the DAV element</td>
<td>(optional) Specifies the directory where the locking database will be maintained.</td>
</tr>
<tr>
<td>lockdbupdateinterval</td>
<td>The value specified in the DAV element.</td>
<td>(optional) specifies the frequency with which the memory representation of the lock database should be synced up to the disk copy of the lock database. The interval is specified in seconds. A value of zero disables the memory representation of the lock database.</td>
</tr>
<tr>
<td>minlocktimeout</td>
<td>minlocktimeout attribute value specified in the DAV element</td>
<td>(optional) Minimum lifetime of a lock in seconds, -1 implies never expires, 0 turns locking off.</td>
</tr>
<tr>
<td>propdbupdateinterval</td>
<td>The value specified in the DAV element.</td>
<td>(optional) specifies the frequency with which the memory representation of the property database should be synced up to the disk copy of the database. The interval is specified in seconds. A value of zero disables the memory representation of the property database.</td>
</tr>
</tbody>
</table>
Search Elements

Search elements are as follows:

- SEARCH
- SEARCHCOLLECTION
- DISPLAYNAME

SEARCH

Defines search related configuration parameters for a given VS.

Subelements

The following table describes subelements for the SEARCH element.

<table>
<thead>
<tr>
<th>Table 2-23</th>
<th>SEARCH subelements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Required</td>
</tr>
<tr>
<td>WEBAPP</td>
<td>zero or one</td>
</tr>
</tbody>
</table>
Attributes

The following table describes attributes for the `SEARCH` element.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>maxhits</code></td>
<td><code>none</code></td>
<td>The maximum number of results that will be retrieved by the search engine in a single search.</td>
</tr>
</tbody>
</table>

SEARCHCOLLECTION

Specifies a searchable index of documents called a search collection.

Subelements

The following table describes subelements for the `SEARCHCOLLECTION` element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DISPLAYNAME</code></td>
<td>zero or one</td>
<td>Optional display name that can be used while displaying searchable collections to the end user.</td>
</tr>
<tr>
<td><code>DESCRIPTION</code></td>
<td>zero or one</td>
<td>Contains a text description of the collection.</td>
</tr>
<tr>
<td><code>PROPERTY</code></td>
<td>zero or more</td>
<td>Contains name-value pairs to pass extra configuration information to the search engine.</td>
</tr>
</tbody>
</table>
Attributes
The following table describes attributes for the SEARCHCOLLECTION element.

Table 2-26  SEARCHCOLLECTION attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>none</td>
<td>Specifies unique identifier for this collection. Should be a legal XML ID type.</td>
</tr>
<tr>
<td>path</td>
<td>none</td>
<td>Specifies a file system location for storing search collection meta data.</td>
</tr>
<tr>
<td>uri</td>
<td>none</td>
<td>Specifies a URI for the indexable collection of documents.</td>
</tr>
<tr>
<td>docroot</td>
<td>none</td>
<td>Specifies a file system path for the collection of documents.</td>
</tr>
<tr>
<td>enabled</td>
<td>true</td>
<td>Specifies whether a collection can be searched. Legal values are yes, no, true, false, on, off, 1, and 0.</td>
</tr>
</tbody>
</table>

DISPLAYNAME
Specifies a human-readable name for the collection to be used while displaying the collection to the end user. Example:

<DISPLAYNAME> Omega Manual </DISPLAYNAME>

Subelements
none

Attributes
none

Web Application Elements
The Web application elements are as follows:

- WEBAPP
WEBAPP

Defines a Java web application rooted at a given URI within a VS.

Subelements

The following table describes subelements for the WEBAPP element.

<table>
<thead>
<tr>
<th>Table 2-27 webapp subelements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>DESCRIPTION</td>
</tr>
</tbody>
</table>

Attributes

The following table describes attributes for the WEBAPP element.

<table>
<thead>
<tr>
<th>Table 2-28 WEBAPP attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>uri</td>
</tr>
<tr>
<td>path</td>
</tr>
<tr>
<td>enabled</td>
</tr>
</tbody>
</table>
Java Configuration Elements

The Java configuration elements are as follows:

- **JAVA**
- **JVMOPTIONS**
- **PROFILER**
- **SECURITY**
- **AUTHREALM**

**JAVA**

Defines configurable properties for the integrated Java Virtual Machine (JVM), and for Java-based security and resources.

**Subelements**

The following table describes subelements for the **JAVA** element.

<table>
<thead>
<tr>
<th>Table 2-29 JAVA subelements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
</tr>
<tr>
<td>PROPERTY</td>
</tr>
<tr>
<td>JVMOPTIONS</td>
</tr>
<tr>
<td>PROFILER</td>
</tr>
<tr>
<td>SECURITY</td>
</tr>
<tr>
<td>RESOURCES</td>
</tr>
</tbody>
</table>

**Attributes**

The following table describes attributes for the **JAVA** element.
### Table 2-30  JAVA attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>javahome</td>
<td>&lt;install-root&gt;/bin/https/jdk For SVR 4 package-based installation for Solaris: /usr/java</td>
<td>The path to the directory where the JDK is installed.</td>
</tr>
<tr>
<td>debug</td>
<td>false</td>
<td>(optional) If true, the server starts up in debug mode ready for attachment with a JPDA-based (Java Platform Debugger Architecture-based) debugger. Legal values are on, off, yes, no, true, false, 1, 0.</td>
</tr>
<tr>
<td>debugoptions</td>
<td>-Xdebug -Xrunjdwp:transport=dt_socket, server=y, suspend=n</td>
<td>(optional) Specifies JPDA options. A list of debugging options that you can include is available at: <a href="http://java.sun.com/products/jpda/doc/conninv.html#Invocation">http://java.sun.com/products/jpda/doc/conninv.html#Invocation</a></td>
</tr>
<tr>
<td>classpathprefix</td>
<td>none</td>
<td>(optional) Specifies a prefix for the system classpath. You should only prefix the system classpath if you wish to override system classes, such as the XML parser classes. Use this attribute with caution.</td>
</tr>
<tr>
<td>serverclasspath</td>
<td>none</td>
<td>(optional) Specifies the classpath for the environment from which the server was started. This classpath can be accessed using System.getProperty(&quot;java.class.path&quot;).</td>
</tr>
<tr>
<td>classpathsuffix</td>
<td>none</td>
<td>(optional) Specifies a suffix for the system classpath.</td>
</tr>
</tbody>
</table>
### Table 2-30  JAVA attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nativelibrarypathp</code></td>
<td>none</td>
<td>(optional) Specifies a prefix for the native library path. The native library path is the automatically constructed concatenation of the path to the server's native shared libraries, the standard JRE (Java Runtime Environment) native library path, the shell environment setting (LD_LIBRARY_PATH on UNIX), and any path specified in the <code>profiler</code> element. Since this is synthesized, it does not appear explicitly in the server configuration.</td>
</tr>
<tr>
<td><code>refix</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>nativelibrarypathsuffix</code></td>
<td>none</td>
<td>(optional) Specifies a suffix for the native library path.</td>
</tr>
<tr>
<td><code>envclasspathignore</code></td>
<td>true</td>
<td>(optional) If false, the CLASSPATH environment variable is read and appended to the server classpath. The CLASSPATH environment variable is added after the <code>classpath-suffix</code>, at the very end. For a development environment, this value should be set to false. For a production environment, this value should be set to true to prevent environment variable side effects. Legal values are on, off, yes, no, 1, 0, true, false.</td>
</tr>
<tr>
<td><code>bytecodepreprocessor</code></td>
<td>none</td>
<td>(optional) A comma-separated list of class names, each of which must implement the com.sun.appserv.BytecodePreprocessor interface. Each of the specified preprocessor classes is called in the order specified.</td>
</tr>
<tr>
<td><code>dynamicroloadinginterval</code></td>
<td>2</td>
<td>Specifies the interval, in seconds, after which a deployed application is reloaded.</td>
</tr>
<tr>
<td><code>loglevel</code></td>
<td></td>
<td>(optional) Controls the type of messages logged by this element to the errors log. For details, see the description of the <code>level</code> attribute of the <code>log</code> element.</td>
</tr>
</tbody>
</table>
**JVMOPTIONS**

 Defines configurable system-wide Java VM properties, for example:

```xml
<JVMOPTIONS>-Xdebug -Xmx128m</JVMOPTIONS>
```

In addition, a server in a system property,

```java
-Dcom.sun.webserv.startupclasses
```

whose value is a comma-separated list of fully qualified Java classes that server loads into the Virtual Machine upon startup. Example:

```xml
<JVMOPTIONS>
</JVMOPTIONS>
```

For information about the available options, see:


**Subelements**

none

**Attributes**

none

---

**PROFILER**

Configures a profiler for use with the server.

**Subelements**

The following table describes subelements for the `PROFILER` element.

<table>
<thead>
<tr>
<th>Table 2-31</th>
<th>PROFILER subelements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Required</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>zero or more</td>
</tr>
<tr>
<td>JVMOPTIONS</td>
<td>zero or more</td>
</tr>
</tbody>
</table>
Attributes
The following table describes attributes for the \texttt{PROFILER} element.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>classpath</td>
<td>none</td>
<td>(optional) Specifies the classpath for the profiler.</td>
</tr>
<tr>
<td>nativelibrarypath</td>
<td>none</td>
<td>(optional) Specifies the native library path for the profiler.</td>
</tr>
<tr>
<td>enabled</td>
<td>true</td>
<td>(optional) Determines whether the profiler is enabled. Legal values are on, off, yes, no, 1, 0, true, false.</td>
</tr>
</tbody>
</table>

SECURITY
Defines parameters and configuration information needed by the security service.

Subelements
The following table describes subelements for the \texttt{SECURITY} element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{PROPERTY}</td>
<td>zero or more</td>
<td>Specifies a property or a variable.</td>
</tr>
<tr>
<td>\texttt{AUTHREALM}</td>
<td>one or more</td>
<td>Defines a realm for authentication.</td>
</tr>
</tbody>
</table>

Attributes
The following table describes attributes for the \texttt{SECURITY} element.
AUTHREALM

Defines a realm for authentication.

Authentication realms require provider-specific properties, which vary depending on the needs of a particular implementation.

Here is an example of the default file realm:

```
<authrealm name="file"
    classname="com.iplanet.ias.security.auth.realm.file.FileRealm">
    <property name="file" value="instance_dir/config/keyfile"/>
    <property name="jaas-context" value="fileRealm"/>
```

Table 2-34 SECURITY attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>defaultrealm</td>
<td>file</td>
<td>(optional) Specifies the default authentication realm (an AUTHREALM name attribute) for this server instance. The default realm will be used to process authentication events for any web applications which do not otherwise specify which realm to use.</td>
</tr>
<tr>
<td>anonymousrole</td>
<td>ANYONE</td>
<td>(optional) Used as the name for default, or anonymous, role. The anonymous role is always assigned to all principals.</td>
</tr>
<tr>
<td>audit</td>
<td>false</td>
<td>(optional) If true, additional access logging is performed to provide audit information. Legal values are on, off, yes, no, 1, 0, true, false. Audit information consists of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Authentication success and failure events</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Servlet access grants and denials</td>
</tr>
<tr>
<td>loglevel</td>
<td>Value of level attribute of LOG element</td>
<td>(optional) Controls the type of messages logged by this element to the errors log. For details, see the description of the level attribute of the LOG element.</td>
</tr>
</tbody>
</table>
Which properties an AUTHREALM element uses depends on the value of the AUTHREALM element's name attribute. The file realm uses file and jaas-context properties. Other realms use different properties.

Subelements
The following table describes subelements for the AUTHREALM element.

<table>
<thead>
<tr>
<th>Table 2-35 AUTHREALM subelements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>PROPERTY</td>
</tr>
</tbody>
</table>

Attributes
The following table describes attributes for the AUTHREALM element.

<table>
<thead>
<tr>
<th>Table 2-36 AUTHREALM attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>classname</td>
</tr>
</tbody>
</table>

Properties
The standard realms provided have both required and optional properties. A custom realm may have different properties. For details about the properties and configuration characteristics of the AUTHREALM realms, refer to the chapter “Securing Web Applications” in the Sun ONE Web Server 6.1 Programmer’s Guide to Web Applications.

Resource Elements
Resource elements are as follows:

- RESOURCES
Resource Elements

- CUSTOMRESOURCE
- EXTERNALJNDIRESOURCE
- JDBCRESOURCE
- JDBCCONNECTIONPOOL
- CONNECTIONPROPERTY
- MAILRESOURCE

RESOURCES
Contains configured resources, such as database connections.

Subelements
The following table describes subelements for the RESOURCES element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOMRESOURCE</td>
<td>zero or more</td>
<td>Defines a custom resource.</td>
</tr>
<tr>
<td>EXTERNALJNDIRESOURCE</td>
<td>zero or more</td>
<td>Defines a resource that resides in an external JNDI (Java Naming and Directory Interface) repository.</td>
</tr>
<tr>
<td>JDBCRESOURCE</td>
<td>zero or more</td>
<td>Defines a JDBC (Java Database Connectivity) resource.</td>
</tr>
<tr>
<td>JDBCCONNECTIONPOOL</td>
<td>zero or more</td>
<td>Defines the properties that are required for creating a JDBC connection pool.</td>
</tr>
<tr>
<td>MAILRESOURCE</td>
<td>zero or more</td>
<td>Defines the properties that are required for creating a mail resource.</td>
</tr>
</tbody>
</table>

Attributes
none
CUSTOMRESOURCE

Defines a custom resource, which specifies a custom server-wide resource object factory. Such object factories implement the javax.naming.spi.ObjectFactory interface.

Subelements

The following table describes subelements for the CUSTOMRESOURCE element.

<table>
<thead>
<tr>
<th>Table 2-38</th>
<th>CUSTOMRESOURCE subelements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Required</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>zero or one</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>zero or more</td>
</tr>
</tbody>
</table>

Attributes

The following table describes attributes for the CUSTOMRESOURCE element.

<table>
<thead>
<tr>
<th>Table 2-39</th>
<th>CUSTOMRESOURCE attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>Default</td>
</tr>
<tr>
<td>jndiname</td>
<td>none</td>
</tr>
<tr>
<td>restype</td>
<td>none</td>
</tr>
<tr>
<td>factoryclass</td>
<td>none</td>
</tr>
<tr>
<td>enabled</td>
<td>true</td>
</tr>
</tbody>
</table>
EXTERNALJNDIRESOURCE

Defines a resource that resides in an external JNDI repository. For example, a generic Java object could be stored in an LDAP server. An external JNDI factory must implement the `javax.naming.spi.InitialContextFactory` interface.

Subelements

The following table describes subelements for the `EXTERNALJNDIRESOURCE` element.

<table>
<thead>
<tr>
<th>Table 2-40</th>
<th>EXTERNALJNDIRESOURCE subelements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Required</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>zero or one</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>zero or more</td>
</tr>
</tbody>
</table>

Attributes

The following table describes attributes for the `EXTERNALJNDIRESOURCE` element.

<table>
<thead>
<tr>
<th>Table 2-41</th>
<th>EXTERNALJNDIRESOURCE attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>Default</td>
</tr>
<tr>
<td>jndiname</td>
<td>none</td>
</tr>
<tr>
<td>jndilookupname</td>
<td>none</td>
</tr>
<tr>
<td>restype</td>
<td>none</td>
</tr>
<tr>
<td>factoryclass</td>
<td>none</td>
</tr>
<tr>
<td>enabled</td>
<td>true</td>
</tr>
</tbody>
</table>
JDBCRESOURCE

Defines a JDBC (javax.sql.DataSource) resource.

Subelements
The following table describes subelements for the JDBCRESOURCE element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIPTION</td>
<td>zero or one</td>
<td>Contains a text description of this element.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>zero or more</td>
<td>Specifies a property or a variable.</td>
</tr>
</tbody>
</table>

Attributes
The following table describes attributes for the JDBCRESOURCE element.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jndiname</td>
<td>none</td>
<td>Specifies the JNDI name for the resource.</td>
</tr>
<tr>
<td>poolname</td>
<td>none</td>
<td>Specifies the name of the associated JDBC connection pool, defined in a JDBCCONNECTIONPOOL element.</td>
</tr>
<tr>
<td>enabled</td>
<td>true</td>
<td>(optional) Determines whether this resource is enabled at runtime. Legal values are on, off, yes, no, 1, 0, true, false.</td>
</tr>
</tbody>
</table>

JDBCCONNECTIONPOOL

Defines the properties that are required for creating a JDBC connection pool.

NOTE The restype attribute of the JDBCCONNECTIONPOOL element is reserved and ignored in Sun ONE Web Server 6.1. Any value set for this attribute is ignored by the server.
Subelements
The following table describes subelements for the JDBCCONNECTIONPOOL element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIPTION</td>
<td>zero or one</td>
<td>Contains a text description of this element.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>zero or more</td>
<td>Specifies a property or a variable.</td>
</tr>
<tr>
<td>CONNECTIONPROPERTY</td>
<td>zero or more</td>
<td>Specifies the connection properties for the connection pool.</td>
</tr>
</tbody>
</table>

Attributes
The following table describes attributes for the JDBCCONNECTIONPOOL element.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>none</td>
<td>Specifies the name of the connection pool. A JDBCRESOURCE element's poolname attribute refers to this name.</td>
</tr>
<tr>
<td>datasourceclassname</td>
<td>none</td>
<td>Specifies the class name of the associated vendor-supplied data source. This class must implement java.sql.DataSource or java.sql.XADataSource or both.</td>
</tr>
<tr>
<td>steadypoolsize</td>
<td>8</td>
<td>(optional) Specifies the initial and minimum number of connections maintained in the pool.</td>
</tr>
<tr>
<td>maxpoolsize</td>
<td>32</td>
<td>(optional) Specifies the maximum number of connections that can be created to satisfy client requests.</td>
</tr>
<tr>
<td>maxwaittime</td>
<td>60000</td>
<td>(optional) Specifies the amount of time, in milliseconds, that the caller is willing to wait for a connection. If 0, the caller is blocked indefinitely until a resource is available or an error occurs.</td>
</tr>
</tbody>
</table>
## Resource Elements

Chapter 2 Server Configuration Elements in server.xml

### Table 2-45  JDBC CONNECTION POOL attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>poolresizequantity</td>
<td>2</td>
<td>(optional) Specifies the number of connections to be destroyed if the existing number of connections is above the steady-pool-size (subject to the max-pool-size limit). This is enforced periodically at the idle-time-out-in-seconds interval. An idle connection is one that has not been used for a period of idle-time-out-in-seconds.</td>
</tr>
<tr>
<td>idletimeout</td>
<td>300</td>
<td>(optional) Specifies the maximum time that a connection can remain idle in the pool. After this amount of time, the pool can close this connection.</td>
</tr>
<tr>
<td>transactionisolationlevel</td>
<td>default JDBC driver isolation level</td>
<td>(optional) Specifies the transaction isolation level on the pooled database connections. Allowed values are read-uncommitted, read-committed, repeatable-read, or serializable. Applications that change the isolation level on a pooled connection programmatically risk polluting the pool, which can lead to errors. See isolationlevelguaranteed for more details.</td>
</tr>
<tr>
<td>isolationlevelguaranteed</td>
<td>true</td>
<td>(optional) Applicable only when transactionisolationlevel is explicitly set. If true, every connection obtained from the pool is guaranteed to have the desired isolation level. This may impact performance on some JDBC drivers. You can set this attribute to false if you are certain that the hosted applications do not return connections with altered isolation levels.</td>
</tr>
<tr>
<td>connectionvalidationrequired</td>
<td>false</td>
<td>(optional) Specifies whether connections must be validated before being given to the application. If a resource’s validation fails, it is destroyed, and a new resource is created and returned. Legal values are on, off, yes, no, 1, 0, true, false.</td>
</tr>
</tbody>
</table>
Properties

Most JDBC 2.0 drivers allow use of standard property lists to specify the user, password, and other resource configuration information. Although properties are optional with respect to Sun ONE Web Server, some properties may be necessary for most databases. For details, see Section 5.3 of the JDBC 2.0 Standard Extension API.

When properties are specified, they are passed to the vendor’s data source class (specified by the datasourceclassname attribute) using setName(value) methods.

The following table describes some common properties for the JDBCConnectionPool element. The left column lists the property name, and the right column describes what the property does.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connectionvalidationmethod</td>
<td>auto-commit</td>
<td>(optional) Legal values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• auto-commit (default), which uses</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Connection.setAutoCommit(Connection.getAutoCommit())</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• meta-data, which uses</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Connection.getMetaData()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• table, which performs a query on a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>table specified in the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>validation-table-name attribute</td>
</tr>
<tr>
<td>validationtablename</td>
<td>none</td>
<td>(optional) Specifies the table name to be used to perform a query to validate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a connection. This parameter is mandatory if and only if</td>
</tr>
<tr>
<td></td>
<td></td>
<td>connectionvalidationtype is set to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>table.</td>
</tr>
<tr>
<td>failallconnections</td>
<td>false</td>
<td>(optional) If true, closes all connections in the pool if a single validation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>check fails. This parameter is mandatory if and only if</td>
</tr>
<tr>
<td></td>
<td></td>
<td>isconnectionvalidationrequired is set to true. Legal values are on, off,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>yes, no, 1, 0, true, false.</td>
</tr>
</tbody>
</table>
CONNECTIONPROPERTY

Specifies the connection properties for a JDBC connection pool.

Subelements

The following table describes subelements for the CONNECTIONPROPERTY element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIPTION</td>
<td>zero or one</td>
<td>Contains a text description of this element.</td>
</tr>
</tbody>
</table>

Attributes

The following table describes attributes for the CONNECTIONPROPERTY element.
MAILRESOURCE

Defines a JavaMail (javax.mail.Session) resource.

Subelements

The following table describes subelements for the MAILRESOURCE element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIPTION</td>
<td>zero or one</td>
<td>Contains a text description of this element.</td>
</tr>
</tbody>
</table>

Attributes

The following table describes attributes for the MAILRESOURCE element.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jndiname</td>
<td>none</td>
<td>Specifies the JNDI name for the resource.</td>
</tr>
<tr>
<td>storeprotocol</td>
<td>imap</td>
<td>(optional) Specifies the storage protocol service, which connects to a mail server, retrieves messages, and saves messages in folder(s). Example values are imap and pop3.</td>
</tr>
</tbody>
</table>
Chapter 2 Server Configuration Elements in server.xml

Configures the system logging service, which includes the following log files:

- **The errors log** file stores messages from the default virtual server. Messages from other configured virtual servers also go here, unless the logfile attribute is explicitly specified in the **VSCLASS** or **VS** element. The default name is **errors**.
- The access log file stores HTTP access messages from the default virtual server. The default name is access.log. To configure the access log, you use server application functions in the magnus.conf and obj.conf files.

- A virtual server log file stores messages from a VSCLASS or VS element that has an explicitly specified log-file attribute.

### Subelements
The following table describes subelements for the LOG element.

**Table 2-51 LOG subelements**

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROPERTY</td>
<td>zero or more</td>
<td>Specifies a property or a variable.</td>
</tr>
</tbody>
</table>

### Attributes
The following table describes attributes for the LOG element.

**Table 2-52 LOG attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>errors</td>
<td>Specifies the file that stores messages from the default virtual server. Messages from other configured virtual servers also go here, unless the errorlog attribute is explicitly specified in the VS element.</td>
</tr>
<tr>
<td>loglevel</td>
<td>info</td>
<td>Controls the default type of messages logged by other elements to the error log. Allowed values are as follows, from highest to lowest: finest, finer, fine, info, warning, failure, config, security, and catastrophe.</td>
</tr>
<tr>
<td>logvsid</td>
<td>false</td>
<td>(optional) If true, virtual server IDs are displayed in the virtual server logs. This is useful if multiple VS elements share the same log file. Legal values are on, off, yes, no, 1, 0, true, false.</td>
</tr>
</tbody>
</table>
User Database Selection

A USERDB object selects a user database for the parent virtual server. This selection occurs in the following manner:

- The USERDB element’s id attribute maps to an ACL file’s database attribute.
- The USERDB element’s database attribute maps to a dbswitch.conf entry.

This layer between the ACL file and the dbswitch.conf file gives the server administrator full control over which databases virtual server administrators and users have access to.

The dbswitch.conf file establishes the root of the search tree for LDAP databases as follows:

- The base DN in the LDAP URL in dbswitch.conf defines a root object for all further DN specifications. So, for most new installations, it can be empty, because the final base DN is determined in other ways -- either through a dc tree lookup or an explicit basedn value in the USERDB tag.

### Table 2-52 LOG attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>logstdout</td>
<td>true</td>
<td>(optional) If true, redirects stdout output to the errors log. Legal values are on, off, yes, no, 1, 0, true, false.</td>
</tr>
<tr>
<td>logstderr</td>
<td>true</td>
<td>(optional) If true, redirects stderr output to the errors log. Legal values are on, off, yes, no, 1, 0, true, false.</td>
</tr>
<tr>
<td>logtoconsole</td>
<td>true</td>
<td>(optional, UNIX only) If true, redirects log messages to the console.</td>
</tr>
<tr>
<td>createconsole</td>
<td>false</td>
<td>(optional, Windows only) If true, creates a Windows console. Legal values are on, off, yes, no, 1, 0, true, false.</td>
</tr>
<tr>
<td>usesyslog</td>
<td>false</td>
<td>(optional) If true, uses the UNIX syslog service or Windows Event Logging to produce and manage logs. Legal values are on, off, yes, no, 1, 0, true, false.</td>
</tr>
</tbody>
</table>
A new `dbswitch.conf` attribute for LDAP databases, `dcsuffix`, defines the root of the dc tree. This root is relative to the base DN in the LDAP URL. You can use `dcsuffix` if the database is schema compliant. Requirements for schema compliance are listed in “The Sun ONE LDAP Schema” on page 68.

A user database is selected for a virtual server as follows:

- If a VS has no `USERDB` subelement, user- or group-based ACLs fail.
- When no database attribute is present in a virtual server’s ACL definition, the VS must have a `USERDB` subelement with an id attribute of default. The database attribute of the `USERDB` then points to a database in `dbswitch.conf`. If no database attribute is present, default is used.
- If an LDAP database is schema compliant, the base DN of the access is computed using a dc tree lookup of the VS element’s hosts attribute that matches the client-supplied Host header. If no hosts attribute matches, the `servername` attribute of the parent `SERVER` is used. The dc tree lookup is based at the `dcsuffix` DN. The result must contain an `inetDomainBaseDN` attribute that contains the base DN. This base DN is taken as is and is not relative to any of the base DN values.
- If the `basename` attribute of the `USERDB` element is not present and the database is not schema compliant, the access requests are relative to the base DN in the `dbswitch.conf` entry, as in previous Sun ONE Web Server versions.

The Sun ONE LDAP Schema

This section describes the Sun ONE LDAP Schema that defines a set of rules for directory data.

You can use the `dcsuffix` attribute in the `dbswitch.conf` file if your LDAP database meets the requirements outlined in this section. For more information about the `dbswitch.conf` file, see “`dbswitch.conf`” on page 229.

The subtree rooted at an ISP entry (for example, `o=isp`) is called the convergence tree. It contains all directory data related to organizations (customers) served by an ISP.

The subtree rooted at `o=internet` is called the domain component tree, or dc tree. It contains a sparse DNS tree with entries for the customer domains served. These entries are links to the appropriate location in the convergence tree where the data for that domain is located.
The directory tree may be single rooted, which is recommended (for example, \texttt{o=root} may have \texttt{o=isp} and \texttt{o=internet} under it), or have two separate roots, one for the convergence tree and one for the \texttt{dc} tree.

### The Convergence Tree

The top level of the convergence tree must have one organization entry for each customer (or organization), and one for the ISP itself.

Underneath each organization, there must be two \texttt{organizationalUnit} entries: \texttt{ou=People} and \texttt{ou=Groups}. A third, \texttt{ou=Devices}, can be present if device data is to be stored for the organization.

Each user entry must have a unique \texttt{uid} value within a given organization. The namespace under this subtree can be partitioned into various \texttt{ou} entries that aggregate user entries in convenient groups (for example, \texttt{ou=eng}, \texttt{ou=corp}). User \texttt{uid} values must still be unique within the entire \texttt{People} subtree.

User entries in the convergence tree are of type \texttt{inetOrgPerson}. The \texttt{cn}, \texttt{sn}, and \texttt{uid} attributes must be present. The \texttt{uid} attribute must be a valid e-mail name (specifically, it must be a valid local-part as defined in RFC822). It is recommended that the \texttt{cn} contain name initial \texttt{sn}. It is recommended that the RDN of the user entry be the \texttt{uid} value. User entries must contain the auxiliary class \texttt{inetUser} if they are to be considered enabled for service or valid.

User entries can also contain the auxiliary class \texttt{inetSubscriber}, which is used for account management purposes. If an \texttt{inetUserStatus} attribute is present in an entry and has a value of \texttt{inactive} or \texttt{deleted}, the entry is ignored.

Groups are located under the \texttt{Groups} subtree and consist of LDAP entries of type \texttt{groupOfUniqueNames}.

### The Domain Component (dc) Tree

The \texttt{dc} tree contains hierarchical \texttt{domain} entries, each of which is a DNS name component.

Entries that represent the domain name of a customer are overlaid with the LDAP auxiliary class \texttt{inetDomain}. For example, the two LDAP entries
\texttt{dc=customer1,dc=com,o=Internet,o=root} and
\texttt{dc=customer2,dc=com,o=Internet,o=root} contain the \texttt{inetDomain} class, but \texttt{dc=com,o=Internet,o=root} does not. The latter is present only to provide structure to the tree.
Entries with an `inetDomain` attribute are called virtual domains. These must have the attribute `inetDomainBaseDN` filled with the DN of the top level organization entry where the data of this domain is stored in the convergence tree. For example, the virtual domain entry in `dc=cust2,dc=com,o=Internet,o=root` would contain the attribute `inetDomainBaseDN` with value `o=Cust2,o=isp,o=root`.

If an `inetDomainStatus` attribute is present in an entry and has a value of `inactive` or `deleted`, the entry is ignored.

Some variables are defined in `server.xml` for use in the `obj.conf` file. The following file fragment defines a `docroot` variable:

```xml
<PROPERTY name="docroot" value="/server/docs/class2/acme" />
```

A `docroot` variable allows different document root directories to be assigned for different virtual servers. The variable is then used in the `obj.conf` file. For example:

```xml
NameTrans fn=document-root root="$docroot"
```

Using this `docroot` variable allows you to define different document roots for different virtual servers within the same virtual server class.

A variable is found in `obj.conf` when the following regular expression matches:

```
\$[A-Za-z][A-Za-z0-9_]*
```

This expression represents a `$` followed by one or more alphanumeric characters. A delimited version (`"${property}"`) is not supported. To get a regular `$` character, use `$$` to have variable substitution.

A special variable, `id`, is always available within a `VS` element and refers to the value of the `id` attribute. It is predefined and cannot be overridden. The `id` attribute uniquely identifies a virtual server. For example:

```xml
<PROPERTY name=docroot value="/export/$id" />
```
If the `id` attribute of the parent `VS` element is `myserver`, the `docroot` variable is set to the value `/export/myserver`.

