Sun Java System Web Server 6.1
SP9 Administrator's
Configuration File Reference

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 820–5703
Contents

Preface ................................................................................................................................................... 15

1 Basics of Server Operation ................................................................................................................. 21
  Configuration Files ............................................................................................................................. 21
    server.xml ...................................................................................................................................... 22
    magnus.conf ................................................................................................................................. 22
    obj.conf ........................................................................................................................................ 22
    mime.types .................................................................................................................................... 23
  Other Configuration Files .............................................................................................................. 23
  Directory Structure ............................................................................................................................. 23
    All Platforms ................................................................................................................................. 23
    UNIX and Linux Platforms .......................................................................................................... 25
  Dynamic Reconfiguration .................................................................................................................. 26

2 Server Configuration Elements in server.xml ................................................................................. 27
  sun-web-server_6_1.dtd File ............................................................................................................. 27
  Subelements ..................................................................................................................................... 28
  Data ................................................................................................................................................ 28
  Attributes ....................................................................................................................................... 29
  Elements in the server.xml File ....................................................................................................... 29
  Core Server Elements ...................................................................................................................... 30
    SERVER ........................................................................................................................................ 30
    PROPERTY ................................................................................................................................. 31
    DESCRIPTION ............................................................................................................................. 32
    VARS ............................................................................................................................................ 32
  Listener Elements ............................................................................................................................ 33
    LS .................................................................................................................................................. 33
<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSLPARAMS</td>
<td>35</td>
</tr>
<tr>
<td>MIME</td>
<td>36</td>
</tr>
<tr>
<td>ACLFILE</td>
<td>37</td>
</tr>
<tr>
<td>VSCLASS</td>
<td>38</td>
</tr>
<tr>
<td>VS</td>
<td>39</td>
</tr>
<tr>
<td>QOSPARAMS</td>
<td>41</td>
</tr>
<tr>
<td>USERDB</td>
<td>42</td>
</tr>
<tr>
<td>WebDAV Elements</td>
<td>43</td>
</tr>
<tr>
<td>DAV</td>
<td>43</td>
</tr>
<tr>
<td>DAVCOLLECTION</td>
<td>45</td>
</tr>
<tr>
<td>Search Elements</td>
<td>46</td>
</tr>
<tr>
<td>SEARCH</td>
<td>47</td>
</tr>
<tr>
<td>SEARCHCOLLECTION</td>
<td>47</td>
</tr>
<tr>
<td>DISPLAYNAME</td>
<td>48</td>
</tr>
<tr>
<td>Web Application Elements</td>
<td>49</td>
</tr>
<tr>
<td>WEBAPP</td>
<td>49</td>
</tr>
<tr>
<td>Java Configuration Elements</td>
<td>50</td>
</tr>
<tr>
<td>JAVA</td>
<td>50</td>
</tr>
<tr>
<td>JVMOPTIONS</td>
<td>52</td>
</tr>
<tr>
<td>PROFILER</td>
<td>53</td>
</tr>
<tr>
<td>SECURITY</td>
<td>54</td>
</tr>
<tr>
<td>AUTHREALM</td>
<td>55</td>
</tr>
<tr>
<td>Resource Elements</td>
<td>56</td>
</tr>
<tr>
<td>RESOURCES</td>
<td>56</td>
</tr>
<tr>
<td>CUSTOMRESOURCE</td>
<td>57</td>
</tr>
<tr>
<td>EXTERNALJNDIRESOURCE</td>
<td>58</td>
</tr>
<tr>
<td>JDBCRESOURCE</td>
<td>59</td>
</tr>
<tr>
<td>JDBC CONNECTIONPOOL</td>
<td>59</td>
</tr>
<tr>
<td>CONNECTIONPROPERTY</td>
<td>63</td>
</tr>
<tr>
<td>MAILRESOURCE</td>
<td>64</td>
</tr>
<tr>
<td>LOG</td>
<td>65</td>
</tr>
<tr>
<td>Subelements</td>
<td>65</td>
</tr>
<tr>
<td>User Database Selection</td>
<td>66</td>
</tr>
<tr>
<td>Sun Java System LDAP Schema</td>
<td>67</td>
</tr>
<tr>
<td>Convergence Tree</td>
<td>68</td>
</tr>
<tr>
<td>Domain Component (dc) Tree</td>
<td>68</td>
</tr>
</tbody>
</table>