**Other Important Variables**

In a default installation, the following variables are used to configure various aspects of the server’s operation. Unlike the `$id` variable, they are not predefined in the server, and they can be overridden.

**General Variables**

The following table lists general `server.xml` variables. The left column lists variables, and the right column lists descriptions of those variables.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>docroot</code></td>
<td>The document root of the virtual server. Typically evaluated as the parameter to the <code>document-root</code> function in the <code>obj.conf</code> file.</td>
</tr>
<tr>
<td><code>accesslog</code></td>
<td>The access log file for a virtual server.</td>
</tr>
</tbody>
</table>

**send-cgi Variables**

The following table lists `server.xml` variables used by the `send-cgi` function in the `obj.conf` file. The left column lists variables, and the right column lists descriptions of those variables.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>user</code></td>
<td>The value of the <code>user</code> CGI parameter.</td>
</tr>
<tr>
<td><code>group</code></td>
<td>The value of the <code>group</code> CGI parameter.</td>
</tr>
<tr>
<td><code>chroot</code></td>
<td>The value of the <code>chroot</code> CGI parameter.</td>
</tr>
<tr>
<td><code>dir</code></td>
<td>The value of the <code>dir</code> CGI parameter.</td>
</tr>
<tr>
<td><code>nice</code></td>
<td>The value of the <code>nice</code> CGI parameter.</td>
</tr>
</tbody>
</table>
For more information about the `send-cgi` function, see the Sun ONE Web Server 6.1 NSAPI Programmer’s Guide.

Variable Evaluation

Variables are evaluated when generating specific objectsets for individual virtual servers. Evaluation is recursive: variable values can contain other variables. For example:

```xml
...<VSCLASS>
    ...
    <VS ...
        ...
          <PROPERTY name=docroot value="$docrootbase/nonjava/$id" />
    </VS>
    <VS ...>
        ...
          <PROPERTY name=docroot value="$docrootbase/java/$id" />
    </VS>
    ...
    <PROPERTY name=docrootbase value="/export" />
</VSCLASS>
...```

Variables in subelements override variables in the parent elements. For example, it is possible to set a variable for a class of virtual servers and override it with a definition of the same variable in an individual virtual server.

Sample server.xml File

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!--
 Copyright (c) 2003 Sun Microsystems, Inc. All rights reserved.
-->
```
Use is subject to license terms.

```xml
<!DOCTYPE SERVER PUBLIC "-//Sun Microsystems Inc./DTD Sun ONE Web Server 6.1//EN" "file://home/nb136819/space/servers/s1ws61/bin/https/dtds/sun-web-server_6_1.dtd">
<SERVER>
  <PROPERTY name="docroot" value="/home/nb136819/space/servers/s1ws61/docs"/>
  <PROPERTY name="accesslog" value="/home/nb136819/space/servers/s1ws61/https-admserv/logs/access"/>
  <PROPERTY name="user" value=""/>
  <PROPERTY name="group" value=""/>
  <PROPERTY name="chroot" value=""/>
  <PROPERTY name="dir" value=""/>
  <PROPERTY name="nice" value=""/>
  <LS id="ls1" port="5555" servername="plaza.india.sun.com" defaultvs="vs-admin"/>
  <LS id="ls2" port="9999" servername="plaza.india.sun.com" defaultvs="useradmin"/>
  <MIME id="mime1" file="mime.types"/>
  <ACLFILE id="acl1" file="/home/nb136819/space/servers/s1ws61/httpacl/generated.https-admserv.acl"/>
  <VSCLASS id="vsclass-admin" objectfile="obj.conf">
    <VS id="vs-admin" connections="ls1" mime="mime1" aclids="acl1" urlhosts="plaza.india.sun.com">
      <PROPERTY name="docroot" value="/home/nb136819/space/servers/s1ws61/docs"/>
      <USERDB id="default"/>
      <WEBAPP uri="/admin-app" path="/home/nb136819/space/servers/s1ws61/bin/https/webapps/admin-app"/>
    </VS>
  </VSCLASS>
</SERVER>
```
<VSCLASS id="userclass" objectfile="userclass.obj.conf">
    <VS id="useradmin" connections="ls2" mime="mime1" aclids="acl1" urlhosts="plaza.india.sun.com">
        <PROPERTY name="docroot" value="/home/nb136819/space/servers/slws61/docs"/>
        <USERDB id="default"/>
        <WEBAPP uri="/user-app" path="/home/nb136819/space/servers/slws61/bin/https/webapps/user-app"/>
    </VS>
</VSCLASS>

    <JVMOPTIONS>-Dorg.xml.sax.parser=org.xml.sax.helpers.XMLReaderAdapter</JVMOPTIONS>
    <JVMOPTIONS>-Dorg.xml.sax.driver=org.apache.crimson.parser.XMLReaderImpl</JVMOPTIONS>
    <JVMOPTIONS>-Djava.security.manager</JVMOPTIONS>
    <JVMOPTIONS>-Djava.security.policy=/home/nb136819/space/servers/slws61/https-admserv/config/server.policy</JVMOPTIONS>
    <JVMOPTIONS>-Djava.security.auth.login.config=/home/nb136819/space/servers/slws61/https-admserv/config/login.conf</JVMOPTIONS>
    <JVMOPTIONS>-Djava.util.logging.manager=com.iplanet.ias.server.logging.ServerLogManager</JVMOPTIONS>
    <JVMOPTIONS>-Xms128m -Xmx256m</JVMOPTIONS>
</JAVA>
<SECURITY defaultrealm="file" anonymousrole="ANYONE" audit="false">
    <AUTHREALM name="file" classname="com.iplanet.ias.security.auth.realm.file.FileRealm">
        <PROPERTY name="file" value="/home/nb136819/space/servers/s1ws61/https-admserv/config/key file"/>
        <PROPERTY name="jaas-context" value="fileRealm"/>
    </AUTHREALM>
</SECURITY>

<RESOURCES>
    </RESOURCES>
</JAVA>

<LOG file="/home/nb136819/space/servers/s1ws61/https-admserv/logs/errors" loglevel="info"/>

</SERVER>
When the Sun ONE Web Server starts up, it looks in a file called `magnus.conf` in the `server-id/config` directory to establish a set of global variable settings that affect the server’s behavior and configuration. Sun ONE Web Server executes all the directives defined in `magnus.conf`. The order of the directives is not important.

**NOTE**  
When you edit the `magnus.conf` file, you must restart the server for the changes to take effect.

This chapter lists the global settings that can be specified in `magnus.conf` in Sun ONE Web Server 6.1.

The categories are:

- **Init Functions**
- **Server Information**
- **Language Issues**
- **DNS Lookup**
- **Threads, Processes, and Connections**
- **Native Thread Pools**
- **CGI**
- **Error Logging and Statistic Collection**
- **ACL**
Init Functions

- Security
- Chunked Encoding
- Miscellaneous Directives

For an alphabetical list of directives, see Appendix D, “Alphabetical List of Server Configuration Elements”.

For a list of magnus.conf directives deprecated in Sun ONE Web Server 6.1, see Deprecated Directives.

NOTE Much of the functionality of the file cache is controlled by a configuration file called nsfc.conf. For information about nsfc.conf, see “nsfc.conf” on page 233.

Init Functions

The Init functions load and initialize server modules and plugins, and initialize log files. For more information about these functions, see the Sun ONE Web Server NSAPI Programmer’s Guide.

Server Information

This sub-section lists the directives in magnus.conf that specify information about the server. They are:

- ExtraPath
- TempDir
- TempDirSecurity
- User

ExtraPath

Appends the specified directory name to the PATH environment variable. This is used for configuring Java on Windows. There is no default value; you must specify a value.
Syntax

```
ExtraPath  path
```

**TempDir**

Specifies the directory on the local volume that the server uses for its temporary files. On UNIX, this directory must be owned by, and writable by, the user the server runs as. See also the directives `User` and `TempDirSecurity`.

Syntax

```
TempDir  path
```

**Default**

```
/tmp (UNIX)
TEMP (environment variable for Windows)
```

**TempDirSecurity**

Determines whether the server checks if the TempDir directory is secure. On UNIX, specifying `TempDirSecurity off` allows the server to use `/tmp` as a temporary directory.

**CAUTION** Specifying `TempDirSecurity off` or using `/tmp` as a temporary directory on UNIX is highly discouraged. Using `/tmp` as a temporary directory opens a number of potential security risks.

Syntax

```
TempDirSecurity  [on|off]
```

**Default**

```
on
```

**User**

**Windows:** The `User` directive specifies the user account the server runs with. By using a specific user account (other than LocalSystem), you can restrict or enable system features for the server. For example, you can use a user account that can mount files from another machine.
UNIX: The `User` directive specifies the UNIX 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 that someone compromises the server, they won’t be able to change configuration files and gain broader access to your machine. 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 user account.

**Default**

If there is no `User` directive, the server runs with the user account it was started with.

**Examples**

```
User http
User server
User nobody
```

### Language Issues

This section lists the directives in `magnus.conf` related to language issues. The following directive is supported:

- `DefaultLanguage`

#### DefaultLanguage

For an international version of the server, this directive specifies the default language for the server. The default language is used for both the client responses and administration. Values are `en` (English), `fr` (French), `de` (German) or `ja` (Japanese).

**Default**

The default is `en`. 
DNS Lookup

This section lists the directives in magnus.conf that affect DNS (Domain Name System) lookup. The directives are:

- **AsyncDNS**
- **DNS**

**AsyncDNS**
Specifies whether asynchronous DNS is allowed. This directive is ignored. Even if the value is set to on, the server does not perform asynchronous DNS lookups.

**DNS**
The **DNS** directive specifies whether the server performs DNS lookups on clients that access the server. When a client connects to your server, the server knows the client’s IP address but not its host name (for example, it knows the client as 198.95.251.30, rather than its host name *www.a.com*). The server will resolve the client’s IP address into a host name for operations like access control, CGI, error reporting, and access logging.

If your server responds to many requests per day, you might want (or need) to stop host name resolution; doing so can reduce the load on the DNS or NIS (Network Information System) server.

**Syntax**
```
DNS [on | off]
```

**Default**
DNS host name resolution is on as a default.

**Example**
```
DNS on
```

Threads, Processes, and Connections

In Sun ONE Web Server 6.1, acceptor threads on a listen socket accept connections and put them onto a connection queue. Session threads then pick up connections from the queue and service the requests. The session threads post more session threads if required at the end of the request. The policy for adding new threads is based on the connection queue state:
Each time a new connection is returned, the number of connections waiting in the queue (the backlog of connections) is compared to the number of session threads already created. If it is greater than the number of threads, more threads are scheduled to be added the next time a request completes.

The previous backlog is tracked, so that if it is seen to be increasing over time, and if the increase is greater than the ThreadIncrement value, and the number of session threads minus the backlog is less than the ThreadIncrement value, then another ThreadIncrement number of threads are scheduled to be added.

The process of adding new session threads is strictly limited by the RqThrottle value.

To avoid creating too many threads when the backlog increases suddenly (such as the startup of benchmark loads), the decision whether more threads are needed is made only once every 16 or 32 times a connection is made based on how many session threads already exist.

This subsection lists the directives in magnus.conf that affect the number and timeout of threads, processes, and connections. They are:

- AcceptTimeout
- ConnQueueSize
- HeaderBufferSize
- KeepAliveQueryMaxSleepTime
- KeepAliveQueryMeanTime
- KeepAliveThreads
- KeepAliveTimeout
- KernelThreads
- ListenQ
- MaxKeepAliveConnections
- MaxProcs (UNIX Only)
- PostThreadsEarly
- RcvBufSize
- RqThrottle
- RqThrottleMin
- SndBufSize
Threads, Processes, and Connections

- **StackSize**
- **StrictHttpHeaders**
- **TerminateTimeout**
- **ThreadIncrement**
- **UseNativePoll (UNIX only)**

Also see the section “Native Thread Pools” on page 88 for directives for controlling the pool of native kernel threads.

For more information about performance tuning, see the Sun ONE Web Server 6.1 Performance Tuning, Sizing, and Scaling Guide.

**AcceptTimeout**

Specifies the number of seconds the server waits for data to arrive from the client. If data does not arrive before the timeout expires then the connection is closed. By setting it to less than the default 30 seconds, you can free up threads sooner. However, you may also disconnect users with slower connections.

**Syntax**

`AcceptTimeout seconds`

**Default**

30 seconds for servers that don’t use hardware encryption devices and 300 seconds for those that do.

**ConnQueueSize**

Specifies the number of outstanding (yet to be serviced) connections that the web server can have. It is recommended that this value always be greater than the operating system limit for the maximum number of open file descriptors per process.

This setting can have performance implications. For more information, see the Sun ONE Web Server 6.1 Performance Tuning, Sizing, and Scaling Guide.

**Default**

The default value is 4096.
Threads, Processes, and Connections

**HeaderBufferSize**
The size (in bytes) of the buffer used by each of the request processing threads for reading the request data from the client. The maximum number of request processing threads is controlled by the `RqThrottle` setting.

**Default**
The default value is 8192 (8 KB).

**KeepAliveQueryMaxSleepTime**
This directive specifies an upper limit to the time slept (in milliseconds) after polling keep-alive connections for further requests.

**Default**
The default is 100.

On lightly loaded systems that primarily service keep-alive connections, you can lower this number to enhance performance. However doing so can increase CPU usage.

**KeepAliveQueryMeanTime**
This directive specifies the desired keep-alive latency in milliseconds.

**Default**
The default value of 100 is appropriate for almost all installations.

Note that CPU usage will increase with lower `KeepAliveQueryMeanTime` values.

**KeepAliveThreads**
This directive determines the number of threads in the keep-alive subsystem. It is recommended that this number be a small multiple of the number of processors on the system (for example, a 2 CPU system should have 2 or 4 keep alive threads). The maximum number of keep-alive connections allowed (`MaxKeepAliveConnections`) should also be taken into consideration when choosing a value for this setting.

**Default**
1
KeepAliveTimeout
This directive determines the maximum time that the server holds open an HTTP Keep-Alive connection or a persistent connection between the client and the server. The Keep-Alive feature for earlier versions of the server allows the client/server connection to stay open while the server processes the client request. The default connection is a persistent connection that remains open until the server closes it or the connection has been open for longer than the time allowed by KeepAliveTimeout.

The timeout countdown starts when the connection is handed over to the keep-alive subsystem. If there is no activity on the connection when the timeout expires, the connection is closed.

Default
The default value is 30 seconds. The maximum value is 300 seconds (5 minutes).

KernelThreads
Sun ONE Web Server can support both kernel-level and user-level threads whenever the operating system supports kernel-level threads. Local threads are scheduled by NSPR (Netscape Portable Runtime) within the process, whereas kernel threads are scheduled by the host operating system. Usually, the standard debugger and compiler are intended for use with kernel-level threads. By setting KernelThreads to 1 (on), you ensure that the server uses only kernel-level threads, not user-level threads. By setting KernelThreads to 0 (off), you ensure that the server uses only user-level threads, which may improve performance.

Default
The default is 0 (off).

ListenQ
Specifies the maximum number of pending connections on a listen socket. Connections that time out on a listen socket whose backlog queue is full will fail.

Default
The default value is platform-specific: 4096 (AIX), 200 (), 128 (all others).

MaxKeepAliveConnections
Specifies the maximum number of Keep-Alive and persistent connections that the server can have open simultaneously. Values range from 0 to 32768.
Threads, Processes, and Connections

Default

MaxProcs (UNIX Only)
Specifies the maximum number of processes that the server can have running simultaneously. If you don’t include `MaxProcs` in your `magnus.conf` file, the server defaults to running a single process.

One process per processor is recommended if you are running in multi-process mode. In Sun ONE Web Server 6.1, there is always a primordial process in addition to the number of active processes specified by this setting.

Additional discussion of this and other server configuration and performance tuning issues can be found in the Sun ONE Web Server 6.1 Performance Tuning, Sizing, and Scaling Guide.

Default
1

PostThreadsEarly
If this directive is set to 1 (on), the server checks the whether the minimum number of threads are available at a listen socket after accepting a connection but before sending the response to the request. Use this directive when the server will be handling requests that take a long time to handle, such as those that do long database connections.

Default
0 (off)

RcvBufSize
Specifies the size (in bytes) of the receive buffer used by sockets. Allowed values are determined by the operating system.

Default
The default value is determined by the operating system. Typical defaults are 4096 (4K), 8192 (8K).

RqThrottle
Specifies the maximum number of request processing threads that the server can handle simultaneously. Each request runs in its own thread.
This setting can have performance implications. For more information, see the Sun ONE Web Server 6.1 *Performance Tuning, Sizing, and Scaling Guide*.

**RqThrottleMin**  
Specifies the number of request processing threads that are created when the server is started. As the load on the server increases, more request processing threads are created (up to a maximum of $RqThrottle$ threads).

**SndBufSize**  
Specifies the size (in bytes) of the send buffer used by sockets.

**Default**  
The default value is determined by the operating system. Typical defaults are 4096 (4K), 8192 (8K).

**StackTrace**  
Determines the maximum stack size for each request handling thread.

**Default**  
The most favorable machine-specific stack size.

**StrictHttpHeaders**  
Controls strict HTTP header checking. If strict HTTP header checking is on, the server rejects connections that include inappropriately duplicated headers.

**Syntax**  
`StrictHttpHeaders [on|off]`

**Default**  
on

**TerminateTimeout**  
Specifies the time that the server waits for all existing connections to terminate before it shuts down.

**Default**  
30 seconds
Native Thread Pools

**ThreadIncrement**
The number of additional or new request processing threads created to handle an increase in the load on the server, for example when the number of pending connections (in the request processing queue) exceeds the number of idle request processing threads.

When a server starts up, it creates $RqThrottleMin$ number of request processing threads. As the load increases, it creates $ThreadIncrement$ additional request processing threads until $RqThrottle$ request processing threads have been created.

**Default**
The default value is 10.

**UseNativePoll (UNIX only)**
Uses a platform-specific poll interface when set to 1(on). Uses the NSPR poll interface in the KeepAlive subsystem when set to 0 (off).

**Default**
1 (on)

Native Thread Pools

This section lists the directives for controlling the size of the native kernel thread pool. You can also control the native thread pool by setting the system variables $NSCP\_POOL\_STACKSIZE$, $NSCP\_POOL\_THREADMAX$, and $NSCP\_POOL\_WORKQUEUEMAX$. If you have set these values as environment variables and also in $magnus\.conf$, the environment variable values will take precedence.

The native pool on UNIX is normally not engaged, as all threads are OS-level threads. Using native pools on UNIX may introduce a small performance overhead as they’ll require an additional context switch; however, they can be used to localize the jvm.stickyAttach effect or for other purposes, such as resource control and management or to emulate single-threaded behavior for plug-ins.

On Windows, the default native pool is always being used and Sun ONE Web Server uses fibers (user-scheduled threads) for initial request processing. Using custom additional pools on Windows introduces no additional overhead.

The directives are:

- **NativePoolStackSize**
NativePoolStackSize
Determines the stack size of each thread in the native (kernel) thread pool.

Default
0

NativePoolMaxThreads
Determines the maximum number of threads in the native (kernel) thread pool.

Default

NativePoolMinThreads
Determines the minimum number of threads in the native (kernel) thread pool.

Default
1

NativePoolQueueSize
Determines the number of threads that can wait in the queue for the thread pool. If all threads in the pool are busy, then the next request-handling thread that needs to use a thread in the native pool must wait in the queue. If the queue is full, the next request-handling thread that tries to get in the queue is rejected, with the result that it returns a busy response to the client. It is then free to handle another incoming request instead of being tied up waiting in the queue.

Default
0

This section lists the directives in magnus.conf that affect requests for CGI programs. The directives are:
• CGIExpirationTimeout
• CGIStubIdleTimeout
• CGIWaitPid (UNIX Only)
• MaxCGIStubs
• MinCGIStubs

**CGIExpirationTimeout**
This directive specifies the maximum time in seconds that CGI processes are allowed to run before being killed.

The value of `CGIExpirationTimeout` should not be set too low — 300 seconds (5 minutes) would be a good value for most interactive CGIs; but if you have CGIs that are expected to take longer without misbehaving, then you should set it to the maximum duration you expect a CGI program to run normally. A value of 0 disables CGI expiration, which means that there is no time limit for CGI processes.

Note that on Windows platforms `init-cgi` time-out does not work, so you must use `CGIExpirationTimeout`.

**Default**
0

**CGIStubIdleTimeout**
This directive causes the server to kill any CGIStub processes that have been idle for the number of seconds set by this directive. Once the number of processes is at `MinCGIStubs`, the server does not kill any more processes.

**Default**
30

**CGIWaitPid (UNIX Only)**
For UNIX platforms, when `CGIWaitPid` is set to on, the action for the SIGCHLD signal is the system default action for the signal. If a NSAPI plugin fork/execs a child process, it should call `waitpid` with its child process `pid` when `CGIWaitPid` is enabled to avoid leaving “defunct” processes when its child process terminates. When `CGIWaitPid` is enabled, the SHTML engine waits explicitly on its exec cmd child processes. Note that this directive has no effect on CGI.
Default

on

MaxCGIStubs
Controls the maximum number of CGIStub processes the server can spawn. This is the maximum concurrent CGIStub processes in execution, not the maximum number of pending requests. The default value should be adequate for most systems. Setting this too high may actually reduce throughput.

Default

10

MinCGIStubs
Controls the number of processes that are started by default. The first CGIStub process is not started until a CGI program has been accessed. Note that if you have an init-cgi directive in the magnus.conf file, the minimum number of CGIStub processes are spawned at startup. The value must be less than the MaxCGIStubs value.

Default

2

WincgiTimeout
WinCGI processes that take longer than this value are terminated when this timeout (in seconds) expires.

Default

60

Error Logging and Statistic Collection
This section lists the directives in magnus.conf that affect error logging and the collection of server statistics. They are:

- ErrorLogDateFormat
- LogFlushInterval
- PidLog
**ErrorLogDateFormat**
The `ErrorLogDateFormat` directive specifies the date format that the server logs use.

**Syntax**

```
ErrorLogDateFormat format
```

The `format` can be any format valid for the C library function `strftime`. See Appendix C, “Time Formats”.

**Default**

```
%d/%b/%Y:%H:%M:%S
```

**LogFlushInterval**

This directive determines the log flush interval, in seconds, of the log flush thread for the access log.

**Default**

```
30
```

**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 this 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. The server won’t start if it can’t log the process ID.

**Syntax**

```
PidLog file
```

The `file` is the full path name and file name where the process ID is stored.

**Default**

There is no default.

**Examples**

```
PidLog /var/ns-server/logs/pid
```
This section lists the directives in magnus.conf relevant to access control lists (ACLs). They are:

- ACLCacheLifetime
- ACLUserCacheSize
- ACLGroupCacheSize

**ACLCacheLifetime**

ACLCacheLifetime determines the number of seconds before cache entries expire. Each time an entry in the cache is referenced, its age is calculated and checked against ACLCacheLifetime. The entry is not used if its age is greater than or equal to the ACLCacheLifetime. If this value is set to 0, the cache is turned off.

If you use a large number for this value, you may need to restart the Sun ONE Web Server when you make changes to the LDAP entries. For example, if this value is set to 120 seconds, the Sun ONE Web Server might be out of sync with the LDAP server for as long as two minutes. If your LDAP entries are not likely to change often, use a large number.

**Default**

120

**ACLUserCacheSize**

ACLUserCacheSize determines the number of users in the User Cache.

**Default**

200

**ACLGroupCacheSize**

ACLGroupCacheSize determines how many group IDs can be cached for a single UID/cache entry.

**Default**

4
Security

This section lists the directives in magnus.conf that affect server access and security issues for Sun ONE Web Server. They are:

- **Security**
- **ServerString**
- **SSLCacheEntries**
- **SSLClientAuthDataLimit**
- **SSLClientAuthTimeout**
- **SSLSessionTimeout**
- **SSL3SessionTimeout**

**Security**
The Security directive globally enables or disables SSL by making certificates available to the server instance. It must be on for virtual servers to use SSL. If enabled, the user is prompted for the administrator password (in order to access certificates, and so on).

**NOTE**
When you create a secure listen socket through the Server Manager, security is automatically turned on globally in magnus.conf.
When you create a secure listen socket manually in server.xml, security must be turned on by editing magnus.conf.

**Syntax**
Security [on|off]

**Default**
off

**Example**
Security off

**ServerString**
Allows the administrator to change the string sent with the Server HTTP header.
Syntax

ServerString string

string is the new string to send as the header. All characters, including quotes, will be sent. The string none, will cause the header to not be sent at all.

Example
ServerString My Own Server/1.0
ServerString none

SSLCacheEntries
Specifies the number of SSL sessions that can be cached. There is no upper limit.

Syntax
SSLCacheEntries number

If the number is 0, the default value, which is 10000, is used.

SSLClientAuthDataLimit
Specifies the maximum amount of application data, in bytes, that is buffered during the client certificate handshake phase.

Default
The default value is 1048576 (1 MB).

SSLClientAuthTimeout
Specifies the number of seconds after which the client certificate handshake phase times out.

Default
60

SSLSessionTimeout
The SSLSessionTimeout directive controls SSL2 session caching.

Syntax
SSLSessionTimeout seconds

The seconds value is the number of seconds until a cached SSL2 session becomes invalid. If the SSLSessionTimeout directive is specified, the value of seconds is silently constrained to be between 5 and 100 seconds.
Default
The default value is 100.

SSL3SessionTimeout
The SSL3SessionTimeout directive controls SSL3 session caching.

Syntax
SSL3SessionTimeout seconds
The seconds value is the number of seconds until a cached SSL3 session becomes invalid. The default value is 86400 (24 hours). If the SSL3SessionTimeout directive is specified, the value of seconds is silently constrained to be between 5 and 86400 seconds.

Chunked Encoding
This section lists directives that control chunked encoding. For more information, see the Sun ONE Web Server 6.1 NSAPI Programmer’s Guide.

- UseOutputStreamSize
- ChunkedRequestBufferSize
- ChunkedRequestTimeout
These directives have equivalent Service SAF parameters in obj.conf. The obj.conf parameters override these directives. For more information, see the Sun ONE Web Server 6.1 NSAPI Programmer’s Guide.

UseOutputStreamSize
The UseOutputStreamSize directive determines the default output stream buffer size for the net_read and netbuf_grab NSAPI functions.

NOTE
The UseOutputStreamSize parameter can be set to 0 in the obj.conf file to disable output stream buffering. For the magnus.conf file, setting UseOutputStreamSize to 0 has no effect.

Syntax
UseOutputStreamSize size
The size value is the number of bytes.

**Default**
The default value is 8192 (8 KB).

**ChunkedRequestBufferSize**
The ChunkedRequestBufferSize directive determines the default buffer size for “un-chunking” request data.

**Syntax**
ChunkedRequestBufferSize size

The size value is the number of bytes.

**Default**
The default value is 8192.

**ChunkedRequestTimeout**
The ChunkedRequestTimeout directive determines the default timeout for “un-chunking” request data.

**Syntax**
ChunkedRequestTimeout seconds

The seconds value is the number of seconds.

**Default**
The default value is 60 (1 minute).

---

**Miscellaneous Directives**

This section lists the following miscellaneous directives in magnus.conf:

- ChildRestartCallback
- Favicon
- HTTPVersion
- MaxRqHeaders
- Umask (UNIX only)
ChildRestartCallback
This directive forces the callback of NSAPI functions that were registered using the `daemon_atrestart` function when the server is restarting or shutting down. Values are `on`, `off`, `yes`, `no`, `true`, or `false`.

Default
no

Favicon
To turn off the internal `favicon.ico` support, add the following line to `magnus.conf`:
Favicon off

HTTPVersion
The current HTTP version used by the server in the form `m.n`, where `m` is the major version number and `n` the minor version number.

Default
The default value is `1.1`.

MaxRqHeaders
Specifies the maximum number of header lines in a request. Values range from 1 to 512.

Default
64

Umask (UNIX only)
This directive specifies the umask value used by the NSAPI functions `System_fopenWA()` and `System_fopenRW()` to open files in different modes. Valid values for this directive are standard UNIX umask values.
Deprecated Directives

The following directives have been deprecated in Sun ONE Web Server 6.1:

- AdminLanguage
- ClientLanguage
- NetsiteRoot
- ServerID
- ServerName
- ServerRoot

AdminLanguage
For an international version of the server, this directive specifies the language for the Server Manager. Values are en (English), fr (French), de (German) or ja (Japanese).

Default
The default is en.

ClientLanguage
For an international version of the server, this directive specifies the language for client messages (such as File Not Found). Values are en (English), fr (French), de (German) or ja (Japanese).

Default
The default is en.

NetsiteRoot
Specifies the absolute pathname to the top-level directory under which server instances can be found. This directive is used by the Administration Server. There is no default value; you must specify a value.
Syntax

NetsiteRoot  path

ServerID
Specifies the server ID, such as https-boots.mcom.com.

ServerName
Specifies the server name.

ServerRoot
Specifies the server root. This directive is set during installation and is commented out. Unlike other directives, the server expects this directive to start with #. Do not change this directive. If you do, the Server Manager may not function properly.

Syntax

#ServerRoot  path

Example

#ServerRoot  d:/netscape/server4/https-boots.mcom.com

Summary of Init Functions and Directives in magnus.conf

Purpose
Contains global variable settings that affect server functioning. This file is read only at server start-up.

Location
server_root/https-admserv/config
server_root/https-admserv/conf_bk
server_root/https-server_id/config
server_root/https-server_id/conf_bk

Syntax
Init functions have the following syntax:

Init fn=function  param1="value1"  . . . paramN="valueN"
In the following table Table 3-1, functions are in bold to distinguish them from parameters.