Sun Java System Web Server 6.1 SP9 Administrator’s Configuration File Reference
Variables ...............................................................................................................................................69
Format of a Variable ............................................................................................................................69
The id Variable .....................................................................................................................................69
Other Important Variables ..................................................................................................................70
Variable Evaluation .............................................................................................................................70
Sample server.xml File ..........................................................................................................................71

3 Syntax and Use of magnus.conf .........................................................................................................75
Init Functions .......................................................................................................................................76
Server Information ...............................................................................................................................76
ExtraPath ..............................................................................................................................................76
TempDir ..............................................................................................................................................76
TempDirSecurity .................................................................................................................................77
User ....................................................................................................................................................77
Language Issues ...................................................................................................................................78
DefaultLanguage .................................................................................................................................78
DNS Lookup .........................................................................................................................................78
AsyncDNS ..........................................................................................................................................78
DNS ....................................................................................................................................................79
Threads, Processes, and Connections .................................................................................................79
AcceptTimeout ......................................................................................................................................80
ConnQueueSize ....................................................................................................................................81
HeaderBufferSize .................................................................................................................................81
KeepAliveQueryMaxSleepTime .............................................................................................................81
KeepAliveQueryMeanTime .....................................................................................................................81
KeepAliveThreads ...............................................................................................................................82
KeepAliveTimeout ...............................................................................................................................82
KernelThreads ......................................................................................................................................82
ListenQ ................................................................................................................................................83
MaxKeepAliveConnections ..................................................................................................................83
MaxProcs (UNIX Only) .......................................................................................................................83
PostThreadsEarly ...............................................................................................................................83
RcvBufSize ..........................................................................................................................................84
RqThrottle ..........................................................................................................................................84
SndBufSize ..........................................................................................................................................84
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>StackSize</td>
<td>84</td>
</tr>
<tr>
<td>StrictHttpHeaders</td>
<td>85</td>
</tr>
<tr>
<td>TerminateTimeout</td>
<td>85</td>
</tr>
<tr>
<td>ThreadIncrement</td>
<td>85</td>
</tr>
<tr>
<td>UseNativePoll (UNIX only)</td>
<td>85</td>
</tr>
<tr>
<td>Native Thread Pools</td>
<td>86</td>
</tr>
<tr>
<td>NativePoolStackSize</td>
<td>86</td>
</tr>
<tr>
<td>NativePoolMaxThreads</td>
<td>86</td>
</tr>
<tr>
<td>NativePoolMinThreads</td>
<td>86</td>
</tr>
<tr>
<td>NativePoolQueueSize</td>
<td>87</td>
</tr>
<tr>
<td>CGI</td>
<td>87</td>
</tr>
<tr>
<td>CGIExpirationTimeout</td>
<td>87</td>
</tr>
<tr>
<td>CGIStubIdleTimeout</td>
<td>88</td>
</tr>
<tr>
<td>CGIWaitPid (UNIX Only)</td>
<td>88</td>
</tr>
<tr>
<td>MaxCGIStubs</td>
<td>88</td>
</tr>
<tr>
<td>MinCGIStubs</td>
<td>88</td>
</tr>
<tr>
<td>WincgiTimeout</td>
<td>88</td>
</tr>
<tr>
<td>Error Logging and Statistic Collection</td>
<td>89</td>
</tr>
<tr>
<td>ErrorLogDateFormat</td>
<td>89</td>
</tr>
<tr>
<td>LogFlushInterval</td>
<td>89</td>
</tr>
<tr>
<td>PidLog</td>
<td>90</td>
</tr>
<tr>
<td>ACL</td>
<td>90</td>
</tr>
<tr>
<td>ACLCacheLifetime</td>
<td>90</td>
</tr>
<tr>
<td>ACLUserCacheSize</td>
<td>91</td>
</tr>
<tr>
<td>ACLGroupCacheSize</td>
<td>91</td>
</tr>
<tr>
<td>Security</td>
<td>91</td>
</tr>
<tr>
<td>Security</td>
<td>91</td>
</tr>
<tr>
<td>ServerString</td>
<td>92</td>
</tr>
<tr>
<td>SSLCacheEntries</td>
<td>92</td>
</tr>
<tr>
<td>SSLClientAuthDataLimit</td>
<td>93</td>
</tr>
<tr>
<td>SSLClientAuthTimeout</td>
<td>93</td>
</tr>
<tr>
<td>SSLSessionTimeout</td>
<td>93</td>
</tr>
<tr>
<td>SSL3SessionTimeout</td>
<td>93</td>
</tr>
<tr>
<td>Chunked Encoding</td>
<td>94</td>
</tr>
<tr>
<td>UseOutputStreamSize</td>
<td>94</td>
</tr>
<tr>
<td>ChunkedRequestBufferSize</td>
<td>94</td>
</tr>
</tbody>
</table>
4 Predefined SAFs in obj.conf .................................................................113
   List of SAFs .................................................................................................................................114
   The bucket Parameter ...................................................................................................................115
   AuthTrans ......................................................................................................................................116
      basic-auth .................................................................................................................................117
      basic-ncsa .................................................................................................................................118
      get-sslid ....................................................................................................................................119
      match-browser ..........................................................................................................................120
      qos-handler ...............................................................................................................................121
      set-variable ..............................................................................................................................122
   NameTrans ....................................................................................................................................126
      assign-name ..............................................................................................................................126
      document-root ..........................................................................................................................128
      home-page ...............................................................................................................................129
      match-browser ..........................................................................................................................130
      ntrans-dav ................................................................................................................................130
      ntrans-j2ee ................................................................................................................................131
      pfx2dir ......................................................................................................................................131
      redirect ......................................................................................................................................133
      set-variable ..............................................................................................................................134
Contents

strip-params ................................................................. 134
unix-home ........................................................................... 135
PathCheck ............................................................................. 136
  check-acl ................................................................. 136
  find-compressed .......................................................... 137
  deny-existence ............................................................ 139
  find-index ...................................................................... 139
  find-links ....................................................................... 140
  find-pathinfo ............................................................... 141
  get-client-cert ............................................................ 142
  load-config .................................................................... 143
  match-browser ............................................................. 145
  nt-uri-clean ............................................................... 145
  ntcgicheck ..................................................................... 146
  pcheck-dav ..................................................................... 147
  require-auth ................................................................. 147
  set-variable .................................................................... 148
  set-virtual-index .......................................................... 148
  ssl-check ...................................................................... 149
  ssl-logout ...................................................................... 150
  unix-uri-clean ............................................................ 150
Parameters ........................................................................ 150
Example ............................................................................ 151
See Also ............................................................................ 151

ObjectType .......................................................................... 151
  force-type ...................................................................... 152
Parameters ........................................................................ 152
Example ............................................................................ 153
See Also ............................................................................ 153
  match-browser ............................................................. 153
set-default-type ............................................................... 153
Parameters ........................................................................ 153
Example ............................................................................ 154
set-variable ....................................................................... 154
shtml-hacktype ............................................................... 154
Parameters ........................................................................ 154
<table>
<thead>
<tr>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example .......................................................................................................................... 155</td>
</tr>
<tr>
<td>type-by-exp .................................................................................................................. 155</td>
</tr>
<tr>
<td>Parameters .................................................................................................................... 155</td>
</tr>
<tr>
<td>Example ....................................................................................................................... 156</td>
</tr>
<tr>
<td>See Also ....................................................................................................................... 156</td>
</tr>
<tr>
<td>type-by-extension ....................................................................................................... 156</td>
</tr>
</tbody>
</table>

**Input** ..................................................................................................................... 157

- insert-filter ............................................................................................................. 158
- match-browser ........................................................................................................ 159
- remove-filter .......................................................................................................... 159
- set-variable ............................................................................................................ 159

**Output** .................................................................................................................. 160

- insert-filter ............................................................................................................. 160
- match-browser ........................................................................................................ 161
- remove-filter .......................................................................................................... 161
- set-variable ............................................................................................................ 162

**Service** .................................................................................................................. 162

- type .......................................................................................................................... 162
- method ..................................................................................................................... 162
- query ........................................................................................................................ 162
- UseOutputStreamSize ............................................................................................. 163
- flushTimer ............................................................................................................... 163
- ChunkedRequestBufferSize .................................................................................... 163
- ChunkedRequestTimeout ......................................................................................... 163
- add-footer ............................................................................................................... 165
- add-header ............................................................................................................... 166
- append-trailer ......................................................................................................... 167
- imagemap .................................................................................................................. 169
- index-common ........................................................................................................ 169
- index-simple ............................................................................................................ 171
- key-too-small ......................................................................................................... 172
- list-dir ...................................................................................................................... 173
- make-dir .................................................................................................................. 175
- match-browser ........................................................................................................ 175
- query-handler ......................................................................................................... 176
- remove-dir ............................................................................................................... 176
5 MIME Types ...........................................................................................................................................205
  Introduction ........................................................................................................................................205
  Determining the MIME Type ..............................................................................................................206
  How the Type Affects the Response .................................................................................................206
  What Does the Client Do with the MIME Type? ..........................................................................207
  Syntax of the MIME Types File ........................................................................................................207
  Sample MIME Types File ..................................................................................................................207