Directives have the following syntax:

\texttt{directive \ value}

**See Also**

Sun ONE Web Server 6.1 \textit{NSAPI Programmer's Guide}

## Init Functions

The following table lists the Init functions available in the magnus.conf file:

<table>
<thead>
<tr>
<th>Function/Parameter</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| \texttt{cindex-init} | \texttt{opts} | (None)        | (optional) is a string of letters specifying the options to activate. Currently there is only one possible option:
\begin{itemize}
\item \texttt{s} tells the server to scan each HTML file in the directory being indexed for the contents of the HTML \texttt{<TITLE>} tag to display in the description field. The \texttt{<TITLE>} tag must be within the first 255 characters of the file.
\end{itemize} |

Changes the default characteristics for fancy indexing.
Table 3-1: magnus.conf Init functions

<table>
<thead>
<tr>
<th>Function/Parameter</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>widths</td>
<td>Comma separated numbers of characters</td>
<td>Minimums required to display column titles</td>
<td>(optional) Specifies the width for each of the four columns in the indexing display: name, last-modified date, size, and description respectively. The final three values can each be set to 0 to turn the display for that column off. The name column cannot be turned off.</td>
</tr>
<tr>
<td>timezone</td>
<td>GMT or local</td>
<td>local</td>
<td>(optional, iPlanet Web Server 4.x only) Indicates whether the last-modified time is shown in local time or in Greenwich Mean Time.</td>
</tr>
<tr>
<td>format</td>
<td>Format for the UNIX strftime() function</td>
<td>%d-%b-%Y %H:%M</td>
<td>(optional, iPlanet Web Server 4.x only) Determines the format of the last modified date display.</td>
</tr>
<tr>
<td>ignore</td>
<td>Wildcard pattern</td>
<td>.*</td>
<td>(optional) Specifies a wildcard pattern for file names the server should ignore while indexing. File names starting with a period (.) are always ignored.</td>
</tr>
<tr>
<td>icon-uri</td>
<td>/mc-icons/</td>
<td></td>
<td>(optional) Specifies the URI prefix the index-common function uses when generating URLs for file icons (.gif files). If icon-uri is different from the default, the pfx2dir function in the NameTrans directive must be changed so that the server can find these icons.</td>
</tr>
</tbody>
</table>
**Table 3-1 magnus.conf Init functions**

<table>
<thead>
<tr>
<th>Function/Parameter</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>define-perf-bucket</code></td>
<td>Creates a performance bucket, which you can use to measure the performance of SAFs in <code>obj.conf</code> (see the <em>Sun ONE Web Server 6.1 NSAPI Programmer's Guide</em>). This function works only if the <code>perf-init</code> function is enabled.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>A name for the bucket, for example <code>cgi-bucket</code>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>A description of what the bucket measures, for example <code>CGI Stats</code>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>dns-cache-init</code></td>
<td>Configures DNS caching.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cache-size</td>
<td>32 to 32768 (32K)</td>
<td>1024</td>
<td>(optional) Specifies how many entries are contained in the cache.</td>
</tr>
<tr>
<td>expire</td>
<td>1 to 31536000 seconds (1 year)</td>
<td>1200 seconds (20 minutes) (optional) specifies how long (in seconds) it takes for a cache entry to expire.</td>
<td></td>
</tr>
<tr>
<td><code>flex-init</code></td>
<td>Initializes the flexible logging system.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>logFileName</td>
<td>A path or file name</td>
<td>The full path to the log file or a file name relative to the server's <code>logs</code> directory. In this example, the log file name is <code>access</code> and the path is <code>/logdir/access</code>: access=&quot;/logdir/access&quot;</td>
<td></td>
</tr>
<tr>
<td>format</td>
<td>Specifies the format of each log entry in the log file. See the <em>Sun ONE Web Server 6.1 NSAPI Programmer's Guide</em> for more information.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 3-1 magnus.conf Init functions

<table>
<thead>
<tr>
<th>Function/Parameter</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buffer-size</td>
<td>Number of bytes</td>
<td>8192</td>
<td>Specifies the size of the global log buffer.</td>
</tr>
<tr>
<td>buffers-per-file</td>
<td>The lower bound is 1. There always needs to be at least one buffer per file. The upper bound is dictated by the number of buffers that exist. The upper bound on the number of buffers that exist can be defined by the num-buffers parameter.</td>
<td>Determined by the server</td>
<td>Specifies the number of buffers for a given log file</td>
</tr>
<tr>
<td>num-buffers</td>
<td></td>
<td>1000</td>
<td>Specifies the maximum number of logging buffers to use.</td>
</tr>
<tr>
<td>flex-rotate-init</td>
<td></td>
<td></td>
<td>Enables rotation for logs.</td>
</tr>
<tr>
<td>rotate-start</td>
<td>A 4-digit string indicating the time in 24-hour format</td>
<td></td>
<td>Indicates the time to start rotation. For example, 0900 indicates 9 am while 1800 indicates 9 pm.</td>
</tr>
<tr>
<td>rotate-interval</td>
<td>Number of minutes</td>
<td></td>
<td>Indicates the number of minutes to elapse between each log rotation.</td>
</tr>
</tbody>
</table>
Table 3-1  magnus.conf Init functions

<table>
<thead>
<tr>
<th>Function/Parameter</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rotate-access</td>
<td>yes, no</td>
<td>yes</td>
<td>(optional) determines whether common-log, flex-log, and record-useragent logs are rotated. For more information, see the Sun ONE Web Server 6.1 NSAPI Programmer’s Guide.</td>
</tr>
<tr>
<td>rotate-error</td>
<td>yes, no</td>
<td>yes</td>
<td>(optional) determines whether error logs are rotated.</td>
</tr>
<tr>
<td>rotate-callback</td>
<td>A path</td>
<td></td>
<td>(optional) specifies the file name of a user-supplied program to execute following log file rotation. The program is passed the post-rotation name of the rotated log file as its parameter.</td>
</tr>
<tr>
<td>init-cgi</td>
<td></td>
<td></td>
<td>Changes the default settings for CGI programs.</td>
</tr>
<tr>
<td>timeout</td>
<td>Number of seconds</td>
<td>300</td>
<td>(optional) specifies how many seconds the server waits for CGI output before terminating the script.</td>
</tr>
<tr>
<td>cgistub-path</td>
<td></td>
<td></td>
<td>(optional) specifies the path to the CGI stub binary. If not specified, iPlanet Web Server looks in the following directories, in the following order, relative to the server instance’s config directory: ../private/Cgistub, then ../../bin/https/bin/Cgistub. For information about installing an suid Cgistub, see the Sun ONE Web Server 6.1 NSAPI Programmer’s Guide.</td>
</tr>
</tbody>
</table>
### Table 3-1  
magnus.conf Init functions

<table>
<thead>
<tr>
<th>Function/Parameter</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>env-variable</td>
<td>(optional) specifies the name and value for an environment variable that the server places into the environment for the CGI.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>init-clf</td>
<td>Initializes the Common Log subsystem.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>logFileName</td>
<td>A path or file name</td>
<td>Specifies either the full path to the log file or a file name relative to the server’s logs directory.</td>
<td></td>
</tr>
<tr>
<td>init-uhome</td>
<td>Loads user home directory information.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pwfile</td>
<td>(optional) specifies the full file system path to a file other than /etc/passwd. If not provided, the default UNIX path (/etc/passwd) is used.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>load-modules</td>
<td>Loads shared libraries into the server.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>shlib</td>
<td>Specifies either the full path to the shared library or dynamic link library or a file name relative to the server configuration directory.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>funcs</td>
<td>A comma separated list with no spaces</td>
<td>A list of the names of the functions in the shared library or dynamic link library to be made available for use by other Init or Service directives. The dash (-) character may be used in place of the underscore (_) character in function names.</td>
<td></td>
</tr>
</tbody>
</table>
### Table 3-1  magnus.conf Init functions

<table>
<thead>
<tr>
<th>Function/Parameter</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NativeThread</td>
<td>yes,no</td>
<td>yes</td>
<td>(optional) specifies which threading model to use. no causes the routines in the library to use user-level threading, yes enables kernel-level threading.</td>
</tr>
<tr>
<td>pool</td>
<td></td>
<td></td>
<td>The name of a custom thread pool, as specified in thread-pool-init.</td>
</tr>
<tr>
<td>nt-console-init</td>
<td></td>
<td></td>
<td>Enables the NT console, which is the command-line shell that displays standard output and error streams.</td>
</tr>
<tr>
<td>stderr</td>
<td>console</td>
<td></td>
<td>Directs error messages to the NT console.</td>
</tr>
<tr>
<td>stdout</td>
<td>console</td>
<td></td>
<td>Directs output to the NT console.</td>
</tr>
<tr>
<td>perf-init</td>
<td></td>
<td></td>
<td>Enables system performance measurement via performance buckets.</td>
</tr>
<tr>
<td>disable</td>
<td>true,false</td>
<td>true</td>
<td>Disables the function when true.</td>
</tr>
<tr>
<td>pool-init</td>
<td></td>
<td></td>
<td>Configures pooled memory allocation.</td>
</tr>
<tr>
<td>free-size</td>
<td>1048576 bytes or less</td>
<td></td>
<td>(optional) maximum size in bytes of free block list.</td>
</tr>
<tr>
<td>disable</td>
<td>true,false</td>
<td>false</td>
<td>(optional) flag to disable the use of pooled memory if true.</td>
</tr>
<tr>
<td>register-http-method</td>
<td></td>
<td></td>
<td>Lets you extend the HTTP protocol by registering new HTTP methods.</td>
</tr>
<tr>
<td>methods</td>
<td>A comma separated list</td>
<td></td>
<td>Names of the methods you are registering.</td>
</tr>
<tr>
<td>stats-init</td>
<td></td>
<td></td>
<td>Enables reporting of performance statistics in XML format.</td>
</tr>
</tbody>
</table>
Table 3-1  magnus.conf Init functions

<table>
<thead>
<tr>
<th>Function/Parameter</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>profiling</td>
<td>yes, no</td>
<td>no</td>
<td>Enables NSAPI performance profiling using buckets. This can also be enabled through perf-init.</td>
</tr>
<tr>
<td>update-interval</td>
<td>1 or greater</td>
<td>5</td>
<td>The period in seconds between statistics updates within the server.</td>
</tr>
<tr>
<td>virtual-servers</td>
<td>1 or greater</td>
<td>1000</td>
<td>The maximum number of virtual servers for which statistics are tracked. This number should be set higher than the number of virtual servers configured.</td>
</tr>
</tbody>
</table>

thread-pool-init

- name
- maxthreads
- minthreads
- queueSize: Number of bytes
- stackSize: Number of bytes

Configures an additional thread pool.

Directives

The following table lists
### Table 3-2  magnus.conf directives

<table>
<thead>
<tr>
<th>Directive</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcceptTimeout</td>
<td>Any number of seconds</td>
<td>30 for servers that don’t use hardware encryption devices and 300 for those that do</td>
<td>Specifies the number of seconds the server waits for data to arrive from the client. If data does not arrive before the timeout expires then the connection is closed.</td>
</tr>
<tr>
<td>ACLCacheLifetime</td>
<td>Any number of seconds</td>
<td>120</td>
<td>Determines the number of seconds before cache entries expire. Each time an entry in the cache is referenced, its age is calculated and checked against ACLCacheLifetime. The entry is not used if its age is greater than or equal to the ACLCacheLifetime. If this value is set to 0, the cache is turned off.</td>
</tr>
<tr>
<td>ACLUserCacheSize</td>
<td></td>
<td>200</td>
<td>Determines the number of users in the User Cache.</td>
</tr>
<tr>
<td>ACLGroupCacheSize</td>
<td></td>
<td>4</td>
<td>Determines how many group IDs can be cached for a single UID/cache entry.</td>
</tr>
<tr>
<td>AsyncDNS</td>
<td>on, off</td>
<td>off</td>
<td>Specifies whether asynchronous DNS is allowed.</td>
</tr>
<tr>
<td>CGIExpirationTimeout</td>
<td>Any number of seconds</td>
<td>300 (5 minutes) recommended</td>
<td>Specifies the maximum time in seconds that CGI processes are allowed to run before being killed.</td>
</tr>
<tr>
<td>CGISStubIdleTimeout</td>
<td>Any number of seconds</td>
<td>30</td>
<td>Causes the server to kill any CGISStub processes that have been idle for the number of seconds set by this directive. Once the number of processes is at MinCGISstubs, the server does not kill any more processes.</td>
</tr>
<tr>
<td>CGIWaitPid</td>
<td>on, off</td>
<td>on</td>
<td>(UNIX only) makes the action for the SIGCHLD signal the system default action for the signal. Makes the SHTML engine wait explicitly on its exec cmd child processes.</td>
</tr>
</tbody>
</table>
### Table 3-2 magnus.conf directives

<table>
<thead>
<tr>
<th>Directive</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChildRestartCallback</td>
<td>on, off, yes, no,</td>
<td>no</td>
<td>Forces the callback of NSAPI functions that were registered using the daemon_atrestart function when the server is restarting or shutting down.</td>
</tr>
<tr>
<td></td>
<td>true, false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>Any number of bytes</td>
<td>8192</td>
<td>Determines the default buffer size for “un-chunking” request data.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>Any number of seconds</td>
<td>60 (1 minute)</td>
<td>Determines the default timeout for “un-chunking” request data.</td>
</tr>
<tr>
<td>ConnQueueSize</td>
<td>Any number of connections</td>
<td>4096</td>
<td>Specifies the number of outstanding (yet to be serviced) connections that the web server can have.</td>
</tr>
<tr>
<td></td>
<td>(including 0)</td>
<td></td>
<td>This setting can have performance implications. For more information, see the Sun ONE Web Server 6.1 Performance Tuning, Sizing, and Scaling Guide.</td>
</tr>
<tr>
<td>DefaultLanguage</td>
<td>en (English), fr</td>
<td>en</td>
<td>Specifies the default language for the server. The default language is used for both the client responses and administration.</td>
</tr>
<tr>
<td></td>
<td>(French), de (German), ja (Japanese)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DNS</td>
<td>on, off</td>
<td>on</td>
<td>Specifies whether the server performs DNS lookups on clients that access the server.</td>
</tr>
<tr>
<td>ErrorLogDateFormat</td>
<td>See the manual page</td>
<td>%d/%b/%Y:%H:%M:%S</td>
<td>The date format for the error log.</td>
</tr>
<tr>
<td></td>
<td>for the C library</td>
<td></td>
<td>function strftime</td>
</tr>
<tr>
<td>ExtraPath</td>
<td>A path</td>
<td>(none)</td>
<td>Appends the specified directory name to the PATH environment variable. This is used for configuring Java on Windows NT. There is no default value; you must specify a value.</td>
</tr>
<tr>
<td>Directive</td>
<td>Allowed Values</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Favicon</td>
<td>On / Off</td>
<td>on</td>
<td>Provides the web server administrator the ability to disable or change the icon which appears in the web address book or favorites list on Internet Explorer browsers (so, favicon translates as favorite icon).</td>
</tr>
<tr>
<td>HeaderBufferSize</td>
<td>Any number of bytes</td>
<td>8192 (8 KB)</td>
<td>The size (in bytes) of the buffer used by each of the request processing threads for reading the request data from the client. The maximum number of request processing threads is controlled by the RqThrottle setting.</td>
</tr>
<tr>
<td>HTTPVersion</td>
<td>m.n; m is the major version number and n the minor version number</td>
<td>1.1</td>
<td>The current HTTP version used by the server.</td>
</tr>
<tr>
<td>KeepAliveQueryMaxSleepTime</td>
<td>100</td>
<td>On lightly loaded systems that primarily service keep-alive connections, you can lower this number to enhance performance. However doing so can increase CPU usage.</td>
<td>This directive specifies an upper limit to the time slept (in milliseconds) after polling keep-alive connections for further requests.</td>
</tr>
</tbody>
</table>
**Table 3-2** magnus.conf directives

<table>
<thead>
<tr>
<th>Directive</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeepAliveQueryMeanTime</td>
<td>100 is appropriate for almost all installations.</td>
<td>Note that CPU usage will increase with lower KeepAliveQueryMeanTime values.</td>
<td>This directive specifies the desired keep-alive latency in milliseconds.</td>
</tr>
<tr>
<td>KeepAliveThreads</td>
<td>Any number of threads</td>
<td>1</td>
<td>Specifies the number of threads in the keep-alive subsystem. It is recommended that this number be a small multiple of the number of processors on the system.</td>
</tr>
<tr>
<td>KeepAliveTimeout</td>
<td>300 seconds maximum</td>
<td>30</td>
<td>Determines the maximum time that the server holds open an HTTP Keep-Alive connection or a persistent connection between the client and the server.</td>
</tr>
<tr>
<td>KernelThreads</td>
<td>0 (off), 1 (on)</td>
<td>0 (off)</td>
<td>If on, ensures that the server uses only kernel-level threads, not user-level threads. If off, uses only user-level threads.</td>
</tr>
<tr>
<td>ListenQ</td>
<td>Ranges are platform-specific</td>
<td>4096 (AIX), 200 (NT), 128 (all others)</td>
<td>Defines the number of incoming connections for a server socket.</td>
</tr>
<tr>
<td>LogFlushInterval</td>
<td>Any number of seconds</td>
<td>30</td>
<td>Determines the log flush interval, in seconds, of the log flush thread.</td>
</tr>
<tr>
<td>MaxCGIStubs</td>
<td>Any number of CGI stubs</td>
<td>10</td>
<td>Controls the maximum number of CGIStubs processes the server can spawn. This is the maximum concurrent CGIStub processes in execution, not the maximum number of pending requests.</td>
</tr>
</tbody>
</table>
Table 3-2  magnus.conf directives

<table>
<thead>
<tr>
<th>Directive</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxKeepAliveConnections</td>
<td>0 - 32768</td>
<td></td>
<td>Specifies the maximum number of Keep-Alive and persistent connections that the server can have open simultaneously.</td>
</tr>
<tr>
<td>MaxProcs</td>
<td>1</td>
<td></td>
<td>(UNIX only) Specifies the maximum number of processes that the server can have running simultaneously.</td>
</tr>
<tr>
<td>MaxRqHeaders</td>
<td>1 - 512</td>
<td>64</td>
<td>Specifies the maximum number of header lines in a request.</td>
</tr>
<tr>
<td>MinCGIStubs</td>
<td>Any number less than MaxCGIStubs</td>
<td>2</td>
<td>Controls the number of processes that are started by default.</td>
</tr>
<tr>
<td>NativePoolMaxThreads</td>
<td>Any number of threads</td>
<td></td>
<td>Determines the maximum number of threads in the native (kernel) thread pool.</td>
</tr>
<tr>
<td>NativePoolMinThreads</td>
<td>Any number of threads</td>
<td>1</td>
<td>Determines the minimum number of threads in the native (kernel) thread pool.</td>
</tr>
<tr>
<td>NativePoolQueueSize</td>
<td>Any nonnegative number</td>
<td>0</td>
<td>Determines the number of threads that can wait in the queue for the thread pool.</td>
</tr>
<tr>
<td>NativePoolStackSize</td>
<td>Any nonnegative number</td>
<td>0</td>
<td>Determines the stack size of each thread in the native (kernel) thread pool.</td>
</tr>
<tr>
<td>PidLog</td>
<td>A valid path to a file</td>
<td>(none)</td>
<td>Specifies a file in which to record the process ID (pid) of the base server process.</td>
</tr>
<tr>
<td>PostThreadsEarly</td>
<td>1 (on), 0 (off)</td>
<td>0 (off)</td>
<td>If on, checks whether the minimum number of threads are available at a socket after accepting a connection but before sending the response to the request.</td>
</tr>
<tr>
<td>RcvBufSize</td>
<td>Range is platform-specific</td>
<td>0 (uses platform-specific default)</td>
<td>Controls the size of the receive buffer at the server’s sockets.</td>
</tr>
</tbody>
</table>
Summary of Init Functions and Directives in magnus.conf

Table 3-2  magnus.conf directives

<table>
<thead>
<tr>
<th>Directive</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RqThrottle</td>
<td>Any number of requests (including 0)</td>
<td></td>
<td>Specifies the maximum number of simultaneous request processing threads that the server can handle simultaneously per socket. This setting can have performance implications. For more information, see the Sun ONE Web Server 6.1 Performance Tuning, Sizing, and Scaling Guide.</td>
</tr>
<tr>
<td>RqThrottleMin</td>
<td>Any number less than RqThrottle</td>
<td></td>
<td>Specifies the number of request processing threads that are created when the server is started. As the load on the server increases, more request processing threads are created (up to a maximum of RqThrottle threads).</td>
</tr>
<tr>
<td>Security</td>
<td>on, off</td>
<td>off</td>
<td>Globally enables or disables SSL by making certificates available to the server instance. Must be on for virtual servers to use SSL.</td>
</tr>
<tr>
<td>SndBufSize</td>
<td>Range is platform-specific</td>
<td>0 (uses platform-specific default)</td>
<td>Controls the size of the send buffer at the server’s sockets.</td>
</tr>
<tr>
<td>SSL3SessionTimeout</td>
<td>5 - 86400 (24 hours)</td>
<td>86400</td>
<td>The number of seconds until a cached SSL3 session becomes invalid.</td>
</tr>
<tr>
<td>SSLCacheEntries</td>
<td>A non-negative integer</td>
<td>10000 (used if 0 is specified)</td>
<td>Specifies the number of SSL sessions that can be cached. There is no upper limit.</td>
</tr>
<tr>
<td>SSLClientAuthDataLimit</td>
<td>Number of Bytes</td>
<td>1048576 (1MB)</td>
<td>Specifies the maximum amount of application data that is buffered during the client certificate handshake phase.</td>
</tr>
<tr>
<td>SSLClientAuthTimeout</td>
<td>Any number of seconds</td>
<td>60</td>
<td>Specifies the number of seconds after which the client certificate handshake phase times out.</td>
</tr>
</tbody>
</table>
### Table 3-2  magnus.conf directives

<table>
<thead>
<tr>
<th>Directive</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSLSessionTimeout</td>
<td>5 - 100</td>
<td>100</td>
<td>Specifies the number of seconds until a cached SSL2 session becomes invalid.</td>
</tr>
<tr>
<td>StackSize</td>
<td>Number of Bytes</td>
<td>The most favorable machine-specific stack size.</td>
<td>Determines the maximum stack size for each request handling thread.</td>
</tr>
<tr>
<td>StrictHttpHeaders</td>
<td>on, off</td>
<td>off</td>
<td>If on, rejects connections that include inappropriately duplicated headers.</td>
</tr>
<tr>
<td>TempDir</td>
<td>A path</td>
<td>/tmp (UNIX)</td>
<td>Specifies the directory the server uses for its temporary files. On UNIX, this directory should be owned by, and writable by, the user the server runs as.</td>
</tr>
<tr>
<td>TempDirSecurity</td>
<td>on, off</td>
<td>on</td>
<td>Determines whether the server checks if the TempDir directory is secure. On UNIX, specifying TempDirSecurity off allows the server to use /tmp as a temporary directory.</td>
</tr>
<tr>
<td>TerminateTimeout</td>
<td>Any number of seconds</td>
<td>30</td>
<td>Specifies the time in seconds that the server waits for all existing connections to terminate before it shuts down.</td>
</tr>
<tr>
<td>ThreadIncrement</td>
<td>Any number of threads</td>
<td>10</td>
<td>The number of additional or new request processing threads created to handle an increase in the load on the server.</td>
</tr>
<tr>
<td>Umask</td>
<td>A standard UNIX umask value</td>
<td>(none)</td>
<td>UNIX only: Specifies the umask value used by the NSAPI functions System_fopenWA() and System_fopenRW() to open files in different modes.</td>
</tr>
</tbody>
</table>


**Table 3-2**  magnus.conf directives

<table>
<thead>
<tr>
<th>Directive</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UseNativePoll</td>
<td>1 (on), 0 (off)</td>
<td>1 (on)</td>
<td>Uses a platform-specific poll interface when set to 1 (on). Uses the NSPR poll interface in the KeepAlive subsystem when set to 0 (off).</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>Any number of bytes</td>
<td>8192 (8 KB)</td>
<td>Determines the default output stream buffer size for the net_read and netbuf_grab NSAPI functions.</td>
</tr>
<tr>
<td>User</td>
<td>A login name, 8 characters or less</td>
<td>(none)</td>
<td>(Windows NT) specifies the user account the server runs with, allowing you to restrict or enable system features for the server. (UNIX) 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.</td>
</tr>
<tr>
<td>WinCGITimeout</td>
<td>Any number of seconds</td>
<td>60</td>
<td>WinCGI processes that take longer than this value are terminated when this timeout expires.</td>
</tr>
</tbody>
</table>
Chapter 4

Predefined SAFs in obj.conf

The obj.conf configuration file contains directives that instruct the Sun™ Open Net Environment (Sun ONE) Web Server how to handle HTTP and HTTPS requests from clients and service web server content such as native server plugins and CGI programs. You can modify and extend the request-handling process by adding or changing the instructions in obj.conf.

All obj.conf files are located in the instance_dir/config directory, where instance_dir is the path to the installation directory of the server instance. There is one obj.conf file for each virtual server class, unless several virtual server classes are configured to share an obj.conf file. Whenever this guide refers to "the obj.conf file," it refers to all obj.conf files or to the obj.conf file for the virtual server class being described.

By default, the obj.conf file for the initial virtual server class is named obj.conf, and the obj.conf files for the administrator-defined virtual server classes are named virtual_server_class_id.obj.conf. Editing one of these files directly or through the Administration interface changes the configuration of a virtual server class.

This chapter describes the standard directives and predefined Server Application Functions (SAFs) that are used in the obj.conf file to give instructions to the server. For details about the syntax and use of the obj.conf file, refer to the Sun ONE Web Server 6.1 NSAPI Programmer’s Guide.

Each SAF has its own arguments, which are passed to it by a directive in obj.conf. Every SAF is also passed additional arguments that contain information about the request (such as what resource was requested and what kind of client requested it), and any other server variables created or modified by SAFs called by previously invoked directives. Each SAF may examine, modify, or create server variables. Each SAF returns a result code that tells the server whether it succeeded, did nothing, or failed.
This chapter includes functions that are part of the core functionality of Sun ONE Web Server. It does not include functions that are available only if additional components, such as server-parsed HTML, are enabled.

This chapter covers the following stages:

- AuthTrans
- NameTrans
- PathCheck
- ObjectType
- Input
- Output
- Service
- AddLog
- Error

For an alphabetical list of predefined SAFs, see Appendix A on page 375.

The following table lists the SAFs that can be used with each directive.

<table>
<thead>
<tr>
<th>Directive</th>
<th>Server Application Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthTrans</td>
<td>basic-auth, basic-ncsa, get-sslid, match-browser, qos-handler, set-variable</td>
</tr>
<tr>
<td>NameTrans</td>
<td>assign-name, document-root, home-page, match-browser, ntrans-dav, ntrans-j2ee, pfx2dir, redirect, set-variable, strip-params, unix-home</td>
</tr>
</tbody>
</table>
### Table 4-1 Available Server Application Functions (SAFs) Per Directive

<table>
<thead>
<tr>
<th>Directive</th>
<th>Server Application Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PathCheck</strong></td>
<td>check-acl, deny-existence, find-index, find-links, find-pathinfo, get-client-cert, load-config, match-browser, nt-uri-clean, ntcgicheck, require-auth, set-variable, set-virtual-index, ssl-check, ssl-logout, unix-uri-clean</td>
</tr>
<tr>
<td><strong>ObjectType</strong></td>
<td>force-type, match-browser, set-default-type, set-variable, shtml-hacktype, type-by-exp, type-by-extension</td>
</tr>
<tr>
<td><strong>Input</strong></td>
<td>insert-filter, match-browser, remove-filter, set-variable</td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td>insert-filter, match-browser, remove-filter, set-variable</td>
</tr>
</tbody>
</table>
Table 4-1  Available Server Application Functions (SAFs) Per Directive

<table>
<thead>
<tr>
<th>Directive</th>
<th>Server Application Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>add-footer</td>
</tr>
<tr>
<td></td>
<td>add-header</td>
</tr>
<tr>
<td></td>
<td>append-trailer</td>
</tr>
<tr>
<td></td>
<td>imagemap</td>
</tr>
<tr>
<td></td>
<td>index-common</td>
</tr>
<tr>
<td></td>
<td>index-simple</td>
</tr>
<tr>
<td></td>
<td>key-tooosmall</td>
</tr>
<tr>
<td></td>
<td>list-dir</td>
</tr>
<tr>
<td></td>
<td>make-dir</td>
</tr>
<tr>
<td></td>
<td>match-browser</td>
</tr>
<tr>
<td></td>
<td>query-handler</td>
</tr>
<tr>
<td></td>
<td>remove-dir</td>
</tr>
<tr>
<td></td>
<td>remove-file</td>
</tr>
<tr>
<td></td>
<td>remove-filter</td>
</tr>
<tr>
<td></td>
<td>rename-file</td>
</tr>
<tr>
<td></td>
<td>send-cgi</td>
</tr>
<tr>
<td></td>
<td>send-error</td>
</tr>
<tr>
<td></td>
<td>send-file</td>
</tr>
<tr>
<td></td>
<td>send-range</td>
</tr>
<tr>
<td></td>
<td>send-shellcgi</td>
</tr>
<tr>
<td></td>
<td>send-wincgi</td>
</tr>
<tr>
<td></td>
<td>service-dump</td>
</tr>
<tr>
<td></td>
<td>service-j2ee</td>
</tr>
<tr>
<td></td>
<td>service-trace</td>
</tr>
<tr>
<td></td>
<td>set-variable</td>
</tr>
<tr>
<td></td>
<td>shtml_send</td>
</tr>
<tr>
<td></td>
<td>stats-xml</td>
</tr>
<tr>
<td></td>
<td>upload-file</td>
</tr>
<tr>
<td>AddLog</td>
<td>common-log</td>
</tr>
<tr>
<td></td>
<td>flex-log</td>
</tr>
<tr>
<td></td>
<td>match-browser</td>
</tr>
<tr>
<td></td>
<td>record-useragent</td>
</tr>
<tr>
<td></td>
<td>set-variable</td>
</tr>
<tr>
<td>Error</td>
<td>error-j2ee</td>
</tr>
<tr>
<td></td>
<td>match-browser</td>
</tr>
<tr>
<td></td>
<td>qos-error</td>
</tr>
<tr>
<td></td>
<td>query-handler</td>
</tr>
<tr>
<td></td>
<td>remove-filter</td>
</tr>
<tr>
<td></td>
<td>send-error</td>
</tr>
<tr>
<td></td>
<td>set-variable</td>
</tr>
</tbody>
</table>


The bucket Parameter

The following performance buckets are predefined in Sun ONE Web Server:

- The **default-bucket** records statistics for the functions not associated with any user-defined or built-in bucket.
- The **all-requests** bucket records .perf statistics for all NSAPI SAFs, including those in the default-bucket.

You can define additional performance buckets in the `magnus.conf` file (see the `perf-init` and `define-perf-bucket` functions).

You can measure the performance of any SAF in `obj.conf` by adding a `bucket=bucket-name` parameter to the function, for example `bucket=cache-bucket`.

To list the performance statistics, use the `service-dump` Service function.

As an alternative, you can use the `stats-xml` Service function to generate performance statistics; use of buckets is optional.

For more information about performance buckets, see the Sun ONE Web Server 6.1 *Performance Tuning, Sizing, and Scaling Guide*.

AuthTrans

AuthTrans stands for Authorization Translation. AuthTrans directives give the server instructions for checking authorization before allowing a client to access resources. AuthTrans directives work in conjunction with PathCheck directives. Generally, an AuthTrans function checks if the user name and password associated with the request are acceptable, but it does not allow or deny access to the request; that is left to a PathCheck function.

The server handles the authorization of client users in two steps:

- **AuthTrans** validates authorization information sent by the client in the Authorization header.
- **PathCheck** checks that the authorized user is allowed access to the requested resource.

The authorization process is split into two steps so that multiple authorization schemes can be easily incorporated, and to provide the flexibility to have resources that record authorization information, but do not require it.
AuthTrans functions get the user name and password from the headers associated with the request. When a client initially makes a request, the user name and password are unknown so the AuthTrans functions and PathCheck functions work together to reject the request, since they can’t validate the user name and password. When the client receives the rejection, its usual response is to present a dialog box asking for the user name and password to enter the appropriate realm, and then the client submits the request again, this time including the user name and password in the headers.

If there is more than one AuthTrans directive in obj.conf, each function is executed in order until one succeeds in authorizing the user.