6 Other Server Configuration Files .....................................................................................................209
  certmap.conf .........................................................................................................................................209
    Purpose ........................................................................................................................................209
    Location .......................................................................................................................................209
    Syntax .........................................................................................................................................210
    See Also ....................................................................................................................................210
  dbswitch.conf ......................................................................................................................................211
    Purpose ........................................................................................................................................211
    Location .......................................................................................................................................211
    Syntax .........................................................................................................................................211
    See Also ....................................................................................................................................212
  Deployment Descriptors .....................................................................................................................213
    Purpose ........................................................................................................................................213
    Location .......................................................................................................................................213
    See Also ....................................................................................................................................214
  generated.instance.acl .........................................................................................................................214
    Purpose ........................................................................................................................................214
    Location .......................................................................................................................................214
    See Also ....................................................................................................................................214
  login.conf ...........................................................................................................................................214
    Purpose ........................................................................................................................................214
    Location .......................................................................................................................................214
    See Also ....................................................................................................................................214
  nsfc.conf ...........................................................................................................................................215
    Purpose ........................................................................................................................................215
    Location .......................................................................................................................................215
    Syntax .........................................................................................................................................215
    See Also ....................................................................................................................................215
<table>
<thead>
<tr>
<th>File</th>
<th>Purpose</th>
<th>Location</th>
<th>Syntax</th>
<th>See Also</th>
</tr>
</thead>
<tbody>
<tr>
<td>password.conf</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>server.policy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*.clfilter</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### A Configuration Changes Between iPlanet Web Server 4.1 and Sun Java System Web Server 6.1

<table>
<thead>
<tr>
<th>File</th>
<th>Purpose</th>
<th>Location</th>
<th>Syntax</th>
<th>See Also</th>
</tr>
</thead>
<tbody>
<tr>
<td>magnus.conf</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>obj.conf</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>contexts.properties</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rules.properties</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>servlets.properties</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### B Configuration Changes Between iPlanet Web Server 6.0 and Sun Java System Web Server 6.1

<table>
<thead>
<tr>
<th>File</th>
<th>Purpose</th>
<th>Location</th>
<th>Syntax</th>
<th>See Also</th>
</tr>
</thead>
<tbody>
<tr>
<td>magnus.conf</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Init Functions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Directives</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>obj.conf</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>server.xml</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Preface

This guide discusses the purpose and use of the configuration files for Sun Java System 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” on page 15
- “Using the Documentation” on page 15
- “How This Guide Is Organized” on page 17
- “Documentation Conventions” on page 18
- “Product Support” on page 19

Who Should Use This Guide

The intended audience for this guide is one who administers and maintains the Sun Java System Web Server.

This guide assumes you are familiar with the following topics:

- J2SE specification
- HTTP
- HTML
- XML
- Java programming
- Java APIs as defined in servlet, JSP, and JDBC specifications
- Relational database concepts

Using the Documentation

The Sun Java System Web Server manuals are available as online files in PDF and HTML formats at http://docs.sun.com/app/docs/coll/1308.5.

The following table lists the tasks and concepts described in the Sun Java System Web Server manuals.
TABLE P–1  Sun Java System 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 Java System 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 Java System Web Server and its various components, supported platforms, and environments</td>
<td></td>
</tr>
<tr>
<td>■ Migrating from Sun Java System Web Server 4.1 or 6.0 to Sun Java System Web Server 6.1</td>
<td></td>
</tr>
<tr>
<td>Performing the following administration tasks:</td>
<td></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, Standard Edition (J2SE 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>
</tbody>
</table>
TABLE P-1  Sun Java System Web Server Documentation Roadmap  (Continued)

<table>
<thead>
<tr>
<th>For Information About</th>
<th>See the Following</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using programming technologies and APIs to do the following:</td>
<td></td>
</tr>
<tr>
<td>  Extend and modify Sun Java System Web Server</td>
<td>Programmer’s Guide</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>
<tr>
<td>Creating custom Netscape Server Application Programmer’s Interface (NSAPI) plug-ins</td>
<td>NSAPI Programmer’s Guide</td>
</tr>
<tr>
<td>Implementing servlets and JavaServer Pages™ (JSP) technology in Sun Java System Web Server</td>
<td>Programmers Guide to Web Applications</td>
</tr>
<tr>
<td>Editing configuration files</td>
<td>Administrator’s Configuration File Reference Guide</td>
</tr>
<tr>
<td>Tuning Sun Java System Web Server to optimize performance</td>
<td>Performance Tuning, Sizing, and Scaling Guide</td>
</tr>
</tbody>
</table>

How This Guide Is Organized

This guide has the following chapters:

■ Chapter 1, Basics of Server Operation
  Introduces the major configuration files that control the Sun Java System Web Server and describes how to activate and edit them.