The following AuthTrans-class functions are described in detail in this section:

- **basic-auth** calls a custom function to verify user name and password. Optionally determines the user’s group.
- **basic-ncsa** verifies user name and password against an NCSA-style or system DBM database. Optionally determines the user’s group.
- **get-sslid** retrieves a string that is unique to the current SSL session and stores it as the ssl-id variable in the Session->client parameter block.
- **match-browser** matches specific strings in the User-Agent string supplied by the browser, and then modifies the behavior of Sun ONE Web Server based upon the results by setting values for specified variables.
- **qos-handler** handles the current quality of service statistics.
- **set-variable** enables you to change server settings based upon conditional information in a request, and to manipulate variables in parameter blocks by using specific commands.

**basic-auth**

Applicable in AuthTrans-class directives.

The basic-auth function calls a custom function to verify authorization information sent by the client. The Authorization header is sent as part of the basic server authorization scheme.

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

**Parameters**

The following table describes parameters for the basic-auth function.
Table 4-2 basic-auth parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth-type</td>
<td>Specifies the type of authorization to be used. This should always be basic.</td>
</tr>
<tr>
<td>userdb</td>
<td>(Optional) Specifies the full path and file name of the user database to be used for user verification. This parameter will be passed to the user function.</td>
</tr>
<tr>
<td>userfn</td>
<td>Name of the user custom function to verify authorization. This function must have been previously loaded with load-modules. It has the same interface as all of the SAFs, but it is called with the user name (user), password (pw), user database (userdb), and group database (groupdb) if supplied, in the pb parameter. The user function should check the name and password using the database and return REQ_NOACTION if they are not valid. It should return REQ_PROCEED if the name and password are valid. The basic-auth function will then add auth-type, auth-user (user), auth-db (userdb), and auth-password (pw, Windows only) to the rq-&gt;vars pblock.</td>
</tr>
<tr>
<td>groupdb</td>
<td>(Optional) Specifies the full path and file name of the user database. This parameter will be passed to the group function.</td>
</tr>
<tr>
<td>groupfn</td>
<td>(Optional) Name of the group custom function that must have been previously loaded with load-modules. It has the same interface as all of the SAFs, but it is called with the user name (user), password (pw), user database (userdb), and group database (groupdb) in the pb parameter. It also has access to the auth-type, auth-user (user), auth-db (userdb), and auth-password (pw, Windows only) parameters in the rq-&gt;vars pblock. The group function should determine the user’s group using the group database, add it to rq-&gt;vars as auth-group, and return REQ_PROCEED if found. It should return REQ_NOACTION if the user’s group is not found.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Examples

In magnus.conf:
In `obj.conf`:

```ini
[Init]
fn=load-modules
shlib=/path/to/mycustomauth.so
funcs=hardcoded_auth
```

```ini
[AuthTrans]
fn=basic-auth
auth-type=basic
userfn=hardcoded_auth
```

```ini
[PathCheck]
fn=require-auth
auth-type=basic
realm="Marketing Plans"
```

**See Also**

`require-auth`

### basic-ncsa

Applicable in `AuthTrans-class directives`.

The `basic-ncsa` function verifies authorization information sent by the client against a database. The `Authorization` header is sent as part of the basic server authorization scheme.

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

**Parameters**

The following table describes parameters for the `basic-ncsa` function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>auth-type</code></td>
<td>Specifies the type of authorization to be used. This should always be <code>basic</code>.</td>
</tr>
<tr>
<td><code>dbm</code></td>
<td>(Optional) Specifies the full path and base file name 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 the <code>userfile</code> parameter as well.</td>
</tr>
</tbody>
</table>
Examples

```
AuthTrans fn=basic-ncsa auth-type=basic
dbm=/sun/server61/userdb/rs
PathCheck fn=require-auth auth-type=basic realm="Marketing Plans"
AuthTrans fn=basic-ncsa auth-type=basic
PathCheck fn=require-auth auth-type=basic realm="Marketing Plans"
```

See Also
require-auth

get-sslid

Applicable in AuthTrans-class directives.

**NOTE** This function is provided for backward compatibility only. The functionality of get-sslid has been incorporated into the standard processing of an SSL connection.

The get-sslid function retrieves a string that is unique to the current SSL session, and stores it as the ssl-id variable in the Session->client parameter block.
If the variable $ssl-id$ is present when a CGI is invoked, it is passed to the CGI as the $HTTPS_SESSIONID$ environment variable.

The $get-sslid$ function has no parameters and always returns $REQ_NOACTION$. It has no effect if SSL is not enabled.

**Parameters**
The following table describes parameters for the $get-sslid$ function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

## match-browser

Applicable in all stage directives.

The $match-browser$ SAF matches specific strings in the User-Agent string supplied by the browser, and then modifies the behavior of Sun ONE Web Server based upon the results by setting values for specified variables.

**Syntax**

```
stage fn="match-browser" browser="string" name="value" [name="value" ...]
```

**Parameters**
The following table describes parameter values for the $match-browser$ function.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$stage$</td>
<td>Stage directive used in obj.conf processing (NameTrans, PathCheck, and so on). The $match-browser$ function is applicable in all stage directives.</td>
</tr>
<tr>
<td>$string$</td>
<td>Wildcard pattern to compare against the User-Agent header (for example, &quot;<em>Mozilla</em>&quot;).</td>
</tr>
<tr>
<td>$name$</td>
<td>Variable to be changed. The $match-browser$ SAF indirectly invokes the $set-variable$ SAF. For a list of valid variables, see $set-variable$.</td>
</tr>
<tr>
<td>$value$</td>
<td>New value for the specified variable.</td>
</tr>
</tbody>
</table>
Example
The following AuthTrans directive instructs Sun ONE Web Server to do as follows when the browser’s User-Agent header contains the string Broken or broken. The server will:

- Not send the SSL3 and TLS close_notify packet (see “ssl-unclean-shutdown” on page 132).
- Not honor requests for HTTP Keep-Alive (see “keep-alive” on page 131).
- Use the HTTP/1.0 protocol rather than HTTP/1.1 (see “http-downgrade” on page 131).

```xml
AuthTrans fn="match-browser"
  browser="*\[Bb\]roken*"
  ssl-unclean-shutdown="true"
  keep-alive="disabled"
  http-downgrade="1.0"
```

See Also
set-variable

qos-handler
Applicable in AuthTrans-class directives.

The qos-handler function examines the current quality of service statistics for the virtual server, virtual server class, and global server, logs the statistics, and enforces the QOS parameters by returning an error. This must be the first AuthTrans function configured in the default object in order to work properly.

The code for this SAF is one of the examples provided in the Sun ONE Web Server 6.1 NSAPI Programmer’s Guide.

For more information, see the Sun ONE Web Server 6.1 Performance Tuning, Sizing, and Scaling Guide.

Parameters
The following table describes parameters for the qos-handler function.
set-variable

Applicable in all stage directives.

The `set-variable` function enables you to change server settings based upon conditional information in a request. It can also be used to manipulate variables in parameter blocks with the following commands:

- **insert-pblock**="name=value"
  
  Adds a new value to the specified `pblock`.

- **set-pblock**="name=value"
  
  Sets a new value in the specified `pblock`, replacing any existing value(s) with the same name.

- **remove-pblock**="name"
  
  Removes all values with the given name from the specified `pblock`.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucket</td>
<td>(Optional) Common to all <code>obj.conf</code> functions.</td>
</tr>
</tbody>
</table>

**Example**

```
AuthTrans fn=qos-handler
```

**See Also**

`qos-error`

**NOTE**

For more information about parameter blocks, see the Sun ONE Web Server 6.1 NSAPI Programmer’s Guide.
Syntax

\[\text{stage fn}=\text{\texttt{set-variable}} \{\text{insert} | \text{set} | \text{remove}\} \text{-pblock} = \text{\texttt{name=value}} \ldots \]
\[\text{name} = \text{\texttt{value}} \ldots \]

Parameters

The following table describes parameter values for the \texttt{set-variable} function.
Table 4-7  set-variable parameter values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pblob</td>
<td>One of the following Session/Request parameter block names:</td>
</tr>
<tr>
<td></td>
<td>- client: Contains the IP address of the client machine and the DNS name of the remote machine. For more information, see the description of the Session-&gt;client function in the “Data Structure Reference” chapter of the Sun ONE Web Server 6.1 NSAPI Programmer’s Guide.</td>
</tr>
<tr>
<td></td>
<td>- vars: Contains the server’s working variables, which includes anything not specifically found in the reqpb, headers, or srvhds pblocks. The contents of this pblock differ, depending upon the specific request and the type of SAF.</td>
</tr>
<tr>
<td></td>
<td>- reqpb: Contains elements of the HTTP request, which includes the HTTP method (GET, POST, and so on), the URI, the protocol (generally HTTP/1.0), and the query string. This pblock doesn’t usually change during the request-response process.</td>
</tr>
<tr>
<td></td>
<td>- headers: Contains all the request headers (such as User-Agent, If-Modified-Since, and so on) received from the client in the HTTP request. This pblock doesn’t usually change during the request-response process. For more information about request headers, see the “Hypertext Transfer Protocol” chapter of the Sun ONE Web Server 6.1 NSAPI Programmer’s Guide.</td>
</tr>
<tr>
<td></td>
<td>- srvhds: Contains the response headers (such as Server, Date, Content-type, Content-length, and so on) that are to be sent to the client in the HTTP response. For more information about response headers, see the “Hypertext Transfer Protocol” chapter of the Sun ONE Web Server 6.1 NSAPI Programmer’s Guide.</td>
</tr>
<tr>
<td>name</td>
<td>The variable to set.</td>
</tr>
<tr>
<td>value</td>
<td>The string assigned to the variable specified by name.</td>
</tr>
</tbody>
</table>

Note: For more information about parameter blocks, see the Sun ONE Web Server 6.1 NSAPI Programmer’s Guide.
Variables
The following tables lists variables supported by the set-variable SAF.

Table 4-8  Supported Variables

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abort</td>
<td>A value of true indicates the result code should be set to REQ_ABORTED. Setting the result code to REQ_ABORTED will abort the current request and send an error to the browser. For information about result codes, see the “Creating Custom SAFs” chapter of the Sun ONE Web Server 6.1 NSAPI Programmer’s Guide.</td>
</tr>
<tr>
<td>error</td>
<td>Sets the error code to be returned in the event of an aborted browser request.</td>
</tr>
<tr>
<td>escape</td>
<td>A boolean value signifying whether a URL should be escaped using util_uri_escape. For information about util_uri_escape, see the “NSAPI Function Reference” chapter of the Sun ONE Web Server 6.1 NSAPI Programmer’s Guide.</td>
</tr>
<tr>
<td>find-pathinfo-forward</td>
<td>Path information after the file name in a URI. See find-pathinfo.</td>
</tr>
<tr>
<td>http-downgrade</td>
<td>HTTP version number (for example, 1.0).</td>
</tr>
<tr>
<td>http-upgrade</td>
<td>HTTP version number (for example, 1.0).</td>
</tr>
<tr>
<td>keep-alive</td>
<td>A boolean value that establishes whether a keep-alive request from a browser will be honored.</td>
</tr>
<tr>
<td>name</td>
<td>Specifies an additional named object in the obj.conf file whose directives will be applied to this request. See also assign-name.</td>
</tr>
<tr>
<td>noaction</td>
<td>A value of true indicates the result code should be set to REQ_NOACTION. For AuthTrans, NameTrans, Service, and Error stage SAFs, setting the result code to REQ_NOACTION indicates that subsequent SAFs in that stage should be allowed to execute. For information about result codes, see the “Creating Custom SAFs” chapter of the Sun ONE Web Server 6.1 NSAPI Programmer’s Guide.</td>
</tr>
<tr>
<td>nostat</td>
<td>Causes the server not to perform the stat() function for a URL when possible. See also assign-name.</td>
</tr>
<tr>
<td>senthdrs</td>
<td>A boolean value that indicates whether HTTP response headers have been sent to the client.</td>
</tr>
</tbody>
</table>
Examples

- To deny HTTP keep-alive requests for a specific server class (while still honoring keep-alive requests for the other classes), add this AuthTrans directive to the obj.conf for the server class, and set the variable keep-alive to disabled:

  AuthTrans fn="set-variable" keep-alive="disabled"

To cause that same server class to use HTTP/1.0 while the rest of the server classes use HTTP/1.1, the AuthTrans directive would be:

  AuthTrans fn="set-variable" keep-alive="disabled" http-downgrade="true"

- To insert an HTTP header into each response, add a NameTrans directive to obj.conf, using the insert-pblock command and specifying srvhdrs as your Session/Request parameter block.

  For example, to insert the HTTP header P3P, you would add the following line to each request:

  NameTrans fn="set-variable" insert-srvhdrs="P3P"
To terminate processing a request based upon certain URIs, use a <Client> tag to specify the URIs and an AuthTrans directive that sets the variable abort to true when there is a match. Your <Client> tag would be comparable to the following:

```xml
<Client uri="*(system32|root.exe)*">
  AuthTrans fn="set-variable" abort="true"
</Client>
```

See Also
match-browser

NameTrans

NameTrans stands for Name Translation. NameTrans directives translate virtual URLs to physical directories on your server. For example, the URL

http://www.test.com/some/file.html

could be translated to the full file system path

/usr/Sun/WebServer61/server1/docs/some/file.html

NameTrans directives should appear in the default object. If there is more than one NameTrans directive in an object, the server executes each one in order until one succeeds.

The following NameTrans-class functions are described in detail in this section:

- **assign-name** tells the server to process directives in a named object.
- **document-root** translates a URL into a file system path by replacing the http://server-name/ part of the requested resource with the document root directory.
- **home-page** translates a request for the server’s root home page (/) to a specific file.
- **match-browser** matches specific strings in the User-Agent string supplied by the browser, and then modifies the behavior of Sun ONE Web Server based upon the results by setting values for specified variables.
- **ntrans-dav** determines whether a request should be handled by the WebDAV subsystem and if so, creates a dav objectset.
- **ntrans-j2ee** determines whether a request maps to a Java™ technology-based web application context.
- **pfx2dir** translates any URL beginning with a given prefix to a file system directory and optionally enables directives in an additional named object.
- **redirect** redirects the client to a different URL.
- **set-variable** enables you to change server settings based upon conditional information in a request, and to manipulate variables in parameter blocks by using specific commands.
- **strip-params** removes embedded semicolon-delimited parameters from the path.
- **unix-home** translates a URL to a specified directory within a user’s home directory.

### assign-name

Applicable in **NameTrans**-class directives.

The **assign-name** function specifies the name of an object in **obj.conf** that matches the current request. The server then processes the directives in the named object in preference to the ones in the default object.

For example, consider the following directive in the default object:

```
NameTrans fn=assign-name name=personnel from=/personnel
```

Let’s suppose the server receives a request for `http://server-name/personnel`. After processing this **NameTrans** directive, the server looks for an object named `personnel` in **obj.conf**, and continues by processing the directives in the `personnel` object.

The **assign-name** function always returns **REQ_NOACTION**.

#### Parameters

The following table describes parameters for the **assign-name** function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>from</code></td>
<td>Wildcard pattern that specifies the path to be affected.</td>
</tr>
<tr>
<td><code>name</code></td>
<td>Specifies an additional named object in <strong>obj.conf</strong> whose directives will be applied to this request.</td>
</tr>
</tbody>
</table>
find-pathinfo-forward  (Optional) Makes the server look for the PATHINFO forward in the path right after the \texttt{ntrans-base} instead of backward from the end of path as the server function \texttt{assign-name} does by default.

The value you assign to this parameter is ignored. If you do not wish to use this parameter, leave it out.

The \texttt{find-pathinfo-forward} parameter is ignored if the \texttt{ntrans-base} parameter is not set in \texttt{rq->vars}. By default, \texttt{ntrans-base} is set.

This feature can improve performance for certain URLs by reducing the number of stats performed.

nostat  (Optional) Prevents the server from performing a stat on a specified URL whenever possible.

The effect of \texttt{nostat=}"virtual-path" in the NameTrans function \texttt{assign-name} is that the server assumes that a stat on the specified \textit{virtual-path} will fail. Therefore, use \texttt{nostat} only when the path of the \textit{virtual-path} does not exist on the system, for example, for NSAPI plugin URLs, to improve performance by avoiding unnecessary stats on those URLs.

When the default PathCheck server functions are used, the server does not stat for the paths
\texttt{/ntrans-base/virtual-path} and
\texttt{/ntrans-base/virtual-path/*} if \texttt{ntrans-base} is set (the default condition); it does not stat for the URLs
\texttt{/virtual-path} and \texttt{/virtual-path/*} if \texttt{ntrans-base} is not set.

bucket  (Optional) Common to all \texttt{obj.conf} functions.
Example

```
# This NameTrans directive is in the default object.
NameTrans fn=assign-name name=personnel from=/a/b/c/pers ...
<Object name=personnel>
...additional directives..
</Object>

NameTrans fn="assign-name" from="/perf" find-pathinfo-forward=""
name="perf"

NameTrans fn="assign-name" from="/nsfc" nostat="/nsfc"
name="nsfc"
```

document-root

Applicable in NameTrans-class directives.

The document-root function specifies the root document directory for the server. If the physical path has not been set by a previous NameTrans function, the http://server-name/ part of the path is replaced by the physical path name for the document root.

When the server receives a request for http://server-name/somepath/somefile, the document-root function replaces http://server-name/ with the value of its root parameter. For example, if the document root directory is /usr/sun/webserver61/server1/docs, then when the server receives a request for http://server-name/a/b/file.html, the document-root function translates the path name for the requested resource to /usr/sun/webserver61/server1/docs/a/b/file.html.

This function always returns REQ_PROCEED. NameTrans directives listed after this will never be called, so be sure that the directive that invokes document-root is the last NameTrans directive.

There can be only one root document directory. To specify additional document directories, use the pfx2dir function to set up additional path name translations.

Parameters
The following table describes parameters for the document-root function.
The **home-page** function specifies the home page for your server. Whenever a client requests the server’s home page (/), they’ll get the document specified.

**Parameters**
The following table describes parameters for the **home-page** function.

**Table 4-11** home-page parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Path and name of the home page file. If path starts with a slash (/), it is assumed to be a full path to a file. This function sets the server’s path variable and returns REQ_PROCEED. If path is a relative path, it is appended to the URI and the function returns REQ_NOACTION continuing on to the other NameTrans directives.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>
Examples

```
NameTrans fn="home-page" path="/path/to/file.html"
NameTrans fn="home-page" path="/path/to/$id/file.html"
```

**match-browser**

Applicable in all stage directives. The **match-browser** SAF matches specific strings in the User-Agent string supplied by the browser, and then modifies the behavior of Sun ONE Web Server based upon the results by setting values for specified variables. See **match-browser**.

**ntrans-dav**

Applicable in **NameTrans-class** directives.

The **ntrans-dav** function determines whether a request should be handled by the WebDAV subsystem and if so, adds a **dav** object to the pipeline.

**Parameters**

The following table describes parameters for the **ntrans-dav** function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies an additional named object in obj.conf whose directives will be applied to this request.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

**Example**

```
NameTrans fn="ntrans-dav" name="dav"
```

**See Also**

**service-dav**
ntrans-j2ee

Applicable in NameTrans-class directives.

The ntrans-j2ee function determines whether a request maps to a Java web application context.

Parameters
The following table describes parameters for the ntrans-j2ee function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Named object in obj.conf whose directives are applied to requests made to Java web applications.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Example

NameTrans fn="ntrans-j2ee" name="j2ee"

See Also

service-j2ee, error-j2ee

pfx2dir

Applicable in NameTrans-class directives.

The pfx2dir function replaces a directory prefix in the requested URL with a real directory name. It also optionally allows you to specify the name of an object that matches the current request. (See the discussion of assign-name for details of using named objects.)

Parameters
The following table describes parameters for the pfx2dir function.
Examples
In the first example, the URL http://server-name/cgi-bin/resource (such as http://x.y.z/cgi-bin/test.cgi) is translated to the physical path name /httpd/cgi-local/resource (such as /httpd/cgi-local/test.cgi), and the server also starts processing the directives in the object named cgi.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>from</td>
<td>URI prefix to convert. It should not have a trailing slash (/).</td>
</tr>
<tr>
<td>dir</td>
<td>Local file system directory path that the prefix is converted to. It should not have a trailing slash (/).</td>
</tr>
<tr>
<td>name</td>
<td>(Optional) Specifies an additional named object in obj.conf whose directives will be applied to this request.</td>
</tr>
<tr>
<td>find-pathinfo-forward</td>
<td>(Optional) Makes the server look for the PATHINFO forward in the path right after the ntrans-base instead of backward from the end of path as the server function find-pathinfo does by default. The value you assign to this parameter is ignored. If you do not wish to use this parameter, leave it out. The find-pathinfo-forward parameter is ignored if the ntrans-base parameter is not set in rq-&gt;vars when the server function find-pathinfo is called. By default, ntrans-base is set. This feature can improve performance for certain URLs by reducing the number of stats performed in the server function find-pathinfo. On Windows, this feature can also be used to prevent the PATHINFO from the server URL normalization process (changing \ to <code>/</code>) when the PathCheck server function find-pathinfo is used. Some double-byte characters have hexadecimal values that may be parsed as URL separator characters such as \ or ~. Using the find-pathinfo-forward parameter can sometimes prevent incorrect parsing of URLs containing double-byte characters.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>
In the second example, the URL http://server-name/icons/resource (such as http://x.y.z/icons/happy/smiley.gif) is translated to the physical path name /users/nikki/images/resource (such as /users/nikki/images/smiley.gif).

The third example shows the use of the find-pathinfo-forward parameter. The URL http://server-name/cgi-bin/resource is translated to the physical path name /export/home/cgi-bin/resource.

### redirect

Applicable in NameTrans-class directives.

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, the server treats the request as a request for the new URL.

#### Parameters

The following table describes parameters for the redirect function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>from</td>
<td>Specifies the prefix of the requested URI to match.</td>
</tr>
<tr>
<td>url</td>
<td>(Maybe optional) Specifies a complete URL to return to the client. If you use this parameter, don’t use url-prefix (and vice versa).</td>
</tr>
</tbody>
</table>
Examples
In the first example, any request for http://server-name/whatever is translated to a request for http://tmpserver/whatever.

```
NameTrans fn=redirect from=/ url-prefix=http://tmpserver
```

In the second example, any request for http://server-name/toopopular/whatever is translated to a request for http://bigger/better/stronger/morepopular/whatever.

```
NameTrans fn=redirect from=/toopopular
url=http://bigger/better/stronger/morepopular
```

set-variable
Applicable in all stage directives. The set-variable SAF enables you to change server settings based upon conditional information in a request, and to manipulate variables in parameter blocks by using specific commands. See set-variable.
strip-params

Applicable in NameTrans-class directives.

The strip-params function removes embedded semicolon-delimited parameters from the path. For example, a URI of /dir1;param1/dir2 would become a path of /dir1/dir2. When used, the strip-params function should be the first NameTrans directive listed.

Parameters
The following table describes parameters for the strip-params function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Example
NameTrans fn=strip-params

unix-home

Applicable in NameTrans-class directives.

UNIX Only. The unix-home function translates user names (typically of the form ~username) into the user’s home directory on the server’s UNIX machine. You specify a URL prefix that signals 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 (elements in the passwd file that are not needed are indicated with *):

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

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

Parameters
The following table describes parameters for the unix-home function.
PathCheck

PathCheck directives check the local file system path that is returned after the NameTrans step. The path is checked for things such as CGI path information and for dangerous elements such as `./` and `../`, and then any access restriction is applied.

If there is more than one PathCheck directive, each of the functions is executed in order.

The following PathCheck-class functions are described in detail in this section:

- **check-acl** checks an access control list for authorization.
- **deny-existence** indicates that a resource was not found.
- **find-index** locates a default file when a directory is requested.
• **find-links** denies access to directories with certain file system links.

• **find-pathinfo** locates extra path info beyond the file name for the PATH_INFO CGI environment variable.

• **get-client-cert** gets the authenticated client certificate from the SSL3 session.

• **load-config** finds and loads extra configuration information from a file in the requested path.

• **match-browser** matches specific strings in the User-Agent string supplied by the browser, and then modifies the behavior of Sun ONE Web Server based upon the results by setting values for specified variables.

• **nt-uri-clean** denies access to requests with unsafe path names by indicating not found.

• **ntcgcicheck** looks for a CGI file with a specified extension.

• **pcheck-dav** inserts a DAV-specific service function.

• **require-auth** denies access to unauthorized users or groups.

• **set-variable** enables you to change server settings based upon conditional information in a request, and to manipulate variables in parameter blocks by using specific commands.

• **set-virtual-index** specifies a virtual index for a directory.

• **ssl-check** checks the secret keysize.

• **ssl-logout** invalidates the current SSL session in the server’s SSL session cache.

• **unix-uri-clean** denies access to requests with unsafe path names by indicating not found.

### check-acl

Applicable in PathCheck-class directives.

The `check-acl` function specifies an access control list (ACL) to use to check whether the client is allowed to access the requested resource. An access control list contains information about who is or is not allowed to access a resource, and under what conditions access is allowed.
Regardless of the order of PathCheck directives in the object, check-acl functions are executed first. They cause user authentication to be performed, if required by the specified ACL, and will also update the access control state.

**Parameters**

The following table describes parameters for the check-acl function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>acl</td>
<td>Name of an access control list.</td>
</tr>
<tr>
<td>path</td>
<td>(Optional) Wildcard pattern that specifies the path for which to apply the ACL.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

**Example**

```bash
PathCheck fn=check-acl acl="*HRonly*"
```

**find-compressed**

Applicable in PathCheck-class directives.

The find-compressed function checks if a compressed version of the requested file is available. If the following conditions are met, find-compressed changes the path to point to the compressed file:

- A compressed version is available.
- The compressed version is at least as recent as the noncompressed version.
- The client supports compression.

Not all clients support compression. The find-compressed function allows you to use a single URL for both the compressed and noncompressed versions of a file. The version of the file that is selected is based on the individual clients' capabilities.

A compressed version of a file must have the same file name as the noncompressed version but with a .gz suffix. For example, the compressed version of a file named /httpd/docs/index.html would be named /httpd/docs/index.html.gz. To compress files, you can use the freely available gzip program.
Because compressed files are sent as is to the client, you should not compress files such as SHTML pages, CGI programs, or pages created with JavaServer Pages™ (JSP™) technology that need to be interpreted by the server. To compress the dynamic content generated by these types of files, use the http-compression filter.

The find-compressed function does nothing if the HTTP method is not GET or HEAD.

**Parameters**
The following table describes parameters for the find-compressed function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| check-age | Specifies whether to check if the compressed version is older than the noncompressed version. Possible values are yes and no.  
- If set to yes, the compressed version will not be selected if it is older than the noncompressed version.  
- If set to no, the compressed version will always be selected, even if it is older than the noncompressed version.  
By default, the value is set to yes. |
| vary      | Specifies whether to insert a Vary: Accept-Encoding header. Possible values are yes or no.  
- If set to yes, a Vary: Accept-Encoding header is always inserted when a compressed version of a file is selected.  
- If set to no, a Vary: Accept-Encoding header is never inserted.  
By default, the value is set to yes. |
| bucket    | (Optional) Common to all obj.conf functions. |
**Example**

```xml
<Object name="default">
NameTrans fn="assign-name" from="*.html" name="find-compressed"
...
</Object>
<Object name="find-compressed">
PathCheck fn="find-compressed"
</Object>
```

**See Also**

http-compression

**deny-existence**

Applicable in PathCheck-class directives.

The `deny-existence` function sends a “not found” message when a client tries to access a specified path. The server sends “not found” instead of “forbidden,” so the user cannot tell if the path exists.

**Parameters**

The following table describes parameters for the `deny-existence` function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>(Optional) Wildcard pattern of the file system path to hide. If the path does not match, the function does nothing and returns <code>REQ_NOACTION</code>. If the path is not provided, it is assumed to match.</td>
</tr>
<tr>
<td>bong-file</td>
<td>(Optional) Specifies a file to send rather than responding with the “not found” message. It is a full file system path.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all <code>obj.conf</code> functions.</td>
</tr>
</tbody>
</table>
**Examples**

```
PathCheck fn=deny-existence path=/usr/sun/server61/docs/private
PathCheck fn=deny-existence bong-file=/svr/msg/go-away.html
```

**find-index**

Applicable in PathCheck-class directives.

The `find-index` function investigates whether 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.

Note that if the file `obj.conf` has a NameTrans directive that calls `home-page`, and the requested directory is the root directory, then the home page rather than the index page is returned to the client.

The `find-index` function does nothing if there is a query string, if the HTTP method is not `GET`, or if the path is that of a valid file.

**Parameters**

The following table describes parameters for the `find-index` function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index-names</td>
<td>Comma-separated list of index file names to look for. Use spaces only if they are part of a file name. Do not include spaces before or after the commas. This list is case-sensitive if the file system is case-sensitive.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all <code>obj.conf</code> functions.</td>
</tr>
</tbody>
</table>

**Example**

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

Applicable in PathCheck-class directives.

UNIX Only. The find-links function searches the current path for symbolic or hard links to other directories or file systems. If any are found, an error is returned. This function is normally used for directories that are not trusted (such as user home directories). It prevents someone from pointing to information that should not be made public.

Parameters
The following table describes parameters for the find-links function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disable</td>
<td>Character string of links to disable:</td>
</tr>
<tr>
<td></td>
<td>•  h is hard links</td>
</tr>
<tr>
<td></td>
<td>•  s is soft links</td>
</tr>
<tr>
<td></td>
<td>•  o allows symbolic links from user home directories only if the user owns the target of the link</td>
</tr>
<tr>
<td>dir</td>
<td>Directory to begin checking. If you specify an absolute path, any request to that path and its subdirectories is checked for symbolic links. If you specify a partial path, any request containing that partial path is checked for symbolic links. For example, if you use /user/ and a request comes in for some/user/directory, then that directory is checked for symbolic links.</td>
</tr>
<tr>
<td>checkFileExistence</td>
<td>Checks linked file for existence and aborts request with 403 (forbidden) if this check fails.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Examples

PathCheck fn=find-links disable=sh dir=/foreign-dir

PathCheck fn=find-links disable=so dir=public_html
See Also
unix-home

find-pathinfo

Applicable in PathCheck-class directives.

The find-pathinfo function finds any extra path information after the file name in the URL and stores it for use in the CGI environment variable PATH_INFO.

Parameters

The following table describes parameters for the find-pathinfo function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Examples

PathCheck fn=find-pathinfo

PathCheck fn=find-pathinfo find-pathinfo-forward=""

get-client-cert

Applicable in PathCheck-class directives.

The get-client-cert function gets the authenticated client certificate from the SSL3 session. It can apply to all HTTP methods, or only to those that match a specified pattern. It only works when SSL is enabled on the server.

If the certificate is present or obtained from the SSL3 session, the function returns REQ_NOACTION, allowing the request to proceed; otherwise, it returns REQ_ABORTED and sets the protocol status to 403 FORBIDDEN, causing the request to fail and the client to be given the FORBIDDEN status.
Parameters

The following table describes parameters for the `get-client-cert` function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dorequest</td>
<td>Controls whether to actually try to get the certificate, or just test for its presence. If dorequest is absent, the default value is 0.</td>
</tr>
<tr>
<td></td>
<td>• 1 tells the function to redo the SSL3 handshake to get a client certificate, if the server does not already have the client certificate. This typically causes the client to present a dialog box to the user to select a client certificate. The server may already have the client certificate if it was requested on the initial handshake, or if a cached SSL session has been resumed.</td>
</tr>
<tr>
<td></td>
<td>• 0 tells the function not to redo the SSL3 handshake if the server does not already have the client certificate.</td>
</tr>
<tr>
<td></td>
<td>If a certificate is obtained from the client and verified successfully by the server, the ASCII base64 encoding of the DER-encoded X.509 certificate is placed in the parameter <code>auth-cert</code> in the <code>Request-&gt;vars</code> pblock, and the function returns <code>REQ_PROCEED</code>, allowing the request to proceed.</td>
</tr>
<tr>
<td>require</td>
<td>Controls whether failure to get a client certificate will abort the HTTP request. If require is absent, the default value is 1.</td>
</tr>
<tr>
<td></td>
<td>• 1 tells the function to abort the HTTP request if the client certificate is not present after dorequest is handled. In this case, the HTTP status is set to <code>PROTOCOL_FORBIDDEN</code>, and the function returns <code>REQ_ABORTED</code>.</td>
</tr>
<tr>
<td></td>
<td>• 0 tells the function to return <code>REQ_NOACTION</code> if the client certificate is not present after dorequest is handled.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Specifies a wildcard pattern for the HTTP methods for which the function will be applied. If method is absent, the function is applied to all requests.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all <code>obj.conf</code> functions.</td>
</tr>
</tbody>
</table>
Example

```bash
# Get the client certificate from the session.
# If a certificate is not already associated with the
# session, request one.
# The request fails if the client does not present a
# valid certificate.
PathCheck fn="get-client-cert" dorequest="1"
```

**load-config**

Applicable in PathCheck-class directives.

The `load-config` function searches for configuration files in document directories and adds the file's contents to the server's existing configuration. These configuration files (also known as dynamic configuration files) specify additional access control information for the requested resource. Depending on the rules in the dynamic configuration files, the server may or may not allow the client to access the requested resource.

Each directive that invokes `load-config` is associated with a base directory, which is either stated explicitly through the `basedir` parameter or derived from the root directory for the requested resource. The base directory determines two things:

- The topmost directory for which requests will invoke this call to the `load-config` function.

  For example, if the base directory is `D:/sun/server61/docs/nikki/`, then only requests for resources in this directory or its subdirectories (and their subdirectories) trigger the search for dynamic configuration files. A request for the resource `D:/sun/server61/docs/somefile.html` does not trigger the search in this case, since the requested resource is in a parent directory of the base directory.

- The topmost directory in which the server looks for dynamic configuration files to apply to the requested resource.

  If the base directory is `D:/sun/server61/docs/nikki/`, the server starts its search for dynamic configuration files in this directory. It may or may not also search subdirectories (but never parent directories), depending on other factors.
When you enable dynamic configuration files through the Server Manager interface, the system writes additional objects with ppath parameters into the obj.conf file. If you manually add directives that invoke load-config to the default object (rather than putting them in separate objects), the Server Manager interface might not reflect your changes.

If you manually add PathCheck directives that invoke load-config to the file obj.conf, put them in additional objects (created with the <OBJECT> tag) rather than putting them in the default object. Use the ppath attribute of the OBJECT tag to specify the partial path name for the resources to be affected by the access rules in the dynamic configuration file. The partial path name can be any path name that matches a pattern, which can include wildcard characters.

For example, the following <OBJECT> tag specifies that requests for resources in the directory D:/sun/server61/docs are subject to the access rules in the file my.nsconfig.

```xml
<Object ppath="D:/sun/server61/docs/*">
  PathCheck fn="load-config" file="my.nsconfig" descend=1 basedir="D:/sun/server61/docs"
</Object>
```

**NOTE** If the ppath resolves to a resource or directory that is higher in the directory tree (or is in a different branch of the tree) than the base directory, the load-config function is not invoked. This is because the base directory specifies the highest-level directory for which requests will invoke the load-config function.

The load-config function returns REQ_PROCEED if configuration files were loaded, REQ_ABORTED on error, or REQ_NOACTION when no files are loaded.

**Parameters**
The following table describes parameters for the load-config function.
Table 4-25  load-config parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>(Optional) Name of the dynamic configuration file containing the access rules to be applied to the requested resource. If not provided, the file name is assumed to be .nsconfig.</td>
</tr>
<tr>
<td>disable-types</td>
<td>(Optional) Specifies a wildcard pattern of types to disable for the base directory, such as magnus-internal/cgi. Requests for resources matching these types are aborted.</td>
</tr>
<tr>
<td>descend</td>
<td>(Optional) If present, specifies that the server should search in subdirectories of this directory for dynamic configuration files. For example, descend=1 specifies that the server should search subdirectories. No descend parameter specifies that the function should search only the base directory.</td>
</tr>
<tr>
<td>basedir</td>
<td>(Optional) Specifies base directory. This is the highest-level directory for which requests will invoke the load-config function, and is also the directory where the server starts searching for configuration files. If basedir is not specified, the base directory is assumed to be the root directory that results from translating the requested resource’s URL to a physical path name. For example, if the request is for <a href="http://server-name/a/b/file.html">http://server-name/a/b/file.html</a>, the physical file name would be /document-root/a/b/file.html.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Examples

In this example, whenever the server receives a request for any resource containing the substring secret that resides in D:/Sun/WebServer61/server1/docs/nikki/ or a subdirectory thereof, it searches for a configuration file called checkaccess.nsconfig.

The server starts the search in the directory D:/Sun/WebServer61/server1/docs/nikki, and searches subdirectories too. It loads each instance of checkaccess.nsconfig that it finds, applying the access control rules contained therein to determine whether the client is allowed to access the requested resource.
**match-browser**

Applicable in all stage directives. The `match-browser` SAF matches specific strings in the User-Agent string supplied by the browser, and then modifies the behavior of Sun ONE Web Server based upon the results by setting values for specified variables. See `match-browser`.

**nt-uri-clean**

Applicable in `PathCheck-class` directives.

**Windows Only.** The `nt-uri-clean` function denies access to any resource whose physical path contains `\..\` or `\` (these are potential security problems).

**Parameters**

The following table describes parameters for the `nt-uri-clean` function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tildeok</td>
<td>If present, allows tilde (~) characters in URIs. This is a potential security risk on the Windows platform, where <code>longfi-1.htm</code> might reference <code>longfilename.htm</code> but does not go through the proper ACL checking. If present, <code>&quot;/&quot;</code> sequences are allowed.</td>
</tr>
<tr>
<td>dotdirok</td>
<td>If present, <code>&quot;/&quot;</code> sequences are allowed.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

**Example**

```
PathCheck fn=nt-uri-clean
```
ntcgicheck

Applicable in PathCheck-class directives.

**Windows Only.** The ntcgicheck function specifies the file name extension to be added to any file name that does not have an extension, or to be substituted for any file name that has the extension `.cgi`.

**Parameters**
The following table describes parameters for the `ntcgicheck` function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>extension</td>
<td>The replacement file extension.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all <code>obj.conf</code> functions.</td>
</tr>
</tbody>
</table>

**Example**

```
PathCheck fn=ntcgicheck extension=pl
```

**See Also**

*unix-uri-clean*

*pcheck-dav*

Applicable in PathCheck-class directives.

The `pcheck-dav` function inserts a DAV-specific service function as the first service function if the `Translate:f` header is present, DAV is enabled for the request uri, and a corresponding source uri for the request uri exists. During the Service stage, this inserted service function restarts the request if necessary; otherwise, `REQ_NOACTION` is returned.
Parameters
The following table describes parameters for the `pcheck-dav` function.

Table 4-28  pcheck-dav parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucket</td>
<td>(Optional) Common to all <code>obj.conf</code> functions.</td>
</tr>
</tbody>
</table>

See Also
`ntrans-dav`, `service-dav`

require-auth

Applicable in `PathCheck`-class directives.

The `require-auth` function allows access to resources only if the user or group is authorized. Before this function is called, an authorization function (such as `basic-auth`) must be called in an `AuthTrans` directive.

If a user was authorized in an `AuthTrans` directive, and the `auth-user` parameter is provided, then the user's name must match the `auth-user` wildcard value. Also, if the `auth-group` parameter is provided, the authorized user must belong to an authorized group, which must match the `auth-user` wildcard value.

Parameters
The following table describes parameters for the `require-auth` function.

Table 4-29  require-auth parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>(Optional) Wildcard local file system path on which this function should operate. If no path is provided, the function applies to all paths.</td>
</tr>
<tr>
<td>auth-type</td>
<td>Type of HTTP authorization used, and must match the <code>auth-type</code> from the previous authorization function in <code>AuthTrans</code>. Currently, <code>basic</code> is the only authorization type defined.</td>
</tr>
<tr>
<td>realm</td>
<td>String sent to the browser indicating the secure area (or realm) for which a user name and password are requested.</td>
</tr>
</tbody>
</table>
Example

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

See Also

basic-auth, basic-ncsa

set-variable

Applicable in all stage directives. The set-variable SAF enables you to change server settings based upon conditional information in a request, and to manipulate variables in parameter blocks by using specific commands. See set-variable.

set-virtual-index

Applicable in PathCheck-class directives.

The set-virtual-index function specifies a virtual index for a directory, which determines the URL forwarding. The index can refer to a LiveWire application, a servlet in its own namespace, a Sun™ ONE Application Server applogic, and so on.

REQ_NOACTION is returned if none of the URIs listed in the from parameter match the current URI. REQ_ABORTED is returned if the file specified by the virtual-index parameter is missing, or if the current URI cannot be found. REQ_RESTART is returned if the current URI matches any one of the URIs mentioned in the from parameter, or if there is no from parameter.
Parameters
The following table describes parameters for the `set-virtual-index` function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual-index</td>
<td>URI of the content generator that acts as an index for the URI the user enters.</td>
</tr>
<tr>
<td>from</td>
<td>(Optional) Comma-separated list of URIs for which this virtual-index is applicable. If from is not specified, the virtual-index always applies.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Example
```
# MyLWApp is a LiveWire application
PathCheck fn=set-virtual-index virtual-index=MyLWApp
```

ssl-check
Applicable in PathCheck-class directives.

If a restriction is selected that is not consistent with the current cipher settings under Security Preferences, this function opens a popup dialog warning that ciphers with larger secret keysizes need to be enabled. This function is designed to be used together with a Client tag to limit access of certain directories to nonexportable browsers.

The function returns `REQ_NOACTION` if SSL is not enabled, or if the secret-keysize parameter is not specified. If the secret keysize for the current session is less than the specified secret-keysize and the bong-file parameter is not specified, the function returns `REQ_ABORTED` with a status of `PROTOCOL_FORBIDDEN`. If the bong file is specified, the function returns `REQ_PROCEED`, and the path variable is set to the bong file name. Also, when a keysize restriction is not met, the SSL session cache entry for the current session is invalidated, so that a full SSL handshake will occur the next time the same client connects to the server.

Requests that use ssl-check are not cacheable in the accelerator file cache if ssl-check returns something other than `REQ_NOACTION`. 
Parameters
The following table describes parameters for the `ssl-check` function.

**Table 4-31**  ssl-check parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>secret-keysize</td>
<td>(Optional) Minimum number of bits required in the secret key.</td>
</tr>
<tr>
<td>bong-file</td>
<td>(Optional) Name of a file (not a URI) to be served if the restriction is not met.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all <code>obj.conf</code> functions.</td>
</tr>
</tbody>
</table>

**ssl-logout**

Applicable in `PathCheck-class` directives.

The `ssl-logout` function invalidates the current SSL session in the server’s SSL session cache. This does not affect the current request, but the next time the client connects, a new SSL session will be created. If SSL is enabled, this function returns `REQ_PROCEED` after invalidating the session cache entry. If SSL is not enabled, it returns `REQ_NOACTION`.

**Parameters**
The following table describes parameters for the `ssl-logout` function.

**Table 4-32**  ssl-logout parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucket</td>
<td>(Optional) Common to all <code>obj.conf</code> functions.</td>
</tr>
</tbody>
</table>

**unix-uri-clean**

Applicable in `PathCheck-class` directives.

**UNIX Only.** The `unix-uri-clean` function denies access to any resource whose physical path contains `/./`, `/../` or `//` (these are potential security problems).

**Parameters**
The following table describes parameters for the `unix-uri-clean` function.
ObjectType

ObjectType directives determine the MIME type of the file to send to the client in response to a request. MIME attributes currently sent are type, encoding, and language. The MIME type is sent to the client as the value of the Content-Type header.

ObjectType directives also set the type parameter, which is used by Service directives to determine how to process the request according to what kind of content is being requested.

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 later directives try to set that attribute to something else, the first setting is used and the subsequent ones are ignored.

The obj.conf file almost always has an ObjectType directive that calls the type-by-extension function. This function instructs the server to look in a particular file (the MIME types file) to deduce the content type from the extension of the requested resource.

The following ObjectType-class functions are described in detail in this section:

- **force-type** sets the Content-Type header for the response to a specific type.

---

**Table 4-33**  
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dotdirok</td>
<td>If present, “//” sequences are allowed.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

**Example**

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

**See Also**

nt-uri-clean
match-browser matches specific strings in the User-Agent string supplied by the browser, and then modifies the behavior of Sun ONE Web Server based upon the results by setting values for specified variables.

set-default-type allows you to define a default charset, content-encoding, and content-language for the response being sent back to the client.

set-variable enables you to change server settings based upon conditional information in a request, and to manipulate variables in parameter blocks by using specific commands.

shtml-hacktype requests that .htm and .html files are parsed for server-parsed HTML commands.

type-by-exp sets the Content-Type header for the response based on the requested path.

type-by-extension sets the Content-Type header for the response based on the file’s extension and the MIME types database.

force-type

Applicable in ObjectType-class directives.

The force-type function assigns a type to requests that do not already have a MIME type. This is used to specify a default object type.

Make sure that the directive that calls this function comes last in the list of ObjectType directives, so that all other ObjectType directives have a chance to set the MIME type first. 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 later directives try to set that attribute to something else, the first setting is used and the subsequent ones are ignored.

Parameters

The following table describes parameters for the force-type function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>(Optional) Type assigned to a matching request (the Content-Type header).</td>
</tr>
</tbody>
</table>
### Table 4-34  force-type parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enc</td>
<td>(Optional) Encoding assigned to a matching request (the Content-Encoding header).</td>
</tr>
<tr>
<td>lang</td>
<td>(Optional) Language assigned to a matching request (the Content-Language header).</td>
</tr>
<tr>
<td>charset</td>
<td>(Optional) Character set for the magnus-charset parameter in rq-&gt;srvhdrs. If the browser sent the Accept-Charset header or its User-Agent is Mozilla™/1.1 or newer, then append “; charset=charset” to content-type, where charset is the value of the magnus-charset parameter in rq-&gt;srvhdrs.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

### Example

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

### See Also

type-by-extension, type-by-exp

**match-browser**

Applicable in all stage directives. The match-browser SAF matches specific strings in the User-Agent string supplied by the browser, and then modifies the behavior of Sun ONE Web Server based upon the results by setting values for specified variables. See match-browser.

**set-default-type**

Applicable in ObjectType-class directives.
The `set-default-type` function allows you to define a default charset, content-encoding, and content-language for the response being sent back to the client.

If the charset, content-encoding, and content-language have not been set for a response, then just before the headers are sent the defaults defined by `set-default-type` are used. Note that by placing this function in different objects in `obj.conf`, you can define different defaults for different parts of the document tree.

**Parameters**
The following table describes parameters for the `set-default-type` function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enc</td>
<td>(Optional) Encoding assigned to a matching request (the Content-Encoding header).</td>
</tr>
<tr>
<td>lang</td>
<td>(Optional) Language assigned to a matching request (the Content-Language header).</td>
</tr>
<tr>
<td>charset</td>
<td>(Optional) Character set for the magnus-charset parameter in <code>rq-&gt;srvhdrs</code>. If the browser sent the Accept-Charset header or its User-agent is Mozilla/1.1 or newer, then append &quot;; charset=charset&quot; to content-type, where charset is the value of the magnus-charset parameter in <code>rq-&gt;srvhdrs</code>.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all <code>obj.conf</code> functions.</td>
</tr>
</tbody>
</table>

**Example**

```
ObjectType fn="set-default-type" charset="iso_8859-1"
```

**set-variable**

Applicable in all stage directives. The `set-variable` SAF enables you to change server settings based upon conditional information in a request, and to manipulate variables in parameter blocks by using specific commands. See `set-variable`.
shtml-hacktype

Applicable in ObjectType-class directives.

The shtml-hacktype function changes the Content-Type of any .htm or .html file to magnus-internal/parsed-html and returns REQ_PROCEED. This provides backward compatibility with server-side includes for files with .htm or .html extensions. The function may also check the execute bit for the file on UNIX systems. The use of this function is not recommended.

Parameters
The following table describes parameters for the shtml-hacktype function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>exec-hack</td>
<td>(UNIX only, optional) Tells the function to change the content-type only if the execute bit is enabled. The value of the parameter is not important; it need only be provided. You may use exec-hack=true.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Example

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

type-by-exp

Applicable in ObjectType-class directives.

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
The following table describes parameters for the type-by-exp function.
Table 4-37  type-by-exp parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>exp</td>
<td>Wildcard pattern of paths for which this function is applied.</td>
</tr>
<tr>
<td>type</td>
<td>(Optional) Type assigned to a matching request (the Content-Type header).</td>
</tr>
<tr>
<td>enc</td>
<td>(Optional) Encoding assigned to a matching request (the Content-Encoding header).</td>
</tr>
<tr>
<td>lang</td>
<td>(Optional) Language assigned to a matching request (the Content-Language header).</td>
</tr>
<tr>
<td>charset</td>
<td>(Optional) is the character set for the magnus-charset parameter in rq-&gt;srvhdrs. If the browser sent the Accept-Charset header or its User-Agent is Mozilla/1.1 or newer, then append “; charset=charset” to content-type, where charset is the value of the magnus-charset parameter in rq-&gt;srvhdrs.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Example

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

See Also

type-by-extension, force-type

type-by-extension

Applicable in ObjectType-class directives.

The type-by-extension function instructs the server to look in a table of MIME type mappings to find the MIME type of the requested resource according to the extension of the requested resource. The MIME type is added to the Content-Type header sent back to the client.
The table of MIME type mappings is created by a MIME element in the server.xml file, which loads a MIME types file or list and creates the mappings. For more information about server.xml and MIME types files, see the Sun ONE Web Server 6.1 Administrator’s Configuration File Reference Guide.

For example, the following two lines are part of a MIME types file:

```
type=text/html   exts=htm,html
type=text/plain   exts=txt
```

If the extension of the requested resource is .htm or .html, the type-by-extension file sets the type to text/html. If the extension is .txt, the function sets the type to text/plain.

**Parameters**
The following table describes parameters for the type-by-extension function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

**Example**

```
Objectype fn=type-by-extension
```

**See Also**
type-by-exp, force-type

**Input**

All `Input` directives are executed when the server or a plugin first attempts to read entity body data from the client.
The **Input** stage allows you to select filters that will process incoming request data read by the **Service** step.

NSAPI filters in Sun ONE Web Server 6.1 enable a function to intercept (and potentially modify) the content presented to or generated by another function.

You can add NSAPI filters that process incoming data by invoking the **insert-filter** SAF in the **Input** stage of the request-handling process. The **Input** directives are executed at most once per request.

You can also define the appropriate position of a specific filter within the filter stack. For example, filters that translate content from XML to HTML are placed higher in the filter stack than filters that compress data for transmission. You can use the **filter_create** function to define the filter's position in the filter stack, and the **init-filter-order** to override the defined position.

When two or more filters are defined to occupy the same position in the filter stack, filters that were inserted later will appear higher than filters that were inserted earlier. That is, the order of **Input** fn="insert-filter" and **Output** fn="insert-filter" directives in **obj.conf** becomes important.

The following **Input**-class functions are described in detail in this section:

- **insert-filter** adds a filter to the filter stack to process incoming data.
- **match-browser** matches specific strings in the **User-Agent** string supplied by the browser, and then modifies the behavior of Sun ONE Web Server based upon the results by setting values for specified variables.
- **remove-filter** removes a filter from the filter stack.
- **set-variable** enables you to change server settings based upon conditional information in a request, and to manipulate variables in parameter blocks by using specific commands.

**insert-filter**

Applicable in **Input**-class directives.

The **insert-filter** SAF is used to add a filter to the filter stack to process incoming (client-to-server) data.

The order of **Input** fn="insert-filter" and **Output** fn="insert-filter" directives can be important.
Returns

Returns REQ_PROCEED if the specified filter was inserted successfully or
REQ_NOACTION if the specified filter was not inserted because it was not required.
Any other return value indicates an error.

Parameters

The following table describes parameters for the insert-filter function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>Specifies the name of the filter to insert.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Example

```
Input fn="insert-filter" filter="http-decompression"
```

match-browser

Applicable in all stage directives. The match-browser SAF matches specific strings in
the User-Agent string supplied by the browser, and then modifies the behavior of
Sun ONE Web Server based upon the results by setting values for specified variables. See
match-browser.

remove-filter

Applicable in Input-, Output-, Service-, and Error-class directives.

The remove-filter SAF is used to remove a filter from the filter stack. If the filter
has been inserted multiple times, only the topmost instance is removed. In general,
it is not necessary to remove filters with remove-filter, as they will be removed
automatically at the end of the request.

Returns

Returns REQ_PROCEED if the specified filter was removed successfully, or
REQ_NOACTION if the specified filter was not part of the filter stack. Any other
return value indicates an error.
Parameters
The following table describes parameters for the `remove-filter` function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>Specifies the name of the filter to remove.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all <code>obj.conf</code> functions.</td>
</tr>
</tbody>
</table>

Example

```
Input fn="remove-filter" filter="http-compression"
```

**set-variable**

Applicable in all stage directives. The `set-variable` SAF enables you to change server settings based upon conditional information in a request, and to manipulate variables in parameter blocks by using specific commands. See `set-variable`.

**Output**

All `Output` directives are executed when the server or a plugin first attempts to write entity body data from the client.

The `Output` stage allows you to select filters that will process outgoing data.

You can add NSAPI filters that process outgoing data by invoking the `insert-filter` SAF in the `Output` stage of the request-handling process. The `Output` directives are executed at most once per request.

You can define the appropriate position of a specific filter within the filter stack. For example, filters that translate content from XML to HTML are placed higher in the filter stack than filters that compress data for transmission. You can use the `filter_create` function to define the filter’s position in the filter stack, and the `init-filter-order` to override the defined position.
When two or more filters are defined to occupy the same position in the filter stack, filters that were inserted later will appear higher than filters that were inserted earlier.

The following Output-class functions are described in detail in this section:

- **insert-filter** adds a filter to the filter stack to process outgoing data.
- **match-browser** matches specific strings in the User-Agent string supplied by the browser, and then modifies the behavior of Sun ONE Web Server based upon the results by setting values for specified variables.
- **remove-filter** removes a filter from the filter stack.
- **set-variable** enables you to change server settings based upon conditional information in a request, and to manipulate variables in parameter blocks by using specific commands.

### insert-filter

Applicable in Output-class directives.

The insert-filter SAF is used to add a filter to the filter stack to process outgoing (server-to-client) data.

The order of Input `fn="insert-filter"` and Output `fn="insert-filter"` directives can be important.

**Returns**

Returns `REQ_PROCEED` if the specified filter was inserted successfully, or `REQ_NOACTION` if the specified filter was not inserted because it was not required. Any other return value indicates an error.

**Parameters**

The following table describes parameters for the insert-filter function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>Specifies the name of the filter to insert.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all <code>obj.conf</code> functions.</td>
</tr>
</tbody>
</table>
Example

```plaintext
Output fn="insert-filter" filter="http-compression"
```

**match-browser**

Applicable in all stage directives. The `match-browser` SAF matches specific strings in the `User-Agent` string supplied by the browser, and then modifies the behavior of Sun ONE Web Server based upon the results by setting values for specified variables. See `match-browser`.

**remove-filter**

Applicable in `Input`, `Output`, `Service`, and `Error`-class directives.

The `remove-filter` SAF is used to remove a filter from the filter stack. If the filter has been inserted multiple times, only the topmost instance is removed. In general, it is not necessary to remove filters with `remove-filter`, as they will be removed automatically at the end of the request.

**Returns**

Returns `REQ_PROCEED` if the specified filter was removed successfully, or `REQ_NOACTION` if the specified filter was not part of the filter stack. Any other return value indicates an error.

**Parameters**

The following table describes parameters for the `remove-filter` function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>Specifies the name of the filter to remove.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all <code>obj.conf</code> functions.</td>
</tr>
</tbody>
</table>

Example

```plaintext
Output fn="remove-filter" filter="http-compression"
```
set-variable

Applicable in all stage directives. The `set-variable` SAF enables you to change server settings based upon conditional information in a request, and to manipulate variables in parameter blocks by using specific commands. See `set-variable`.

Service

The `Service`-class of functions sends the response data to the client.

Every `Service` directive has the following optional parameters to determine whether the function is executed. All optional parameters must match the current request for the function to be executed.

- **type**
  
  (Optional) Specifies a wildcard pattern of MIME types for which this function will be executed. The `magnus-internal/*` MIME types are used only to select a `Service` function to execute.

- **method**
  
  (Optional) Specifies a wildcard pattern of HTTP methods for which this function will be executed. Common HTTP methods are `GET`, `HEAD`, and `POST`.

- **query**
  
  (Optional) Specifies a wildcard pattern of query strings for which this function will be executed.

- **UseOutputStreamSize**
  
  (Optional) Determines the default output stream buffer size, in bytes, for data sent to the client. If this parameter is not specified, the default is 8192 bytes.

  **NOTE**

  The `UseOutputStreamSize` parameter can be set to zero (0) in the `obj.conf` file to disable output stream buffering. For the `magnus.conf` file, setting `UseOutputStreamSize` to zero (0) has no effect.

- **flushTimer**
(Optional) Determines the maximum number of milliseconds between write operations in which buffering is enabled. If the interval between subsequent write operations is greater than the flushTimer value for an application, further buffering is disabled. This is necessary for status-monitoring CGI applications that run continuously and generate periodic status update reports. If this parameter is not specified, the default is 3000 milliseconds.

- **ChunkedRequestBufferSize**

  (Optional) Determines the default buffer size, in bytes, for “un-chunking” request data. If this parameter is not specified, the default is 8192 bytes.

- **ChunkedRequestTimeout**

  (Optional) Determines the default timeout, in seconds, for “un-chunking” request data. If this parameter is not specified, the default is 60 seconds.

If there is more than one Service-class function, the first one matching the optional wildcard parameters (type, method, and query) is executed.

For more information about the UseOutputStreamSize, flushTimer, ChunkedRequestBufferSize, and ChunkedRequestTimeout parameters, see “Buffered Streams” in the Sun ONE Web Server 6.1 NSAPI Programmer’s Guide. The UseOutputStreamSize, ChunkedRequestBufferSize, and ChunkedRequestTimeout parameters also have equivalent magnus.conf directives. For more information, see “Chunked Encoding” in the chapter “Syntax and Use of magnus.conf” in the Sun ONE Web Server 6.1 Administrator’s Configuration File Reference. The obj.conf parameters override the magnus.conf directives.

By default, the server sends the requested file to the client by calling the send-file function. The directive that sets the default is:

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

This directive usually comes last in the set of Service-class directives to give all other Service directives a chance to be invoked. This directive is invoked if the method of the request is GET, HEAD, or POST, and the type does not start with magnus-internal/. Note here that the pattern *~ means “does not match.” For a list of characters that can be used in patterns, see the Sun ONE Web Server 6.1 NSAPI Programmer’s Guide.

The following Service-class functions are described in detail in this section:

- **add-footer** appends a footer specified by a file name or URL to an HTML file.
• **add-header** prepends a header specified by a file name or URL to an HTML file.

• **append-trailer** appends text to the end of an HTML file.

• **imagemap** handles server-side image maps.

• **index-common** generates a fancy list of the files and directories in a requested directory.

• **index-simple** generates a simple list of files and directories in a requested directory.

• **key-too-small** indicates to the client that the provided certificate key size is too small to accept.

• **list-dir** lists the contents of a directory.

• **make-dir** creates a directory.

• **match-browser** matches specific strings in the *User-Agent* string supplied by the browser, and then modifies the behavior of Sun ONE Web Server based upon the results by setting values for specified variables.

• **query-handler** handles the HTML ISINDEX tag.

• **remove-dir** deletes an empty directory.

• **remove-file** deletes a file.

• **remove-filter** removes a refilter from the filter stack.

• **rename-file** renames a file.

• **send-cgi** sets up environment variables, launches a CGI program, and sends the response to the client.

• **send-error** sends an HTML file to the client in place of a specific HTTP response status.

• **send-file** sends a local file to the client.

• **send-range** sends a range of bytes of a file to the client.

• **send-shell-cgi** sets up environment variables, launches a shell CGI program, and sends the response to the client.

• **send-wincgi** sets up environment variables, launches a WinCGI program, and sends the response to the client.

• **service-dav** services static content and restarts the request with the *sourceuri* for dynamic content.
• **service-dump** creates a performance report based on collected performance bucket data.

• **service-j2ee** services requests made to Java web applications.

• **service-trace** services TRACE requests.

• **set-variable** enables you to change server settings based upon conditional information in a request, and to manipulate variables in parameter blocks by using specific commands.

• **shtml_send** parses an HTML file for server-parsed HTML commands.

• **stats-xml** creates a performance report in XML format.

• **upload-file** uploads and saves a file.

**add-footer**

Applicable in **Service**-class directives.

This function appends a footer to an HTML file that is sent to the client. The footer is specified either as a file name or a URI, thus the footer can be dynamically generated. To specify static text as a footer, use the **append-trailer** function.

**Parameters**

The following table describes parameters for the **add-footer** function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>(Optional) Path name to the file containing the footer. Specify either file or uri. By default, the path name is relative. If the path name is absolute, pass the <strong>NSIntAbsFilePath</strong> parameter as yes.</td>
</tr>
<tr>
<td>uri</td>
<td>(Optional) URI pointing to the resource containing the footer. Specify either file or uri.</td>
</tr>
<tr>
<td><strong>NSIntAbsFilePath</strong></td>
<td>(Optional) If the file parameter is specified, the <strong>NSIntAbsFilePath</strong> parameter determines whether the file name is absolute or relative. The default is relative. Set the value to yes to indicate an absolute file path.</td>
</tr>
<tr>
<td><strong>type</strong></td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td><strong>method</strong></td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
</tbody>
</table>
add-header

Applicable in Service-class directives.

This function prepends a header to an HTML file that is sent to the client. The header is specified either as a file name or a URI, thus the header can be dynamically generated.

Parameters
The following table describes parameters for the add-header function.

### Table 4-43  add-footer parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Examples

```
Service type=text/html method=GET fn=add-footer
file="footers/footer1.html"
```

```
Service type=text/html method=GET fn=add-footer
file="D:/Sun/WebServer61/server1/footers/footer1.html"
NSStringAbsFilePath="yes"
```

See Also
append-trailer, add-header
Table 4-44  add-header parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>(Optional) Path name to the file containing the header. Specify either file or uri.</td>
</tr>
<tr>
<td></td>
<td>By default, the path name is relative. If the path name is absolute, pass the NSIntAbsFilePath parameter as yes.</td>
</tr>
<tr>
<td>uri</td>
<td>(Optional) URI pointing to the resource containing the header. Specify either file or uri.</td>
</tr>
<tr>
<td>NSIntAbsFilePath</td>
<td>(Optional) If the file parameter is specified, the NSIntAbsFilePath parameter determines whether the file name is absolute or relative. The default is relative. Set the value to yes to indicate an absolute file path.</td>
</tr>
<tr>
<td>type</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Examples

Service type=text/html method=GET fn=add-header
file="headers/header1.html"

Service type=text/html method=GET fn=add-footer
file="D:/Sun/WebServer61/server1/headers/header1.html"
NSIntAbsFilePath="yes"
See Also
add-footer, append-trailer

append-trailer

Applicable in Service-class directives.

The append-trailer function sends an HTML file and appends text to the end. It only appends text to HTML files. This is typically used for author information and copyright text. The date the file was last modified can be inserted.

Returns REQ_ABORTED if a required parameter is missing, if there is extra path information after the file name in the URL, or if the file cannot be opened for read-only access.

Parameters
The following table describes parameters for the append-trailer function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>trailer</td>
<td>Text to append to HTML documents. The string is unescaped with util_uri_unescape before being sent. The text can contain HTML tags, and can be up to 512 characters long after unescaping and inserting the date. If you use the string :LASTMOD:, which is replaced by the date the file was last modified, you must also specify a time format with timefmt.</td>
</tr>
<tr>
<td>timefmt</td>
<td>(Optional) Time format string for :LASTMOD:. If timefmt is not provided, :LASTMOD: will not be replaced with the time.</td>
</tr>
<tr>
<td>type</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
</tbody>
</table>
The `imagemap` function responds to requests for imagemaps. Imagemaps are images that are divided into multiple areas that each have an associated URL. The information about which URL is associated with which area is stored in a mapping file.

**Parameters**
The following table describes parameters for the `imagemap` function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>type</code></td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td><code>method</code></td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td><code>query</code></td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
</tbody>
</table>

**Examples**

```plaintext
Service type=text/html method=GET fn=append-trailer
trailer="<hr><img src=/logo.gif> Copyright 1999"

# Add a trailer with the date in the format: MM/DD/YY
Service type=text/html method=GET fn=append-trailer timefmt="%D"
trailer="<HR>File last updated on: :LASTMOD:""
The index-common function generates a fancy (or common) list of files in the requested directory. The list is sorted alphabetically. Files beginning with a period (.) are not displayed. Each item appears as an HTML link. This function displays more information than index-simple, including the size, date last modified, and an icon for each file. It may also include a header and/or readme file into the listing.

The Init-class function cindex-init in magnus.conf specifies the format for the index list, including where to look for the images.

If obj.conf contains a call to index-common in the Service stage, magnus.conf must initialize fancy (or common) indexing by invoking cindex-init during the Init stage.

Indexing occurs when the requested resource is a directory that does not contain an index file or a home page, or no index file or home page has been specified by the functions find-index or home-page.

The icons displayed are .gif files dependent on the content-type of the file, as listed in the following table:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Example

```
Service type=magnus-internal/imagemap method=(GET|HEAD)
fn=imagemap
```
The following table describes parameters for the `index-common` function.

### Table 4-48  index-common parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>header</td>
<td>(Optional) Path (relative to the directory being indexed) and name of a file (HTML or plain text) that is included at the beginning of the directory listing to introduce the contents of the directory. The file is first tried with <code>.html</code> added to the end. If found, it is incorporated near the top of the directory list as HTML. If the file is not found, it is tried without the <code>.html</code> and incorporated as preformatted plain text (bracketed by <code>&lt;PRE&gt;</code> and <code>&lt;/PRE&gt;</code>).</td>
</tr>
<tr>
<td>readme</td>
<td>(Optional) Path (relative to the directory being indexed) and name of a file (HTML or plain text) to append to the directory listing. This file might give more information about the contents of the directory, indicate copyrights, authors, or other information. The file is first tried with <code>.html</code> added to the end. If found, it is incorporated at the bottom of the directory list as HTML. If the file is not found, it is tried without the <code>.html</code> and incorporated as preformatted plain text (enclosed by <code>&lt;PRE&gt;</code> and <code>&lt;/PRE&gt;</code>).</td>
</tr>
<tr>
<td>type</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
</tbody>
</table>
index-simple

Applicable in Service-class directives.

The index-simple function generates a simple index of the files in the requested directory. It scans a directory and returns an HTML page to the browser displaying a bulleted list of the files and directories in the directory. The list is sorted alphabetically. Files beginning with a period (.) are not displayed. Each item appears as an HTML link.

Indexing occurs when the requested resource is a directory that does not contain either an index file or a home page, or no index file or home page has been specified by the functions find-index or home-page.

Parameters

The following table describes parameters for the index-simple function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Example

```plaintext
Service fn=index-common type=magnus-internal/directory
method=(GET|HEAD) header=hdr readme=rdme.txt
```

See Also

index-simple, find-index, home-page
Table 4-49  index-simple parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeOut</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Example

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

See Also

*index-common*

**key-toosmall**

Applicable in Service-class directives.

**NOTE**  This function is provided for backward compatibility only and was deprecated in Sun ONE Web Server 4.x. It is replaced by the PathCheck-class SAF *ssl-check*. 
The `key-toosmall` function returns a message to the client specifying that the secret key size for SSL communications is too small. This function is designed to be used together with a Client tag to limit access of certain directories to nonexportable browsers.

**Parameters**

The following table describes parameters for the `key-toosmall` function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

**Example**

```xml
<Object ppath=/mydocs/secret/*>
  Service fn=key-toosmall
</Object>
```

**list-dir**

Applicable in Service-class directives.

The `list-dir` function returns a sequence of text lines to the client in response to a request whose method is `INDEX`. The format of the returned lines is:

```
name type size mimetype
```
The *name* field is the name of the file or directory. It is relative to the directory being indexed. It is URL-encoded, that is, any character might be represented by `%xx`, where `xx` is the hexadecimal representation of the character's ASCII number.

The *type* field is a MIME type such as `text/html`. Directories will be of type `directory`. A file for which the server doesn't have a type will be of type `unknown`.

The *size* field is the size of the file, in bytes.

The *mtime* field is the numerical representation of the date of last modification of the file. The number is the number of seconds since the epoch (Jan 1, 1970 00:00 UTC) since the last modification of the file.

When remote file manipulation is enabled in the server, the `obj.conf` file contains a `Service`-class function that calls `list-dir` for requests whose method is `INDEX`.

**Parameters**

The following table describes parameters for the `list-dir` function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>(Optional) Common to all <code>Service</code>-class functions.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all <code>Service</code>-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all <code>Service</code>-class functions.</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all <code>Service</code>-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all <code>Service</code>-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all <code>Service</code>-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all <code>Service</code>-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all <code>obj.conf</code> functions.</td>
</tr>
</tbody>
</table>

**Example**

```plaintext
Service fn=list-dir method="INDEX"
```
make-dir

Applicable in Service-class directives.

The make-dir function creates a directory when the client sends a request whose method is MKDIR. The function can fail if the server can’t write to that directory.

When remote file manipulation is enabled in the server, the obj.conf file contains a Service-class function that invokes make-dir when the request method is MKDIR.

Parameters
The following table describes parameters for the make-dir function.

**Table 4-52  make-dir parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Example

```
Service fn="make-dir" method="MKDIR"
```
match-browser

Applicable in all stage directives. The match-browser SAF matches specific strings in the User-Agent string supplied by the browser, and then modifies the behavior of Sun ONE Web Server based upon the results by setting values for specified variables. See match-browser.

query-handler

Applicable in Service- and Error-class directives.

NOTE This function is provided for backward compatibility only and is used mainly to support the obsolete ISINDEX tag. If possible, use an HTML form instead.

The query-handler function runs a CGI program instead of referencing the path requested.

Parameters
The following table describes parameters for the query-handler function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Full path and file name of the CGI program to run.</td>
</tr>
<tr>
<td>type</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>
Examples

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

**remove-dir**

Applicable in *Service*-class directives.

The *remove-dir* function removes a directory when the client sends a request whose method is **RMDIR**. The directory must be empty (have no files in it). The function will fail if the directory is not empty or if the server doesn’t have the privileges to remove the directory.

When remote file manipulation is enabled in the server, the *obj.conf* file contains a *Service*-class function that invokes *remove-dir* when the request method is **RMDIR**.

**Parameters**

The following table describes parameters for the *remove-dir* function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>(Optional) Common to all <em>Service</em>-class functions.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all <em>Service</em>-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all <em>Service</em>-class functions.</td>
</tr>
<tr>
<td><code>UseOutputStreamSize</code></td>
<td>(Optional) Common to all <em>Service</em>-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all <em>Service</em>-class functions.</td>
</tr>
<tr>
<td><code>ChunkedRequestBufferSize</code></td>
<td>(Optional) Common to all <em>Service</em>-class functions.</td>
</tr>
<tr>
<td><code>ChunkedRequestTimeout</code></td>
<td>(Optional) Common to all <em>Service</em>-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all <em>obj.conf</em> functions.</td>
</tr>
</tbody>
</table>
Example

```
Service fn="remove-dir" method="RMDIR"
```

**remove-file**

Applicable in Service-class directives.

The `remove-file` function deletes a file when the client sends a request whose method is `DELETE`. It deletes the file indicated by the URL if the user is authorized and the server has the needed file system privileges.

When remote file manipulation is enabled in the server, the `obj.conf` file contains a Service-class function that invokes `remove-file` when the request method is `DELETE`.

**Parameters**

The following table describes parameters for the `remove-file` function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all <code>obj.conf</code> functions.</td>
</tr>
</tbody>
</table>
Example

```
Service fn="remove-file" method="DELETE"
```

**remove-filter**

Applicable in Input-, Output-, Service-, and Error-class directives.

The `remove-filter` SAF is used to remove a filter from the filter stack. If the filter has been inserted multiple times, only the topmost instance is removed. In general, it is not necessary to remove filters with `remove-filter`, as they will be removed automatically at the end of the request.

**Returns**

Returns `REQ_PROCEED` if the specified filter was removed successfully, or `REQ_NOACTION` if the specified filter was not part of the filter stack. Any other return value indicates an error.

**Parameters**

The following table describes parameters for the `remove-filter` function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>Specifies the name of the filter to remove.</td>
</tr>
<tr>
<td>type</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>
Example

```
Service fn="remove-filter" filter="http-compression"
```

rename-file

Applicable in Service-class directives.

The rename-file function renames a file when the client sends a request with a New-URL header whose method is MOVE. It renames the file indicated by the URL to New-URL within the same directory if the user is authorized and the server has the needed file system privileges.

When remote file manipulation is enabled in the server, the `obj.conf` file contains a Service-class function that invokes `rename-file` when the request method is MOVE.

Parameters

The following table describes parameters for the rename-file function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all <code>obj.conf</code> functions.</td>
</tr>
</tbody>
</table>
send-cgi

Applicable in Service-class directives.

The send-cgi function sets up the CGI environment variables, runs a file as a CGI program in a new process, and sends the results to the client.

For details about the CGI environment variables and their NSAPI equivalents, see the Sun ONE Web Server 6.1 NSAPI Programmer's Guide.


There are three ways to change the timing used to flush the CGI buffer:

- Adjust the interval between flushes using the flushTimer parameter.
- Adjust the buffer size using the UseOutputStreamSize parameter.
- Force Sun ONE Web Server to flush its buffer by forcing spaces into the buffer in the CGI script.

Parameters

The following table describes parameters for the send-cgi function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>(UNIX only) Specifies the name of the user to execute CGI programs as.</td>
</tr>
<tr>
<td>group</td>
<td>(UNIX only) Specifies the name of the group to execute CGI programs as.</td>
</tr>
<tr>
<td>chroot</td>
<td>(UNIX only) Specifies the directory to chroot to before execution begins.</td>
</tr>
<tr>
<td>dir</td>
<td>(UNIX only) Specifies the directory to chdir to after chroot, but before execution begins.</td>
</tr>
</tbody>
</table>
### Table 4-58  send-cgi parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rlimit_as</td>
<td>(UNIX only) Specifies the maximum CGI program address space in bytes. You can supply both current (soft) and maximum (hard) limits, separated by a comma. The soft limit must be listed first. If only one limit is specified, both limits are set to this value.</td>
</tr>
<tr>
<td>rlimit_core</td>
<td>(UNIX only) Specifies the maximum CGI program core file size. A value of 0 disables writing cores. You can supply both current (soft) and maximum (hard) limits, separated by a comma. The soft limit must be listed first. If only one limit is specified, both limits are set to this value.</td>
</tr>
<tr>
<td>rlimit_nofile</td>
<td>(UNIX only) Specifies the maximum number of file descriptors for the CGI program. You can supply both current (soft) and maximum (hard) limits, separated by a comma. The soft limit must be listed first. If only one limit is specified, both limits are set to this value.</td>
</tr>
<tr>
<td>nice</td>
<td>(UNIX only) Accepts an increment that determines the CGI program’s priority relative to the server. Typically, the server is run with a nice value of 0 and the nice increment would be between 0 (the CGI program runs at same priority as server) and 19 (the CGI program runs at much lower priority than server). While it is possible to increase the priority of the CGI program above that of the server by specifying a nice increment of -1, this is not recommended.</td>
</tr>
<tr>
<td>type</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>
Example
The following example uses variables defined in the server.xml file for the send-cgi parameters. For more information about defining variables, see the Sun ONE Web Server 6.1 Administrator’s Configuration File Reference.

```
<Object name="default">
  ...
  NameTrans fn="pfx2dir" from="/cgi-bin"
  dir="/home/foo.com/public_html/cgi-bin" name="cgi"
  ...
</Object>

<Object name="cgi">
  ObjectType fn="force-type" type="magnus-internal/cgi"
  Service fn="send-cgi" user="$user" group="$group" dir="$dir"
  chroot="$chroot" nice="$nice"
</Object>
```

send-error
Applicable in Service-class directives.

The send-error function sends an HTML file to the client in place of a specific HTTP response status. This allows the server to present a friendly message describing the problem. The HTML page may contain images and links to the server’s home page or other pages.

Parameters
The following table describes parameters for the send-error function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Specifies the full file system path of an HTML file to send to the client. The file is sent as text/html regardless of its name or actual type. If the file does not exist, the server sends a simple default error page.</td>
</tr>
<tr>
<td>type</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
</tbody>
</table>
send-file

Applicable in Service-class directives.

The send-file function sends the contents of the requested file to the client. It provides the Content-Type, Content-Length, and Last-Modified headers.

Most requests are handled by this function using the following directive (which usually comes last in the list of Service-class directives in the default object, so that it acts as a default):

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

This directive is invoked if the method of the request is GET, HEAD, or POST, and the type does not start with magnus-internal/. Note that the pattern *~ means “does not match.” For a list of characters that can be used in patterns, see the Sun ONE Web Server 6.1 NSAPI Programmer’s Guide.

Parameters

The following table describes parameters for the send-file function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Example

```
Error fn=send-error code=401
path=/sun/server61/docs/errors/401.html
```

Table 4-59  send-error parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>
### Table 4-60  send-file parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nocache</td>
<td>(Optional) Prevents the server from caching responses to static file requests. For example, you can specify that files in a particular directory are not to be cached, which is useful for directories where the files change frequently. The value you assign to this parameter is ignored. If you do not wish to use this parameter, leave it out.</td>
</tr>
<tr>
<td>type</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

#### Example

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

In the following example, the server does not cache static files from `/export/somedir/` when requested by the URL prefix `/myurl`. 
send-range

Applicable in Service-class directives.

When the client requests a portion of a document, by specifying HTTP byte ranges, the send-range function returns that portion.

Parameters

The following table describes parameters for the send-range function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>
Example

| Service fn=send-range |

**send-shellcgi**

Applicable in Service-class directives.

**Windows Only.** The `send-shellcgi` function runs a file as a shell CGI program and sends the results to the client. Shell CGI is a server configuration that lets you run CGI applications using the file associations set in Windows. For information about shell CGI programs, consult the Sun ONE Web Server 6.1 *Administrator’s Guide*.

**Parameters**

The following table describes parameters for the `send-shellcgi` function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions</td>
</tr>
</tbody>
</table>
Examples

Service fn=send-shellcgi
Service type=magnus-internal/cgi fn=send-shellcgi

send-wincgi

Applicable in Service-class directives.

Windows Only. The send-wincgi function runs a file as a Windows CGI program and sends the results to the client. For information about Windows CGI programs, consult the Sun ONE Web Server 6.1 Administrator’s Guide.

Parameters

The following table describes parameters for the send-wincgi function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>
Examples

```
Service fn=send-wincgi
Service type=magnus-internal/cgi fn=send-wincgi
```

**service-dav**

Applicable in `Service`-class directives.

The `service-dav` function services a request to a WebDAV-enabled URI. In response to a request for a WebDAV resource, the `service-dav` function services the static content and restarts the request with the `sourceuri` for dynamic content. The `sourceuri` is identified by the `magnus-internal` setting. If no `sourceuri` is defined for dynamic content, an HTTP error message is returned.

Requests to WebDAV resources are authenticated and authorized by the `AuthTrans` and `PathCheck` NSAPI stages, respectively. By default, all access to `sourceuri` is restricted by the `PathCheck` entry in the `dav` object.

`OPTIONS` on a WebDAV-enabled URI are always handled by the default object’s `service-dav` directive. Therefore, the `OPTIONS` method is not included in the `service-dav` directive of the `dav` object.

In response to an OPTIONS request to a WebDAV-enabled uri (or `sourceuri`), the `service-dav` function in the default object adds the necessary DAV headers and returns control to the core server, which then services the request.

For more information on access control for WebDAV resources, see the Sun ONE Web Server 6.1 *Administrator’s Guide*.

**Parameters**

The following table describes parameters for the `service-dav` function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>method</code></td>
<td>(Optional) Common to all <code>Service</code>-class functions.</td>
</tr>
<tr>
<td><code>bucket</code></td>
<td>(Optional) Common to all <code>obj.conf</code> functions.</td>
</tr>
</tbody>
</table>
Examples

```
<Object name="default">
    ....
    Service
    method="(OPTIONS|PUT|DELETE|COPY|MOVE|PROPFIND|PROPPATCH|LOCK|UNLOCK|MKCOL)" fn="service-dav"
</Object>
```

```
<Object name="dav">
    PathCheck fn="check-acl" acl="dav-src"
    Service fn="service-dav"
    method="(PUT|DELETE|COPY|MOVE|PROPFIND|PROPPATCH|LOCK|UNLOCK|MKCOL)"
</Object>
```

See Also

stats-xml

service-dump

Applicable in Service-class directives.

The service-dump function creates a performance report based on collected performance bucket data (see “The bucket Parameter” on page 121).

To read the report, point the browser here:

```
http://server_id:port/.perf
```

Parameters

The following table describes parameters for the service-dump function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Must be perf for this function.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
</tbody>
</table>
Table 4-65  service-dump parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Examples

```xml
<Object name=default>
  NameTrans fn="assign-name" from="/perf" name="perf"
  ...
</Object>

<Object name=perf>
  Service fn="service-dump"
</Object>
```

See Also

stats-xml

service-j2ee

Applicable in Service-class directives.

The service-j2ee function services requests made to Java web applications.

Parameters

The following table describes parameters for the service-j2ee function.
The `service-trace` function services TRACE requests. TRACE requests are typically used to diagnose problems with web proxy servers located between a web client and web server.
Parameters
The following table describes parameters for the service_trace function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Example

```xml
<Object name="default">
...
Service method="TRACE" fn="service-trace"
...
</Object>
```

set-variable

Applicable in all stage directives. The set-variable SAF enables you to change server settings based upon conditional information in a request, and to manipulate variables in parameter blocks by using specific commands. See set-variable.

shtml_send

Applicable in Service-class directives.
The `shtml_send` function parses an HTML document, scanning for embedded commands. These commands may provide information from the server, include the contents of other files, or execute a CGI program. The `shtml_send` function is only available when the Shtml plugin (`libShtml.so` on UNIX `libShtml.dll` on Windows) is loaded. Refer to the Sun ONE Web Server 6.1 *Programmer’s Guide* for server-parsed HTML commands.

**Parameters**
The following table describes parameters for the `shtml_send` function.

**Table 4-68 shtml-send parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShtmlMaxDepth</td>
<td>Maximum depth of include nesting allowed. The default value is 10.</td>
</tr>
<tr>
<td>addCgiInitVars</td>
<td>(UNIX only) If present and equal to yes (the default is no), adds the environment variables defined in the init-cgi SAF to the environment of any command executed through the SHTML exec tag.</td>
</tr>
<tr>
<td>type</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

**Example**

```
Service type=magnus-internal/shtml_send method=(GET|HEAD)
fn=shtml_send
```
stats-xml

Applicable in Service-class directives.

The stats-xml function creates a performance report in XML format. If performance buckets have been defined, this performance report includes them.

However, you do need to initialize this function using the stats-init function in magnus.conf, then use a NameTrans function to direct requests to the stats-xml function. See the examples below.

The report is generated here:

http://server_id:port/stats-xml/iwsstats.xml

The associated DTD file is here:

http://server_id:port/stats-xml/iwsstats.dtd

For more information about the format of the iwsstats.xml file, see the Sun ONE Web Server 6.1 Performance Tuning, Sizing, and Scaling Guide

Parameters

The following table describes parameters for the stats-xml function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>method</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Examples

In magnus.conf:
In `obj.conf`:

```xml
<Object name="default">
  ...
  NameTrans fn="assign-name" from="/stats-xml/*" name="stats-xml"
  ...
</Object>
...
<Object name="stats-xml">
  Service fn="stats-xml"
</Object>
```

**See Also**

`service-dump`

**upload-file**

Applicable in `Service-class` directives.

The `upload-file` function uploads and saves a new file when the client sends a request whose method is `PUT` if the user is authorized and the server has the needed file system privileges.

When remote file manipulation is enabled in the server, the `obj.conf` file contains a `Service-class` function that invokes `upload-file` when the request method is `PUT`.

**Parameters**

The following table describes parameters for the `upload-file` function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>type</code></td>
<td>(Optional) Common to all <code>Service-class</code> functions.</td>
</tr>
</tbody>
</table>
AddLog

After the server has responded to the request, the AddLog directives are executed to record information about the transaction.

If there is more than one AddLog directive, all are executed.

The following AddLog-class functions are described in detail in this section:

- **common-log** records information about the request in the common log format.
- **flex-log** records information about the request in a flexible, configurable format.
- **match-browser** matches specific strings in the User-Agent string supplied by the browser, and then modifies the behavior of Sun ONE Web Server based upon the results by setting values for specified variables.
- **record-useragent** records the client’s IP address and User-Agent header.

---

**Table 4-70** upload-file parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>method</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>flushTimer</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>ChunkedRequestTimeout</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Example

```
Service fn=upload-file
```
set-variable enables you to change server settings based upon conditional information in a request, and to manipulate variables in parameter blocks by using specific commands.

common-log

Applicable in AddLog-class directives.

The common-log function records request-specific data in the common log format (used by most HTTP servers). There is a log analyzer in the /extras/log_anly directory for Sun ONE Web Server.

The common log must have been initialized previously by the init-clf function. For information about rotating logs, see flex-rotate-init in the Sun ONE Web Server 6.1 NSAPI Programmer’s Guide.

There are also a number of free statistics generators for the common log format.

Parameters

The following table describes parameters for the common-log function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>(Optional) Gives the name of a log file, which must have been given as a parameter to the init-clf function in magnus.conf. If no name is given, the entry is recorded in the global log file.</td>
</tr>
<tr>
<td>iponly</td>
<td>(Optional) Instructs the server to log the IP address of the remote client rather than looking up and logging the DNS name. This will improve performance if DNS is off in the magnus.conf file. The value of iponly has no significance, as long as it exists; you may use iponly=1.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>
Examples

```sh
# Log all accesses to the global log file
AddLog fn=common-log
# Log accesses from outside our subnet (198.93.5.*) to nonlocallog
<Client ip="*-198.93.5.*">
AddLog fn=common-log name=nonlocallog
</Client>
```

See Also

`record-useragent, flex-log`

flex-log

Applicable in `AddLog`-class directives.

The `flex-log` function records request-specific data in a flexible log format. It may also record requests in the common log format. There is a log analyzer in the `/extras/flexanlg` directory for Sun ONE Web Server.

There are also a number of free statistics generators for the common log format.

The log format is specified by the `flex-init` function call. For information about rotating logs, see `flex-rotate-init` in the Sun ONE Web Server 6.1 NSAPI Programmer’s Guide.

Parameters

The following table describes parameters for the `flex-log` function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>(Optional) Gives the name of a log file, which must have been given as a parameter to the <code>flex-init</code> function in <code>magnus.conf</code>. If no name is given, the entry is recorded in the global log file.</td>
</tr>
</tbody>
</table>
match-browser

Applicable in all stage directives. The `match-browser` SAF matches specific strings in the `User-Agent` string supplied by the browser, and then modifies the behavior of Sun ONE Web Server based upon the results by setting values for specified variables. See `match-browser`.
record-useragent

Applicable in AddLog-class directives.

The record-useragent function records the IP address of the client, followed by its User-Agent HTTP header. This indicates what version of the client was used for this transaction.

Parameters

The following table describes parameters for the record-useragent function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>(Optional) Gives the name of a log file, which must have been given as a parameter to the init-clf function in magnus.conf. If no name is given, the entry is recorded in the global log file.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Example

```bash
# Record the client ip address and user-agent to browserlog
AddLog fn=record-useragent name=browserlog
```

See Also

common-log, flex-log

set-variable

Applicable in all stage directives. The set-variable SAF enables you to change server settings based upon conditional information in a request, and to manipulate variables in parameter blocks by using specific commands. See set-variable.
If a Server Application Function results in an error, it sets the HTTP response status code and returns the value `REQ_ABORTED`. When this happens, the server stops processing the request. Instead, it searches for an `Error` directive matching the HTTP response status code or its associated reason phrase, and executes the directive’s function. If the server does not find a matching `Error` directive, it returns the response status code to the client.

The following `Error-class` functions are described in detail in this section:

- **error-j2ee** handles errors that occur during execution of Java™ 2 Platform, Enterprise Edition (J2EE™ platform) applications and modules deployed to the Sun ONE Web Server.
- **match-browser** matches specific strings in the `User-Agent` string supplied by the browser, and then modifies the behavior of Sun ONE Web Server based upon the results by setting values for specified variables.
- **qos-error** returns an error page stating which quality of service limits caused the error and what the value of the QOS statistic was.
- **query-handler** runs a CGI program instead of referencing the path requested.
- **remove-filter** removes a filter from the filter stack.
- **send-error** sends an HTML file to the client in place of a specific HTTP response status.
- **set-variable** enables you to change server settings based upon conditional information in a request, and to manipulate variables in parameter blocks by using specific commands.

**error-j2ee**

Applicable in `Error-class` directives.

The `error-j2ee` function handles errors that occur during execution of web applications deployed to the Sun ONE Web Server individually or as part of full J2EE applications.

**Parameters**

The following table describes parameters for the `error-j2ee` function.
match-browser

Applicable in all stage directives. The match-browser SAF matches specific strings in the User-Agent string supplied by the browser, and then modifies the behavior of Sun ONE Web Server based upon the results by setting values for specified variables. See `match-browser`.

qos-error

Applicable in Error-class directives.

The qos-error function returns an error page stating which quality of service limits caused the error, and what the value of the QOS statistic was.

The code for this SAF is one of the examples in the Sun ONE Web Server 6.1 NSAPI Programmer’s Guide.

For more information, see the Sun ONE Web Server 6.1 Performance Tuning, Scaling, and Sizing Guide.

Parameters

The following table describes parameters for the qos-error function.

### Table 4-74  error-j2ee Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>
**Table 4-75** qos-error parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| code      | (Optional) Three-digit number representing the HTTP response status code, such as 401 or 407. The recommended value is 503. This can be any HTTP response status code or reason phrase according to the HTTP specification. The following is a list of common HTTP response status codes and reason strings:  
  • 401 Unauthorized  
  • 403 Forbidden  
  • 404 Not Found  
  • 500 Server Error |
| bucket     | (Optional) Common to all obj.conf functions. |

**Example**

```
Error fn=qos-error code=503
```

**See Also**

`qos-handler`

**query-handler**

Applicable in Service- and Error-class directives.

**NOTE** This function is provided for backward compatibility only and is used mainly to support the obsolete `ISINDEX` tag. If possible, use an HTML form instead.

The `query-handler` function runs a CGI program instead of referencing the path requested.
Parameters
The following table describes parameters for the query-handler function.

Table 4-76 query-handler parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Full path and file name of the CGI program to run.</td>
</tr>
<tr>
<td>reason</td>
<td>(Optional) Text of one of the reason strings (such as &quot;Unauthorized&quot; or &quot;Forbidden&quot;). The string is not case-sensitive.</td>
</tr>
<tr>
<td>code</td>
<td>(Optional) Three-digit number representing the HTTP response status code, such as 401 or 407. This can be any HTTP response status code or reason phrase according to the HTTP specification. The following is a list of common HTTP response status codes and reason strings:</td>
</tr>
<tr>
<td></td>
<td>• 401 Unauthorized</td>
</tr>
<tr>
<td></td>
<td>• 403 Forbidden</td>
</tr>
<tr>
<td></td>
<td>• 404 Not Found</td>
</tr>
<tr>
<td></td>
<td>• 500 Server Error</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

Examples

Error query=*
fn=query-handler
path=/http/cgi/do-grep

Error query=*
fn=query-handler
path=/http/cgi/proc-info

remove-filter

Applicable in Input-, Output-, Service-, and Error-class directives.

The remove-filter SAF is used to remove a filter from the filter stack. If the filter has been inserted multiple times, only the topmost instance is removed. In general, it is not necessary to remove filters with remove-filter, as they will be removed automatically at the end of the request.
Returns

Returns `REQ_PROCEED` if the specified filter was removed successfully, or `REQ_NOACTION` if the specified filter was not part of the filter stack. Any other return value indicates an error.

Parameters

The following table describes parameters for the `remove-filter` function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>Specifies the name of the filter to remove.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all <code>obj.conf</code> functions.</td>
</tr>
</tbody>
</table>

Example

```
Error fn="remove-filter" filter="http-compression"
```

send-error

Applicable in `Error-class` directives.

The `send-error` function sends an HTML file to the client in place of a specific HTTP response status. This allows the server to present a friendly message describing the problem. The HTML page may contain images and links to the server’s home page or other pages.

Parameters

The following table describes parameters for the `send-error` function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Specifies the full file system path of an HTML file to send to the client. The file is sent as <code>text/html</code> regardless of its name or actual type. If the file does not exist, the server sends a simple default error page.</td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reason</td>
<td>(Optional) Text of one of the reason strings (such as “Unauthorized” or “Forbidden”). The string is not case-sensitive.</td>
</tr>
</tbody>
</table>
| code      | (Optional) Three-digit number representing the HTTP response status code, such as 401 or 407. This can be any HTTP response status code or reason phrase according to the HTTP specification. The following is a list of common HTTP response status codes and reason strings:  
  • 401 Unauthorized  
  • 403 Forbidden  
  • 404 Not Found  
  • 500 Server Error |
| bucket    | (Optional) Common to all obj.conf functions. |

set-variable

Applicable in all stage directives. The set-variable SAF enables you to change server settings based upon conditional information in a request, and to manipulate variables in parameter blocks by using specific commands. See set-variable.
This chapter discusses the MIME types file.

The sections are:

- Introduction
- Determining the MIME Type
- How the Type Affects the Response
- What Does the Client Do with the MIME Type?
- Syntax of the MIME Types File
- Sample MIME Types File

Introduction

The MIME types file in the config directory contains mappings between MIME (Multipurpose Internet Mail Extensions) types and file extensions. For example, the MIME types file maps the extensions .html and .htm to the type text/html:

```
type=text/html exts=htm,html
```

When the Sun ONE Web Server receives a request for a resource from a client, it uses the MIME type mappings to determine what kind of resource is being requested.
MIME types are defined by three attributes: language (\texttt{lang}), encoding (\texttt{enc}), and content-type (\texttt{type}). At least one of these attributes must be present for each type. The most commonly used attribute is \texttt{type}. The server frequently considers the \texttt{type} when deciding how to generate the response to the client. (The \texttt{enc} and \texttt{lang} attributes are rarely used.)

The default MIME types file is called \texttt{mime.types}.

**Determining the MIME Type**

During the Object\texttt{Type} step in the request handling process, the server determines the MIME type attributes of the resource requested by the client. Several different server application functions (SAFs) can be used to determine the MIME type, but the most commonly used one is \texttt{type-by-extension}. This function tells the server to look up the MIME type according to the requested resource's file extension in the MIME types table.

The directive in \texttt{obj.conf} that tells the server to look up the MIME type according to the extension is:

\texttt{ObjectType fn=type-by-extension}

If the server uses a different SAF, such as \texttt{force-type}, to determine the \texttt{type}, then the MIME types table is not used for that particular request.

For more details of the Object\texttt{Type} step, see the Sun ONE Web Server 6.1 NSAPI Programmer's Guide.

**How the Type Affects the Response**

The server considers the value of the \texttt{type} attribute when deciding which Service directive in \texttt{obj.conf} to use to generate the response to the client.

By default, if the \texttt{type} does not start with \texttt{magnus-internal/}, the server just sends the requested file to the client. The directive in \texttt{obj.conf} that contains this instruction is:

\texttt{Service method=(GET|HEAD|POST) type=*-magnus-internal/* fn=send-file}

By convention, all values of \texttt{type} that require the server to do something other than just send the requested resource to the client start with \texttt{magnus-internal/}. 
For example, if the requested resource’s file extension is .map, the type is mapped to magnus-internal/imagemap. If the extension is .cgi, .exe, or .bat, the type is set to magnus-internal/cgi:

<table>
<thead>
<tr>
<th>type</th>
<th>exts</th>
</tr>
</thead>
<tbody>
<tr>
<td>magnus-internal/imagemap</td>
<td>map</td>
</tr>
<tr>
<td>magnus-internal/cgi</td>
<td>cgi, exe, bat</td>
</tr>
</tbody>
</table>

If the type starts with magnus-internal/, the server executes whichever Service directive in obj.conf matches the specified type. For example, if the type is magnus-internal/imagemap, the server uses the imagemap function to generate the response to the client, as indicated by the following directive:

```
Service method=(GET|HEAD) type=magnus-internal/imagemap fn=imagemap
```

**What Does the Client Do with the MIME Type?**

The Service function generates the data and sends it to the client that made the request. When the server sends the data to the client, it also sends headers. These headers include whichever MIME type attributes are known (which is usually type).

When the client receives the data, it uses the MIME type to decide what to do with the data. For browser clients, the usual thing is to display the data in the browser window.

If the requested resource cannot be displayed in a browser but needs to be handled by another application, its type starts with application/, for example application/octet-stream (for .bin file extensions) or application/x-maker (for .fm file extensions). The client has its own set of user-editable mappings that tells it which application to use to handle which types of data.

For example, if the type is application/x-maker, the client usually handles it by opening Adobe® FrameMaker® to display the file.

**Syntax of the MIME Types File**

The first line in the MIME types file identifies the file format and must read:

```
#--Sun Microsystems MIME Information
```
Sample MIME Types File

Other non-comment lines have the following format:

type=type/subtype exts=[file extensions]

- type/subtype is the type and subtype.
- exts are the file extensions associated with this type.

Sample MIME Types File

Here is an example of a MIME types file:

```
#--Sun Microsystems MIME Information
# Do not delete the above line. It is used to identify the file type.
type=application/octet-stream exts=bin,exe
type=application/oda exts=oda
type=application/pdf exts-pdf

type=application/postscript exts=ai,eps,ps
type=application/x-mif exts=mif,fm

type=application/x-gtar exts=gtar

type=application/x-shar exts=shar

type=application/x-tar exts=tar

type=application/mac-binhex40 exts=hqx

type=audio/basic exts=au,snd

type=audio/x-aiff exts=aif,aiff,aifc

type=audio/x-wav exts=wav


type=image/gif exts=gif

type=image/ief exts=ief

type=image/jpeg exts=jpeg,jpg,jpeg

type=image/tiff exts=tiff,tif

type=image/x-rgb exts=rgb

type=image/x-xbitmap exts=xbm

type=image/x-xpixmap exts=xpm

type=image/x-xwindowdump exts=xwd


type=text/html exts=htm,html

type=text/plain exts=txt

type=text/x-setext exts=etx

type=video/mpeg exts=mpeg,mpg,mppe

type=video/quicktime exts=qt,mov

type=video/x-mavideo exts=avi


type=text/plain exts=txt

type=text/x-compress exts=x

type=text/x-uuencode exts=uu,ue


type=magnus-internal/imagemap exts=map

type=magnus-internal/parsed-html exts=html

type=magnus-internal/cgi exts=cgi,exe,bat

type=magnus-internal/jsp exts=jsp
```

Sample MIME Types File
Sample MIME Types File
This chapter summarizes the important configuration files not discussed in other chapters. Configuration files that should never be modified are not listed in this module.

The following configuration files are described in alphabetical order:

- certmap.conf
- dbswitch.conf
- Deployment Descriptors
- generated.instance.acl
- login.conf
- nsfc.conf
- password.conf
- server.policy
- *.clfilter

**Purpose**

Configures how a certificate, designated by `name`, is mapped to an LDAP entry, designated by `issuerDN`.
Location

```bash
server_root/bin/https/install/misc

server_root/userdb
```

Syntax

```bash
certmap name issuerDN
name:property1 [value1]
name:property2 [value2]
...
```

The default certificate is named `default`, and the default `issuerDN` is also named `default`. Therefore, the first certmap defined in the file must be as follows:

```bash
certmap default default
```

You can use `#` at the beginning of a line to indicate a comment.

See Also

*Sun ONE Web Server 6.1 Administrator’s Guide*

The following table describes properties in the `certmap.conf` file. The left column lists the property names. The second column from the left lists allowed values. The third column from the left lists default values. The right column lists property descriptions.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| DNComps     | See Description | Commented out | Used to form the base DN for performing an LDAP search while mapping the certificate to a user entry. Values are as follows:
  * Commented out: takes the user’s DN from the certificate as is.
  * Empty: searches the entire LDAP tree (DN == suffix).
  * Comma-separated attributes: forms the DN. |

---
dbswitch.conf

Table 6-1  certmap.conf properties

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FilterComps</td>
<td>See Description</td>
<td>Commented out</td>
<td>Used to form the filter for performing an LDAP search while mapping the certificate to a user entry. Values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Commented out or empty: sets the filter to &quot;objectclass=*&quot;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Comma-separated attributes: forms the filter.</td>
</tr>
<tr>
<td>verifycert</td>
<td>on or off</td>
<td>off</td>
<td>Specifies whether certificates are verified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(commented out)</td>
<td></td>
</tr>
<tr>
<td>CmapLdapAttr</td>
<td>LDAP attribute name</td>
<td>certSubject DN (commented out)</td>
<td>Specifies the name of the attribute in the LDAP database that contains the DN of the certificate.</td>
</tr>
<tr>
<td>library</td>
<td>Path to shared lib or dll</td>
<td>None</td>
<td>Specifies the library path for custom certificate mapping code.</td>
</tr>
<tr>
<td>InitFn</td>
<td>Name of initialization function</td>
<td>None</td>
<td>Specifies the initialization function in the certificate mapping code referenced by library.</td>
</tr>
</tbody>
</table>

dbswitch.conf

**Purpose**

Specifies the LDAP directory that Sun ONE Web Server uses.

**Location**

`server_root/userdb`

**Syntax**

```
directory name LDAP_URL
name:property1 [value1]
name:property2 [value2]
...
```

The default contents of this file are as follows:
directory default null:///none

Edit the file as follows for anonymous binding over SSL:

directory default ldaps://directory.sun.com:636:/dc%3Dcom

Edit the file as follows for anonymous binding not over SSL:

directory default ldap://directory.sun.com:389:/dc%3Dcom

See Also
User Database Selection

The following table describes properties in the `dbswitch.conf` file. The left column lists the property names. The second column from the left lists allowed values. The third column from the left lists default values. The right column lists property descriptions.

Table 6-2  `dbswitch.conf` properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nsessions</td>
<td>A positive integer</td>
<td>8</td>
<td>The number of LDAP connections for the database.</td>
</tr>
<tr>
<td>dyngroups</td>
<td>off, on, recursive</td>
<td>on</td>
<td>Determines how dynamic groups are handled. If off, dynamic groups are not supported. If on, dynamic groups are supported. If recursive, dynamic groups can contain other groups.</td>
</tr>
<tr>
<td>binddn</td>
<td>A valid DN</td>
<td></td>
<td>The DN used for connecting to the database. If both binddn and bindpw are not present, binding is anonymous.</td>
</tr>
<tr>
<td>bindpw</td>
<td></td>
<td></td>
<td>The password used for connecting to the database. If both binddn and bindpw are not present, binding is anonymous.</td>
</tr>
</tbody>
</table>
### Table 6-2  dbswitch.conf properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dcsuffix</td>
<td>A valid DN (relative to the LDAP URL)</td>
<td>none</td>
<td>If present, the default value of the base DN for the request’s virtual server is determined by a dc tree search of the connection group’s servername attribute, starting at the dcsuffix DN. If not present, the default value of the base DN is the base DN value in the LDAP URL. The basedn attribute of a USERDB element in the server.xml file overrides this value.</td>
</tr>
<tr>
<td>digestauth</td>
<td>off, on</td>
<td>off</td>
<td>Specifies whether the database can perform digest authentication. If on, a special Directory Server plugin is required. For information about how to install this plugin, see the Sun ONE Web Server 6.1 Administrator’s Guide.</td>
</tr>
<tr>
<td>syntax</td>
<td>keyfile, digest, htaccess</td>
<td>keyfile</td>
<td>Specifies what type of file auth-db will be used</td>
</tr>
<tr>
<td>keyfile</td>
<td></td>
<td></td>
<td>Specifies the path to the keyfile. Required, if syntax is set to keyfile.</td>
</tr>
<tr>
<td>digestfile</td>
<td></td>
<td></td>
<td>Specifies the path to the digestfile. Required, if syntax is set to digestfile.</td>
</tr>
<tr>
<td>groupfile</td>
<td></td>
<td></td>
<td>Path to the AuthGroupFile. If the groupfile is the same as the userfile, this file contains both user and group data, otherwise it contains only group data. Required if syntax is set to htaccess. For more information about the syntax of the AuthGroupFile, see the Sun ONE Web Server 6.1 Administrator’s Guide.</td>
</tr>
</tbody>
</table>
Deployment Descriptors

### Purpose
Configures features specific to the Sun ONE Web Server for deployed web applications.

### Location
The `META-INF` or `WEB-INF` directory of a module or application.

### See Also
The following table shows where to find more information about Sun ONE Web Server deployment descriptors. The left column lists the deployment descriptors, and the right column lists where to find more information about those descriptors.

### Table 6-3 Sun ONE Web Server deployment descriptors

<table>
<thead>
<tr>
<th>Deployment Descriptor</th>
<th>Where to Find More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>sun-web.xml</td>
<td>Sun ONE Web Server 6.1 Programmer’s Guide to Web Applications</td>
</tr>
</tbody>
</table>

**generated.instance.acl**

### Purpose
Sets permissions for access to the server instance. This is the default ACL file; you can create and use others.

<table>
<thead>
<tr>
<th>Property</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userfile</td>
<td></td>
<td></td>
<td>Path to the AuthUserFile. If the userfile is the same as the groupfile, this file contains both user and group data, otherwise it contains only user data. Required if syntax is set to <code>htaccess</code>. For more information about the syntax of the AuthUserFile, see the Sun ONE Web Server 6.1 Administrator’s Guide.</td>
</tr>
</tbody>
</table>
login.conf

**Location**

`server_root/config`

**See Also**

*Sun ONE Web Server 6.1 Administrator’s Guide*

**Purpose**

The login module definition configuration file used by the Java Authentication and Authorization Service (JAAS) for client authentication.

**Location**

`server_root/config`

nsfc.conf

**Purpose**

Sets file cache parameters. This file is present only if file cache parameters have been changed from their defaults.

**Location**

`server_root/https-admserv/config`

**Syntax**

`parameter=value`

**See Also**

*Performance Tuning, Sizing, and Scaling Guide for Sun ONE Web Server*

The following table describes properties in the `nsfc.conf` file. The left column lists the property names. The second column from the left lists allowed values. The third column from the left lists default values. The right column lists property descriptions.
### Table 6-4 nsfc.conf properties

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileCacheEnable</td>
<td>on, off</td>
<td>on</td>
<td>Enables the file cache.</td>
</tr>
<tr>
<td>CacheFileContent</td>
<td>on, off</td>
<td>on</td>
<td>Enables caching of file contents, as well as file information for files smaller than MediumFileSizeLimit (smaller than SmallFileSizeLimit if TransmitFile is on).</td>
</tr>
<tr>
<td>MaxAge</td>
<td>Number of seconds</td>
<td>30</td>
<td>The maximum age of a valid cache entry. This setting controls how long cached information is used once a file has been cached. An entry older than MaxAge is replaced by a new entry for the same file.</td>
</tr>
<tr>
<td>MediumFileSizeLimit</td>
<td>Limited by available memory</td>
<td>537600 (525K)</td>
<td>(UNIX only) Maximum size of a file that can be cached as a memory-mapped file (if TransmitFile is off).</td>
</tr>
<tr>
<td>MediumFileSpace</td>
<td>Limited by available memory</td>
<td>10485760 (10 M)</td>
<td>Total size of all files that are cached as memory-mapped files (if TransmitFile is off).</td>
</tr>
<tr>
<td>SmallFileSizeLimit</td>
<td>Limited by available memory</td>
<td>2048 (2K)</td>
<td>(UNIX only) Maximum size of a file that can be read into memory.</td>
</tr>
<tr>
<td>SmallFileSpace</td>
<td>Limited by available memory</td>
<td>1048576 (UNIX, 1 M), 0 (Windows)</td>
<td>Total size of all files that are read into memory.</td>
</tr>
<tr>
<td>TransmitFile</td>
<td>on, off</td>
<td>on</td>
<td>Enables use of the TransmitFile system call. Not supported on IRIX, Compaq, Solaris, or Linux.</td>
</tr>
<tr>
<td>MaxFiles</td>
<td></td>
<td>1024</td>
<td>Maximum number of files in the file cache.</td>
</tr>
<tr>
<td>HashInitSize</td>
<td>Limited by available memory</td>
<td>0</td>
<td>Initial number of hash buckets. If 0, the number of hash buckets is dynamically determined as (2 \times \text{MaxFiles} + 1).</td>
</tr>
</tbody>
</table>
### Purpose

By default, the Sun ONE Web Server prompts the administrator for the SSL key database password before starting up. If you want the Web server to be able to restart unattended, you need to save the password in a `password.conf` file. Be sure that your system is adequately protected so that this file and the key databases are not compromised.

### Location

`server_root/config`

This file is not present by default. You must create it if you need it.

### Syntax

```
PKCS#11_module_name:password
```

If you are using the internal PKCS#11 software encryption module that comes with the server, type the following:

```
internal:password
```

If you are using a different PKCS#11 module, for example for hardware encryption or hardware accelerators, you will need to specify the name of the PKCS#11 module, followed by the password.

### See Also

Sun ONE Web Server 6.1 *Administrator’s Guide*
server.policy

**Purpose**
Controls what access applications have to resources. This is the standard J2SE policy file. The J2SE SecurityManager is not active by default in Sun ONE Web Server 6.1. The policies granted in this policy file do not have any effect unless the SecurityManager is turned on in `server.xml`.

If you wish to use the J2SE SecurityManager you can turn it on by adding the following JVM options:

```
<JVMOPTIONS>-Djava.security.manager</JVMOPTIONS>
<JVMOPTIONS>-Djava.security.policy=server_root/config/server.policy</JVMOPTIONS>
```

**Location**
`server_root/config`

**Syntax**
grant [codeBase "path"] {  
  permission permission_class "package", "permission_type";
  ...
};

**See Also**
- Sun ONE Web Server Programmer's Guide

*.clfilter

**Purpose**
The files `obj.conf.clfilter`, `magnus.conf.clfilter`, and `server.xml.clfilter` contain filter specifications for cluster management operations.

**Location**
`server_root/config`
Appendix A

Configuration Changes Between iPlanet Web Server 4.1 and Sun ONE Web Server 6.1

This chapter summarizes major configuration file changes between the 4.1 and the 6.1 version of Sun ONE Web Server. The following 4.1 files are described:

- magnus.conf
- obj.conf
- contexts.properties
- rules.properties
- servlets.properties

magnus.conf

Table A-1 summarizes the changes in magnus.conf:

<table>
<thead>
<tr>
<th>4.x Directive</th>
<th>6.1 Directive</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcclFileCache</td>
<td>(none)</td>
<td>Obsolete because an NSAPI accelerator cache is no longer necessary</td>
</tr>
<tr>
<td>AcceptLanguage</td>
<td>(none)</td>
<td>See the acceptlanguage attribute of the VSCLASS and VS elements in server.xml</td>
</tr>
</tbody>
</table>
## Table A-1  magnus.conf changes

<table>
<thead>
<tr>
<th>4.x Directive</th>
<th>6.1 Directive</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACLFile</td>
<td>(none)</td>
<td>Maps to the <strong>ACLFILE</strong> element in server.xml</td>
</tr>
<tr>
<td>Address</td>
<td>(none)</td>
<td>Maps to the <strong>LS</strong> element in server.xml</td>
</tr>
<tr>
<td>AdminLanguage</td>
<td>(none)</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>AsyncDNS</td>
<td>AsyncDNS</td>
<td>Ignored. Even if the value is set to on, the server does not perform asynchronous DNS lookup.</td>
</tr>
<tr>
<td>BlockingListenSockets</td>
<td>(none)</td>
<td>See the blocking attribute of the <strong>LS</strong> element in server.xml.</td>
</tr>
<tr>
<td>CGIWaitPid</td>
<td>(none)</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>Ciphers</td>
<td>(none)</td>
<td>See the <strong>ssl2ciphers</strong> attribute of the <strong>SSLPARAMS</strong> element in server.xml</td>
</tr>
<tr>
<td>ClientLanguage</td>
<td>(none)</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>DefaultCharSet</td>
<td>(none)</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>ErrorLog</td>
<td>(none)</td>
<td>See the file attribute of the <strong>LOG</strong> element in server.xml.</td>
</tr>
<tr>
<td>IOTimeout</td>
<td>AcceptTimeout</td>
<td>Use the <strong>AcceptTimeout</strong> directive to specify the number of seconds the server must wait for data from a client before closing the connection.</td>
</tr>
<tr>
<td>LoadObjects</td>
<td>(none)</td>
<td>See the objectfile attribute in the <strong>VSCLASS</strong> element in server.xml.</td>
</tr>
<tr>
<td>LogVerbose</td>
<td>(none)</td>
<td>See the <strong>loglevel</strong> attribute in server.xml.</td>
</tr>
<tr>
<td>MaxThreads</td>
<td>(none)</td>
<td>Obsolete due to new thread handling system.</td>
</tr>
</tbody>
</table>
### Table A-1  magnus.conf changes

<table>
<thead>
<tr>
<th>4.x Directive</th>
<th>6.1 Directive</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MinProcs</td>
<td>(none)</td>
<td>Obsolete due to new thread handling system.</td>
</tr>
<tr>
<td>MinThreads</td>
<td>(none)</td>
<td>Obsolete due to new thread handling system.</td>
</tr>
<tr>
<td>MtaHost</td>
<td>(none)</td>
<td>Ignored.</td>
</tr>
<tr>
<td>NetsiteRoot</td>
<td>(none)</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>Port</td>
<td>(none)</td>
<td>See the LS element in server.xml.</td>
</tr>
<tr>
<td>RootObject</td>
<td>(none)</td>
<td>See the rootobject attribute of the VSCLASS element in server.xml.</td>
</tr>
<tr>
<td>RqThrottleMinPerSocket</td>
<td>(none)</td>
<td>See the the acceptorthreads attribute of the LS element in server.xml.</td>
</tr>
<tr>
<td></td>
<td>(none)</td>
<td>New. Specifies the number of request processing threads that are created when the server is started.</td>
</tr>
<tr>
<td>ServerID</td>
<td>(none)</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>ServerName</td>
<td>(none)</td>
<td>Deprecated. See the servername attribute of the LS element in the server.xml file.</td>
</tr>
<tr>
<td>#ServerRoot</td>
<td>(none)</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>SSL2</td>
<td>(none)</td>
<td>See the ssl2 attribute of the SSLPARAMS element in server.xml</td>
</tr>
<tr>
<td>SSL3</td>
<td>(none)</td>
<td>See the ssl3 attribute of the SSLPARAMS element in server.xml</td>
</tr>
<tr>
<td>SSL3Ciphers</td>
<td>(none)</td>
<td>See the the ssl3tlsciphers attribute of SSLPARAMS element in server.xml</td>
</tr>
<tr>
<td>SSLClientAuth</td>
<td>clientauth</td>
<td>See the clientauth attribute of the SSLPARAMS element in server.xml</td>
</tr>
</tbody>
</table>
The `obj.conf` file has lost its Init directives to the `magnus.conf` file and acquired new directives and parameters. Table A-2 summarizes the changes in the `obj.conf` file. Only the changed directives are listed.

### Table A-2  obj.conf changes

<table>
<thead>
<tr>
<th>4.x Directive</th>
<th>6.1 Directive</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>VirtualServerFile</td>
<td>(none)</td>
<td>Obsolete due to virtual server implementation</td>
</tr>
</tbody>
</table>

#### obj.conf

The `obj.conf` file has lost its Init directives to the `magnus.conf` file and acquired new directives and parameters. Table A-2 summarizes the changes in the `obj.conf` file. Only the changed directives are listed.

#### contexts.properties

The `contexts.properties` file is no longer supported. Servlet contexts or web applications are now defined in the `server.xml` file and configured using the `sun-web.xml` file.

A few `contexts.properties` functions are now in the `server.xml` file.

Table A-3 lists the equivalent functions in the `contexts.properties` and `sun-web.xml` files.
## Table A-3 contexts.properties to sun-web.xml correspondences

<table>
<thead>
<tr>
<th>contexts.properties Property</th>
<th>sun-web.xml Element or Attribute</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>sessionmgr</td>
<td>persistence-type attribute of the session-manager element</td>
<td>Specified using the context-param element in web.xml. For more information, please refer to the Servlet 2.3 specification. To add context attributes, implement the javax.servlet.ServletContextListener interface. For more information, please refer to the Servlet 2.3 specification.</td>
</tr>
<tr>
<td>sessionmgr.initArgs</td>
<td>manager-properties and store-properties attributes of the session-manager element</td>
<td></td>
</tr>
<tr>
<td>initArgs</td>
<td>(none)</td>
<td></td>
</tr>
<tr>
<td>respondCookieVersion</td>
<td>(none)</td>
<td>Will be supported in a future release.</td>
</tr>
<tr>
<td>tempDir</td>
<td>tempdir property</td>
<td></td>
</tr>
<tr>
<td>reloadInterval</td>
<td>dynamic-reload-interval attribute of class-loader element</td>
<td>Specified using the UseOutputStreamSize in obj.conf. See service-j2ee for more information.</td>
</tr>
<tr>
<td>bufferSize</td>
<td>(none)</td>
<td></td>
</tr>
<tr>
<td>docRoot</td>
<td>(none)</td>
<td>Specified in the server.xml file for each virtual server.</td>
</tr>
<tr>
<td>inputStreamLengthCheck</td>
<td>(none)</td>
<td>Obsolete.</td>
</tr>
<tr>
<td>outputStreamFlushTimer</td>
<td>(none)</td>
<td>Obsolete.</td>
</tr>
<tr>
<td>uri</td>
<td>uri attribute of WEBAPP element in server.xml.</td>
<td></td>
</tr>
<tr>
<td>authdb</td>
<td>authdb attribute of auth-native element</td>
<td>Obsolete.</td>
</tr>
</tbody>
</table>
rules.properties

The rules.properties file is no longer supported in Sun ONE Web Server 6.1. The function of the rules.properties file is now handled by the servlet-mapping element in the web.xml file. For more information, see the Servlet 2.3 API specification at:

http://java.sun.com/products/servlet/index.jsp

Table A-3 contexts.properties to sun-web.xml correspondences

<table>
<thead>
<tr>
<th>contexts.properties Property</th>
<th>sun-web.xml Element or Attribute</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>classpath</td>
<td>extra-class-path attribute of class-loader element</td>
<td>Obsolete because each web application has a single class loader as mandated by the Servlet 2.3 specification.</td>
</tr>
<tr>
<td>singleClassLoader</td>
<td>(none)</td>
<td>Specified in the server.xml file for each virtual server.</td>
</tr>
<tr>
<td>serverName</td>
<td>(none)</td>
<td>Obsolete due to web application support.</td>
</tr>
<tr>
<td>contentTypeIgnoreFromSSI</td>
<td>(none)</td>
<td>Obsolete.</td>
</tr>
<tr>
<td>parameterEncoding</td>
<td>parameter-encoding element</td>
<td></td>
</tr>
<tr>
<td>isModifiedCheckAggressive</td>
<td>(none)</td>
<td>Obsolete.</td>
</tr>
<tr>
<td>includeTransparency</td>
<td>(none)</td>
<td></td>
</tr>
</tbody>
</table>

rules.properties

The rules.properties file is no longer supported in Sun ONE Web Server 6.1. The function of the rules.properties file is now handled by the servlet-mapping element in the web.xml file. For more information, see the Servlet 2.3 API specification at:

http://java.sun.com/products/servlet/index.jsp

servlets.properties

The servlets.properties file is no longer supported for the default virtual server and other virtual servers. Most of the same functions are in the sun-web.xml file.

A few servlets.properties functions are in the server.xml file.

A few servlets.properties functions are in the web.xml file. For more information, see the Servlet 2.3 API specification at:

http://java.sun.com/products/servlet/index.jsp

Table A-4 lists the equivalent functions in the servlets.properties and sun-web.xml files.
## Table A-4  servlets.properties to sun-web.xml correspondences for individual servlet properties

<table>
<thead>
<tr>
<th>servlets.properties Property</th>
<th>sun-web.xml Element or Attribute</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>(none)</td>
<td>Specified in a <code>servlet-class</code> element in the <code>web.xml</code> file.</td>
</tr>
<tr>
<td>context</td>
<td>(none)</td>
<td>Obsolete because servlets are hosted within a web application which is deployed at the URI specified as the value of the <code>uri</code> attribute of the <code>WEBAPP</code> element in <code>server.xml</code>.</td>
</tr>
<tr>
<td>classpath</td>
<td>(none)</td>
<td>The Servlet 2.3 specification specifies that servlet classes be packaged in the WEB-INF/classes directory or in .jar archives in the WEB-INF/lib directory.</td>
</tr>
<tr>
<td>initArgs</td>
<td>(none)</td>
<td>Use the <code>init-param</code> element of the <code>&lt;servlet&gt;</code> tag in <code>web.xml</code> to specify servlet-specific initialization parameters.</td>
</tr>
<tr>
<td>startup</td>
<td>(none)</td>
<td>Specified in a <code>load-on-startup</code> element in the <code>web.xml</code> file.</td>
</tr>
</tbody>
</table>
Configuration Changes Between iPlanet Web Server 6.0 and Sun ONE Web Server 6.1

This chapter summarizes major configuration file changes between the 6.0 and the 6.1 version of Sun ONE Web Server. The following files are described:

- magnus.conf
- obj.conf
- server.xml

magnus.conf

This section lists the magnus.conf-related changes in the following areas:

- Init Functions
- Directives

Init Functions

The magnus.conf file in SUN ONE Web Server 6.1 has acquired new Init SAFs as listed in the following table:

<table>
<thead>
<tr>
<th>6.0 Function/Parameter</th>
<th>6.1 Function/Parameter</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSServletEarlyInit</td>
<td>(none)</td>
<td>Removed.</td>
</tr>
</tbody>
</table>
Directives

The magnus.conf file has lost directives to other configuration files and some directives supported by the magnus.conf file in previous releases are now deprecated. The following table summarizes the changes:

<table>
<thead>
<tr>
<th>6.0 Directive</th>
<th>6.1 Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdminLanguage</td>
<td>(none)</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>AsyncDNS</td>
<td>AsyncDNS</td>
<td>Ignored. Even if the value is set to on, the server does not perform asynchronous DNS lookup.</td>
</tr>
<tr>
<td>CGIWaitPid</td>
<td>(none)</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>ClientLanguage</td>
<td>(none)</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>DefaultCharSet</td>
<td>(none)</td>
<td>Ignored.</td>
</tr>
<tr>
<td>ErrorLog</td>
<td>(none)</td>
<td>See the file attribute of the LOG element in server.xml.</td>
</tr>
<tr>
<td>IOTimeout</td>
<td>AcceptTimeout</td>
<td>Use the AcceptTimeout directive to specify the number of seconds the server must wait for data from a client before closing the connection.</td>
</tr>
<tr>
<td>LogVerbose</td>
<td>(none)</td>
<td>See the loglevel attribute of the LOG element in server.xml.</td>
</tr>
<tr>
<td>LogVsid</td>
<td>logvsid</td>
<td>See the logvsid attribute of the LOG element in server.xml.</td>
</tr>
<tr>
<td>NetsiteRoot</td>
<td>(none)</td>
<td>Deprecated.</td>
</tr>
</tbody>
</table>

Table B-2 Changes in magnus.conf directives

Table B-1 magnus.conf Init functions

<table>
<thead>
<tr>
<th>6.0 Function/Parameter</th>
<th>6.1 Function/Parameter</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSServletLateInit</td>
<td>(none)</td>
<td>Removed.</td>
</tr>
<tr>
<td>nt-console-init</td>
<td>createconsole</td>
<td>Removed. On Windows, you can configure the createconsole attribute of the LOG element to redirect stderr output to the console.</td>
</tr>
</tbody>
</table>
The `obj.conf` file has acquired new SAFs and parameters as listed in Table 6-7. Only the new and changed directives are listed.

### Table B-3  `obj.conf` changes

<table>
<thead>
<tr>
<th>Supported in 6.0</th>
<th>Supported in 6.1</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSP092 object</td>
<td>(none)</td>
<td>Removed. Sun ONE Web Server 6.1 supports the JSP 2.3 specification and so, the JSP092 object is not required.</td>
</tr>
</tbody>
</table>

**server.xml**

This section describes the following changes:
- `server.xml` to `server.xml` correspondences
- `start-jvm` and `server.xml` correspondences
- `jvm12.conf` and `server.xml` correspondences

The following table lists the correspondences between the `server.xml` file in iPlanet Web Server 6.0 and the `server.xml` file in Sun ONE Web Server 6.1:

### Table B-4  `server.xml` to `server.xml` correspondences

| legacyls | Not supported. |
The following table lists the correspondences between the `start-jvm` file in iPlanet Web Server 6.0 to the `server.xml` file in Sun ONE Web Server 6.1:

### Table B-5  start-jvm and server.xml correspondences

<table>
<thead>
<tr>
<th>NSES_JDK</th>
<th>javahome</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSES_CLASSPATH</td>
<td>serverclasspath</td>
</tr>
<tr>
<td>NSES_JRE_RUNTIME_LIBPATH</td>
<td>nativelibrarypathprefix</td>
</tr>
<tr>
<td>NSES_JRE_RUNTIME_CLASSPATH</td>
<td>Use the <code>-Xbootclasspath</code> JVM option.</td>
</tr>
</tbody>
</table>

The following table lists the correspondences between the `jvm12.conf` file in iPlanet Web Server 6.0 and the `server.xml` file in Sun ONE Web Server 6.1:

### Table B-6  jvm12.conf and server.xml correspondences

<table>
<thead>
<tr>
<th>jvm.minHeapSize</th>
<th>Use the <code>-Xms&lt;value&gt;</code> JVM option. Example: <code>&lt;JVMOPTIONS&gt;-Xms128m -Xmx256m&lt;/JVMOPTIONS&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>jvm.maxHeapSize</td>
<td>Use the <code>-Xmx&lt;value&gt;</code> JVM option. Example: <code>&lt;JVMOPTIONS&gt;-Xms128m -Xmx256m&lt;/JVMOPTIONS&gt;</code></td>
</tr>
</tbody>
</table>
The following table lists the correspondences between the `web-apps.xml` file in iPlanet Web Server 6.0 and the `sun-web.xml` file in Sun ONE Web Server 6.1:

**Table B-7  web-apps.xml and sun-web.xml correspondences**

<table>
<thead>
<tr>
<th>jvm.enableClassGC</th>
<th>Use the <code>-Xnoclassgc</code> JVM option to disable garbage collection.</th>
</tr>
</thead>
<tbody>
<tr>
<td>jvm.option</td>
<td>Use the <code>JVMOPTIONS</code> element.</td>
</tr>
<tr>
<td>jvm.profiler</td>
<td>Use the <code>PROFILER</code> element.</td>
</tr>
<tr>
<td>jvm.verboseMode</td>
<td>Use the <code>-verbose</code> JVM option.</td>
</tr>
<tr>
<td>jvm.printErrors</td>
<td>Not supported.</td>
</tr>
<tr>
<td>jvm.disableThreadRecycling</td>
<td>Not supported.</td>
</tr>
<tr>
<td>jvm.serializeAttach</td>
<td>Not supported.</td>
</tr>
<tr>
<td>jvm.stickyAttach</td>
<td>Not supported.</td>
</tr>
<tr>
<td>jvm.trace</td>
<td>Configured in the <code>LOGLEVEL</code> element of the web container.</td>
</tr>
<tr>
<td>jvm.allowExit</td>
<td>Refer to information at the following URL for more information about configuring this in the <code>server.policy</code> file: <a href="http://java.sun.com/products/archive/j2se/1.4.1_07/index.html">http://java.sun.com/products/archive/j2se/1.4.1_07/index.html</a></td>
</tr>
<tr>
<td>jvm.include.CLASSPATH</td>
<td>Use the <code>envclasspathignored</code> attribute of the <code>JAVA</code> element.</td>
</tr>
<tr>
<td>jvm.enableDebug</td>
<td>Use the debug and debugoptions attributes of the <code>JAVA</code> element.</td>
</tr>
<tr>
<td>jvm.classpath</td>
<td>Use the <code>classpathprefix</code> and <code>classpathsuffix</code> attributes of the <code>JAVA</code> element.</td>
</tr>
</tbody>
</table>
Appendix C

Time Formats

This module describes the format strings used for dates and times in the server log. These formats are used by the NSAPI function `util_strftime`, by some built-in SAFs such as `append-trailer`, and by server-parsed HTML (`parse-html`).

The formats are similar to those used by the `strftime` C library routine, but not identical.

The following table describes the format strings for dates and times.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Allowed Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>%a</td>
<td>Abbreviated weekday name (3 chars)</td>
</tr>
<tr>
<td>%d</td>
<td>Day of month as decimal number (01-31)</td>
</tr>
<tr>
<td>%S</td>
<td>Second as decimal number (00-59)</td>
</tr>
<tr>
<td>%M</td>
<td>Minute as decimal number (00-59)</td>
</tr>
<tr>
<td>%H</td>
<td>Hour in 24-hour format (00-23)</td>
</tr>
<tr>
<td>%Y</td>
<td>Year with century, as decimal number, up to 2099</td>
</tr>
<tr>
<td>%b</td>
<td>Abbreviated month name (3 chars)</td>
</tr>
<tr>
<td>%h</td>
<td>Abbreviated month name (3 chars)</td>
</tr>
<tr>
<td>%T</td>
<td>Time &quot;HH:MM:SS&quot;</td>
</tr>
<tr>
<td>%X</td>
<td>Time &quot;HH:MM:SS&quot;</td>
</tr>
<tr>
<td>%A</td>
<td>Full weekday name</td>
</tr>
<tr>
<td>%B</td>
<td>Full month name</td>
</tr>
<tr>
<td>%C</td>
<td>&quot;%a %b %e %H:%M:%S %Y&quot;</td>
</tr>
<tr>
<td>%c</td>
<td>Date &amp; time &quot;%m/%d/%y %H:%M:%S&quot;</td>
</tr>
<tr>
<td>Attribute</td>
<td>Allowed Values</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------</td>
</tr>
<tr>
<td>%D</td>
<td>Date &quot;%m/%d/%y&quot;</td>
</tr>
<tr>
<td>%e</td>
<td>Day of month as decimal number (1-31) without leading zeros</td>
</tr>
<tr>
<td>%I</td>
<td>Hour in 12-hour format (01-12)</td>
</tr>
<tr>
<td>%j</td>
<td>Day of year as decimal number (001-366)</td>
</tr>
<tr>
<td>%k</td>
<td>Hour in 24-hour format (0-23) without leading zeros</td>
</tr>
<tr>
<td>%l</td>
<td>Hour in 12-hour format (1-12) without leading zeros</td>
</tr>
<tr>
<td>%m</td>
<td>Month as decimal number (01-12)</td>
</tr>
<tr>
<td>%n</td>
<td>line feed</td>
</tr>
<tr>
<td>%p</td>
<td>A.M./P.M. indicator for 12-hour clock</td>
</tr>
<tr>
<td>%R</td>
<td>Time &quot;%H:%M&quot;</td>
</tr>
<tr>
<td>%r</td>
<td>Time &quot;%I:%M:%S %p&quot;</td>
</tr>
<tr>
<td>%t</td>
<td>tab</td>
</tr>
<tr>
<td>%U</td>
<td>Week of year as decimal number, with Sunday as first day of week (00-51)</td>
</tr>
<tr>
<td>%W</td>
<td>Weekday as decimal number (0-6; Sunday is 0)</td>
</tr>
<tr>
<td>%W</td>
<td>Week of year as decimal number, with Monday as first day of week (00-51)</td>
</tr>
<tr>
<td>%x</td>
<td>Date &quot;%m/%d/%y&quot;</td>
</tr>
<tr>
<td>%y</td>
<td>Year without century, as decimal number (00-99)</td>
</tr>
<tr>
<td>%%</td>
<td>Percent sign</td>
</tr>
</tbody>
</table>
Appendix D

Alphabetical List of Server Configuration Elements

A
AUTHREALM 50

C
CONNECTIONPROPERTY 60
CUSTOMRESOURCE 53

D
DAVCOLLECTION 39
DESCRIPTION 25
DISPLAYNAME 43

E
EXTERNALJNDIRESOURCE 54

J
JAVA 45
JDBCCONNECTIONPOOL 56
JDBCRESOURCE 55
JVMOPTIONS 47

L
LS 26

M
MAILRESOURCE 61
MIME 30

P
PROFILER 48
PROPERTY 24

Q
QOSPARAMS 36

R
RESOURCES 53

S
SEARCH 41
SEARCHCOLLECTION 42
SECURITY 49
SERVER 22
SSLPARAMS 28

U
USERDB 37

V
VARS 25
VS 34
VSCLASS 32

W
WEBAPP 43
Appendix E

Alphabetical List of Predefined SAFs

This appendix provides an alphabetical list for the easy lookup of predefined SAFs.

A

add-footer
add-header
append-trailer
assign-name

B

basic-auth
basic-ncsa

C

check-acl
common-log
deny-existence
document-root

error-j2ee

find-compressed
find-index
find-links
find-pathinfo
flex-log
force-type

get-client-cert
get-sslid

home-page

imagemap
Appendix E Alphabetical List of Predefined SAFs

insert-filter
insert-filter

K
key-tosmall

L
list-dir
load-config

M
make-dir
match-browser

N
ntcgicheck
ntrans-dav
ntrans-j2ee
nt-uri-clean

P
pcheck-dav
pfx2dir
Q
qos-error
qos-handler
query-handler
query-handler

R
record-useragent
redirect
remove-dir
remove-file
remove-filter
remove-filter
remove-filter
remove-filter
rename-file
require-auth

S
send-cgi
send-error
send-error
send-file
send-range
send-shellcgi
send-wincgi
service-dav
service-dump
service-j2ee
service-trace
set-default-type
set-variable
set-virtual-index
shtml_send
shtml-hacktype
ssl-check
ssl-logout
stats-xml
strip-params

t

type-by-exp
type-by-extension

u

unix-home
unix-uri-clean
upload-file
SYMBOLS
.clfilter files 236

A
AccelFileCache directive 237
acceptlanguage 37
AcceptLanguage directive 237
AcceptTimeout
 magus.conf directive 83
access log 66
ACL
 magus.conf directives 93
 acl parameter 146
ACLCacheLifetime
 magus.conf directive 93
ACLCacheLifetime directive 109
ACLFILE 35
ACLFile directive 238
ACLGroupCacheSize
 magus.conf directive 93
ACLGroupCacheSize directive 109
ACLUserCacheSize
 magus.conf directive 93
ACLUserCacheSize directive 109
addCgiInitVars parameter 207
add-footer function 177
add-header function 178
AddLog
 function descriptions 210
Address directive 238
Administration interface
 more information about 11
AdminLanguage
 magus.conf directive 99
AdminLanguage directive 238, 246
alias directory 17
alphabetical reference
 SAFs 257
append-trailer function 180
assign-name function 134
AsyncDNS
 magus.conf directive 81
AsyncDNS directive 109, 238, 246
authdb property 241
auth-group parameter 159
AUTHREALM 54
AuthTrans
 function descriptions 121
auth-type parameter 123, 124, 158
auth-user parameter 159

B
basedir parameter 155
basic-auth function 122
basic-ncsa function 124
Section C

bin directory 18
binddn property 230
bindpw property 230
BlockingListenSockets directive 238
bong-file parameter 148, 161
bucket parameter 121
buffer-size parameter 104
buffers-per-file parameter 104, 213
built-in SAFs 117

CacheFileContent parameter 234
cache-size parameter 103
certificates
  settings in magnus.conf 94
CGI
  settings in magnus.conf 89
CGIExpirationTimeout
  magnus.conf directive 90
CGIExpirationTimeout directive 109
CGISubIdleTimeout
  magnus.conf directive 90
CGISubIdleTimeout directive 109
cgistub-path parameter 105
CGIWaitPid
  magnus.conf directive 90
CGIWaitPid directive 109, 238
charset parameter 164, 165, 167
check-acl function 145
checkFileExistence parameter 150
ChildRestartCallback
  magnus.conf directive 98
ChildRestartCallback directive 110
chroot parameter 194
chunked encoding 96
ChunkedRequestBodySize
  magnus.conf directive 97
  obj.conf Service parameter 175
ChunkedRequestBodySize directive 110

CONNECTIONPROPERTY 63

certificates
  settings in magnus.conf 94
CGI
  settings in magnus.conf 89
CGIExpirationTimeout
  magnus.conf directive 90
CGIExpirationTimeout directive 109
CGISubIdleTimeout
  magnus.conf directive 90
CGISubIdleTimeout directive 109
cgistub-path parameter 105
CGIWaitPid
  magnus.conf directive 90
CGIWaitPid directive 109, 238
charset parameter 164, 165, 167
check-acl function 145
checkFileExistence parameter 150
ChildRestartCallback
  magnus.conf directive 98
ChildRestartCallback directive 110
chroot parameter 194
chunked encoding 96
ChunkedRequestBodySize
  magnus.conf directive 97
  obj.conf Service parameter 175
ChunkedRequestBodySize directive 110

CopyFiles parameter 235
core SAFs 117
Core Server Elements 26
createconsole 67
CUSTOMRESOURCE 57

D
DaemonStats directive 238
dav 42
davcollection 43
day of month 251
dbm parameter 124
dcsuffix property 231
default virtual server
  for a connection group 32
DefaultCharSet directive 238, 246
defaultLanguage
  magnus.conf directive 80
defaultLanguage directive 110
define-perf-bucket function 103
deny-existence function 148
descend parameter 155
description parameter 103
digest directory 19
digestauth property 231
digestfile 231
dir parameter 140, 150, 194
directives
  obj.conf 117
disable parameter 107, 150
disable-types parameter 155
DISPLAYNAME 47
DNComps property 228
DNS
  magnus.conf directive 81
DNS directive 110
DNS lookup
  directives in magnus.conf 81
dns-cache-init function 103
docRoot property 241
docs directory 18
document-root function 136
domain component tree 68
domain component tree (dc) 69
dorequest parameter 152
dotdirok parameter 152, 162
DTD
  Attributes 25
  Data 25
  Subelements 24
dynamic reconfiguration
  overview 21
dyngroups property 230

E
Elements in the server.xml File 26
cenc parameter 164, 165, 167, 222
encoding
  chunked 96
Error directive
  function descriptions 215
error logging
  settings in magnus.conf 91
ErrorLog directive 246
ErrorLogDateFormat
  magnus.conf directive 92
ErrorLogDateFormat directive 110
errors
  sending customized messages 217, 218, 220
errors log 65
escape parameter 142
exec-hack parameter 166
exp parameter 167
expire parameter 103
extension parameter 157
EXTERNALJNDIRESOURCE 58
ExtraPath
  magnus.conf directive 78
ExtraPath directive 110
extras directory 18
F
Favicon
magnus.conf directive 98
file name extensions
MIME types 221
file parameter 155, 177, 179
FileCacheEnable parameter 234
files
mapping types of 221
filter parameter 170, 171, 172, 173
FilterComps property 229
find-index function 149
find-links function 150
find-pathinfo function 151
find-pathinfo-forward parameter 135, 140
flexanlg directory 18
flex-init function 103
flex-log function 212
flex-rotate-init function 104
flushTimer parameter 174
force-type function 163
format parameter 102, 103
free-size parameter 107
from parameter 134, 140, 141, 160
funcs parameter 106

G
get-client-cert function 151
get-sslid function 125
group parameter 194
groupdb parameter 123
groupfn parameter 123
grpfile parameter 125

H
hard links
finding 150
HashInitSize parameter 234
header parameter 183
HeaderBufferSize
magnus.conf directive 84
HeaderBufferSize directive 111
home-page function 137
httpacl directory 18
http-compression filter 147
https-admserv directory 18
https-server_id.domain 19
HTTPVersion
magnus.conf directive 98
HTTPVersion directive 111
HUP signal
PidLog and 92

I
icon-uri parameter 102
ignore parameter 102
imagemap function 181
include directory 19
index-common function 182
index-names parameter 149
index-simple function 184
inetOrgPerson
in convergence tree 69
Init
function descriptions 78
Init functions 101, 240, 245
initArgs property 241, 243
init-cgi function 105
init-clf function 106
InitFn property 229
init-uhome function 106
Input
function descriptions 168
inputStreamLengthCheck property 241
insert-filter SAF 169, 172
iponly function 211, 213
isModifiedCheckAggressive property 242

J
J2SE SecurityManager 236
JAVA 49
Java Configuration Elements 49
JDBCCONNECTIONPOOL 59
JDBCRESOURCE 59
JVMOPTIONS 52

K
KeepAliveQueryMaxSleepTime
magnus.conf directive 84
KeepAliveQueryMeanTime
magnus.conf directive 84
KeepAliveThreads directive 112
KeepAliveTimeout
magnus.conf directive 84, 85
KeepAliveTimeout directive 112
KernelThreads
magnus.conf directive 85
KernelAttached directive 112
keyfile 231
key-toosmall function 185

L
lang parameter 164, 165, 167, 222
language issues
directives in magnus.conf 80
LDAP
iPlanet schema 68
lib directory 20
library property 229
LICENSE.txt 20
links
finding hard links 150
list-dir function 186
Listener Elements 29
ListenerQ
magnus.conf directive 85
ListenerQ directive 112
loadbal directory 20
load-config function 153
load-modules function 106
LoadObjects directive 238
LOG 65
log analyzer 211, 212
log file
analyzer for 211, 212
log_anly directory 18
LogFlushInterval directive 112
logging
settings in magnus.conf 91
login.conf 233
logs directory 18, 19
logs file 20
logstderr 67
logstdout 67
logtoconsole 67
LogVerbose directive 238, 246
LS
id 31
ip attribute 31

M
magnus.conf
changes to 237, 245
miscellaneous directives 97
MAILRESOURCE 64
make-dir function 188
manual directory 19
match-browser function 126
MaxAge parameter 234
MaxCGIStubs
  magnus.conf directive 91
MaxCGIStubs directive 112
MaxFiles parameter 234
MaxKeepAliveConnections
  magnus.conf directive 85
MaxKeepAliveConnections directive 113
MaxProcs
  magnus.conf directive 86
MaxProcs directive 113
MaxRqHeaders
  magnus.conf directive 98
MaxRqHeaders directive 113
MaxThreads directive 238
maxthreads parameter 108
MediumFileSizeLimit parameter 234
MediumFileSpace parameter 234
method parameter 152, 174
methods function 107
mime.types file 222
  sample of 224
  syntax 223
MinCGIStubs
  magnus.conf directive 91
MinCGIStubs directive 113
MinProcs directive 239
MinThreads directive 239
minthreads parameter 108
MMapSessionManager 18, 19
month name 251
MtaHost directive 239

settings in magnus.conf 88
NativePoolMaxThreads
  magnus.conf directive 89
NativePoolMaxThreads directive 113
NativePoolMinThreads
  magnus.conf directive 89
NativePoolMinThreads directive 113
NativePoolQueueSize
  magnus.conf directive 89
NativePoolQueueSize directive 113
NativePoolStackSize
  magnus.conf directive 89
NativePoolStackSize directive 113
NativeThread parameter 107
NetSiteRoot
  magnus.conf directive 99
NetSiteRoot directive 239, 246
nice parameter 195
nocache parameter 198
nostat parameter 135
nsacl directory 20
nsapi directory 20
NSCP_POOL_STACKSIZE 88
NSCP_POOL_THREADMAX 88
NSCP_POOL_WORKQUEUEMAX 88
nsessions property 230
nsfc.conf 233
NSIntAbsFilePath parameter 177, 179
ntcgiCHECK function 157
nt-console-init function 107, 246
nttrans-base 135, 140
nt-uri-clean function 156
num-buffers parameter 104

N
name parameter 134, 140, 144, 211, 212
  of define-perf-bucket function 103
  of thread-pool-init function 108
NameTrans
  function descriptions 133
native thread pools

obj.conf
  changes to 240, 247
directives 117
objectfile 36
ObjectType
<table>
<thead>
<tr>
<th>Function Descriptions</th>
<th>Parameter Encoding</th>
<th>QOSPARAMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>opts parameter 101</td>
<td>function descriptions 162</td>
<td>qosrecomputeinterval 27</td>
</tr>
<tr>
<td>Output function</td>
<td>qos-error function 216</td>
<td>query parameter 174</td>
</tr>
<tr>
<td>function descriptions 171</td>
<td>qos-handler function 127</td>
<td>query-handler function 189, 217</td>
</tr>
<tr>
<td>outputStreamFlushTimer property 241</td>
<td>qosmetricsinterval 27</td>
<td>queueSize parameter 108</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter Encoding</th>
<th>Parameter Encoding</th>
<th>QOSPARAMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameterEncoding property 242</td>
<td>RcvBufSize</td>
<td>magnus.conf directive 86</td>
</tr>
<tr>
<td>parse-html function 240</td>
<td>RcvBufSize directive 113</td>
<td>readme parameter 183</td>
</tr>
<tr>
<td>path parameter 137, 146, 148, 158, 189, 196, 218, 219</td>
<td>README.txt 20</td>
<td>realm parameter 158</td>
</tr>
<tr>
<td>PathCheck function descriptions 144</td>
<td>reason parameter 218, 220</td>
<td>reason parameter 218, 220</td>
</tr>
<tr>
<td>pcheck-dav function 157</td>
<td>record-useragent function 214</td>
<td>redirect function 141</td>
</tr>
<tr>
<td>perf-init function 107</td>
<td>RcvBufSize directive 113</td>
<td>reloadInterval property 241</td>
</tr>
<tr>
<td>pfx2dir function 139</td>
<td>remove-dir function 190</td>
<td>remove-dir function 190</td>
</tr>
<tr>
<td>PidLog</td>
<td>remove-file function 191</td>
<td>remove-file function 191</td>
</tr>
<tr>
<td>magnus.conf directive 92</td>
<td>remove-filter SAF 170, 173</td>
<td>remove-filter SAF 170, 173</td>
</tr>
<tr>
<td>PidLog directive 113</td>
<td>rename-file function 193</td>
<td>rename-file function 193</td>
</tr>
<tr>
<td>plugins directory 19</td>
<td>require parameter 152</td>
<td>require parameter 152</td>
</tr>
<tr>
<td>pool parameter 107</td>
<td>require-auth function 158</td>
<td>require-auth function 158</td>
</tr>
<tr>
<td>pool-init function 107</td>
<td>Resource Elements 55</td>
<td>Resource Elements 55</td>
</tr>
<tr>
<td>Port directive 239</td>
<td>RESOURCES 56</td>
<td>RESOURCES 56</td>
</tr>
<tr>
<td>PostThreadsEarly</td>
<td>respondCookieVersion property 241</td>
<td>restart file 20</td>
</tr>
<tr>
<td>magnus.conf directive 86</td>
<td>rlimit_as parameter 195</td>
<td>rlimit_as parameter 195</td>
</tr>
<tr>
<td>PostThreadsEarly directive 113</td>
<td>rlimit_core parameter 195</td>
<td>rlimit_core parameter 195</td>
</tr>
<tr>
<td>predefined SAFs 117</td>
<td>rlimit_nofile parameter 195</td>
<td>rlimit_nofile parameter 195</td>
</tr>
<tr>
<td>processes</td>
<td>root parameter 137</td>
<td>root parameter 137</td>
</tr>
<tr>
<td>settings in magnus.conf 81</td>
<td>rootobject 37</td>
<td>rootobject 37</td>
</tr>
<tr>
<td>PROFILER 52</td>
<td>RootObject directive 239</td>
<td>rotate-access parameter 105</td>
</tr>
<tr>
<td>profiling parameter 108</td>
<td>rotate-access parameter 105</td>
<td>rotate-access parameter 105</td>
</tr>
<tr>
<td>pwfile parameter 106, 144</td>
<td>rotate-callback parameter 105</td>
<td>rotate-callback parameter 105</td>
</tr>
<tr>
<td></td>
<td>rotate-error parameter 105</td>
<td>rotate-error parameter 105</td>
</tr>
<tr>
<td></td>
<td>rotate-interval parameter 104</td>
<td>rotate-interval parameter 104</td>
</tr>
<tr>
<td></td>
<td>rotate-start parameter 104</td>
<td>rotate-start parameter 104</td>
</tr>
</tbody>
</table>
RqThrottle
  magnus.conf directive 86
RqThrottle directive 114
RqThrottleMin
  magnus.conf directive 87
RqThrottleMinPerSocket directive 114
rules.properties
  changes to 242

S
SAFs
  alphabetical reference 257
  Init 78
  predefined 117
samples directory 19
SEARCH 45
search directory 19, 20
Search Elements 45
SEARCHCOLLECTION 46
secret-keysize parameter 161
Security
  magnus.conf directive 94
SECURITY 53
security
  settings in magnus.conf 94
Security directive 114
send-cgi function 194
send-error function 196, 219
send-file function 197
send-shellcgi function 200
send-wincgi function 201
server
  handling of authorization of client users 121
  HUP signal 92
  killing process of 92
  TERM signal 92
server information
  magnus.conf directives 78
server.policy 236
server.xml 23
  more information 168
  variables defined in 196
server.xml elements
  ACLFILE 35
  AUTHREALM 54
  CONNECTIONPROPERTY 63
  CUSTOMRESOURCE 57
  DAV 42
  DAVCOLLECTION 43
  DESCRIPTION 29
  DISPLAYNAME 47
  EXTERNALJNDIRESOURCE 58
  JAVA 49
  JDBC CONNECTIONPOOL 59
  JDBCRESOURCE 59
  JVMOPTIONS 52
  LOG 65
  LS 30
  MAILRESOURCE 64
  MIME 34
  PROFILER 52
  PROPERTY 28
  QOSPARAMS 40
  RESOURCES 56
  SEARCH 45
  SEARCHCOLLECTION 46
  SECURITY 53
  SERVER 26
  SSLPARAMS 32
  USERDB 40
  VARS 29
  VS 37
  VSCLASS 35
  WEBAPP 48
servercertnickname 33
ServerID
  magnus.conf directive 100
ServerName directive 239
serverName property 242
ServerRoot
  magnus.conf directive 100
ServerRoot directive 239
Service
  function descriptions 174
  service-dav function 202
  service-dump function 203
servlets directory 19
servlets.properties
  changes to 242
SessionData 18
SessionData directory 19
SessionData file 20
sessionmgr property 241
sessionmgr.initArgs property 241
set-default-type function 164
setup directory 20
set-variable function 128
set-virtual-index function 159
shlib parameter 106
shtml_send function 206, 240
shtml-hacktype function 166
ShtmlMaxDepth parameter 207
singleClassLoader property 242
SmallFileSizeLimit parameter 234
SmallFileSpace parameter 234
SndBufSize
  magnus.conf directive 87
SndBufSize directive 114
snmp directory 20
SSL
  settings in magnus.conf 94
ssl2 33
SSL2 directive 239
ssl2ciphers 33
ssl3 33
SSL3 directive 239
SSL3Ciphers directive 239
SSLSessionTimeout
  magnus.conf directive 96
SSLSessionTimeout directive 114
ssltlsCiphers 33
SSLCacheEntries
  magnus.conf directive 94, 95
SSLCacheEntries directive 114
ssl-check function 160
SSLClientAuth directive 239
SSLClientAuthDataLimit
  magnus.conf directive 95
SSLClientAuthDataLimit directive 114
SSLClientAuthTimeout
  magnus.conf directive 95
SSLClientAuthTimeout directive 114
ssl-logout function 161
SSLSessionTimeout
  magnus.conf directive 95
SSLSessionTimeout directive 115
StackSize
  magnus.conf directive 87
StackSize directive 115
StackSize parameter 108
start file 21
startconsole file 20
startsvr.bat 18, 19
startup property 243
statistic collection
  settings in magnus.conf 91
stats-init function 107
stderr parameter 107
stdout parameter 107
stop file 21
stopsvr.bat 18, 19
StrictHttpHeaders
  magnus.conf directive 87
StrictHttpHeaders directive 115
strip-params function 143
subdir parameter 144
Sun ONE LDAP Schema 68
sun-web-server_6_1.dtd 23
symbolic links
  finding 150
syntax 231
  mime.types file 223

TempDir
  magnus.conf directive 79
TempDir directive 115
TempDir parameter 235
tempDir property 241
TempDirSecurity
  magnus.conf directive 79
TempDirSecurity directive 115
TERM signal 92
TerminateTimeout
  magnus.conf directive 87
TerminateTimeout directive 115
thread pools
  settings in magnus.conf 88
ThreadIncrement
  magnus.conf directive 88
ThreadIncrement directive 115
thread-pool-init function 108
threads
  settings in magnus.conf 81
tildeok parameter 156
timefmt parameter 180
timeout parameter 105
timezone parameter 102
tls 33
tlsrollback 34
trailer parameter 180
TransmitFiles parameter 234
type parameter 163, 167, 174, 222
type-by-exp function 166
type-by-extension 222
type-by-extension function 167
uri property 241
URL
  mapping to other servers 139
url parameter 141
url-prefix parameter 142
UseNativePoll
  magnus.conf directive 88
UseNativePoll directive 116
UseOutputStreamSize
  magnus.conf directive 96
  obj.conf Service parameter 174
UseOutputStreamSize directive 116
User
  magnus.conf directive 79
user account
  specifying 79
User Database Selection 67
User directive 116
user home directories
  symbolic links and 150
user parameter 194
USERDB 40, 67
userdb directory 20
userdb parameter 123
userfile 232
userfile parameter 125
userfn parameter 123
usesyslog 67
util_strftime 251

U
Umask
  magnus.conf directive 98
Umask directive 115
Unix user account
  specifying 79
unix-home function 143
unix-uri-clean function 161
update-interval parameter 108
upload-file function 209
uri parameter 177, 179

V
Variable Evaluation 72
Variables
  send-cgi Variables 71
vARIABLES
  General Variables 71
  send-cgi Variables 71
verifystart property 229
virtual server log 66
virtual-index parameter 160
VirtualServerFile directive 240
virtual-servers parameter 108
VS 37
VSCLASS 35
  id 36

W
Web Application Elements 47
WEBAPP 48
WebDAV Elements 41
weekday 251
widths parameter 102
WincgiTimeout
  magnus.conf directive 91
WincgiTimeout directive 116