■ Chapter 2, Server Configuration Elements in server.xml
  Describes the server.xml file, which controls most aspects of server operation.

■ Chapter 3, Syntax and Use of magnus.conf
  Describes the directives you can set in the magnus.conf file to configure the Sun Java System Web Server during initialization.

■ Chapter 4, Predefined SAFs in obj.conf
  describes the predefined SAFs used in the obj.conf file.

■ Chapter 5, MIME Types
  Describes the MIME types file, which maps file extensions to file types.

■ Chapter 6, Other Server Configuration Files
  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 Java System Web Server 6.1
  Describes the changes in configuration files between the 4.x and 6.1 versions of Sun Java System Web Server.
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.
  - *Italic* type is used for code variables.
  - *Italic* type is also used for book titles, emphasis, variables and placeholders, and words used in the literal sense.
  - **Bold** type is used as either a paragraph lead-in or to indicate words used in the literal sense.

- **Installation root directories** are indicated by `install_dir` in this document.

  By default, the location of `install_dir` on UNIX-based platforms is:

  `/opt/SUNWwbsvr/`
On Windows, it is:

C:\Sun\WebServer6.1

**Product Support**

If you have problems with your system, contact customer support using one of the following mechanisms:

- The online support website at:
- The telephone dispatch number associated with your maintenance contract

Please have the following information available prior to contacting support. This helps to ensure that our support staff can best assist you in resolving problems:

- Description of the problem, including the situation where the problem occurs and its impact on your operation
- Machine type, operating system version, and product version, including any patches and other software that might be affecting the problem
- Detailed steps on the methods you have used to reproduce the problem
- Any error logs or core dumps
The configuration and behavior of Sun Java™ System 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” on page 21
- “Directory Structure” on page 23
- “Dynamic Reconfiguration” on page 26

### Configuration Files

The configuration and operation of the Sun Java System 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” on page 22 -- contains most of the server configuration.
- “magnus.conf” on page 22 -- contains global server initialization information.
- “obj.conf” on page 22 -- contains instructions for handling HTTP requests from clients.
- “mime.types” on page 23 -- contains information for determining the content type of requested resources.
**server.xml**

server.xml 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 executes the settings in this file on startup. The server does not check 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 Java System Web Server about how to handle HTTP requests from clients and service web server content such as native server plug-ins and CGI programs. The server checks 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 Java System 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 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 Java System 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, 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 Java System Web Server 6.1. The information is organized in two parts:

- “All Platforms” on page 23
- “UNIX and Linux Platforms” on page 25

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 Java System 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 flexan1g 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 a 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 a 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.

- plug-ins contains directories for Java, search, and other plug-ins. This directory has the following subdirectories:
  - htaccess contains server plug-in for .htaccess access control and htcvert, an .nsconfig to .htaccess converter.
  - digest contains the Digest Authentication Plug-in for Sun Java System Directory Server 5.0, as well as information about the plug-in.
  - samples contains samples and example components, plug-ins and technologies supported by the Sun Java System Web Server servlet engine. This includes binaries, all code, and a build enviroment.
  - servlets contains information about and examples of web-apps applications.
  - include contains various include files.
  - 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 plug-in.
  - nsapi contains header files and example code for creating your own functions using NSAPI. For more information, see the Sun Java System documentation web site at: http://docs.sun.com/app/docs/coll/1308.5
  - search contains information for your server’s search plug-ins.
  - snmp contains information for your server’s SNMP plug-ins.

setup contains the various Sun Java System 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 Java System Web Server Release Notes.

UNIX and Linux Platforms

In addition to the files and directories described in “All Platforms” on page 23 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.
- `start` is the script that starts the Server Manager. The Server Manager lets you configure all servers installed in the server root directory.
- `stop` is the script that stops the Server Manager.

## Dynamic Reconfiguration

Dynamic reconfiguration enables 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 the `server.xml` file and its associated files without restarting the server.

To access the dynamic reconfiguration screen and install a new configuration dynamically, click the Apply link in the upper right corner of the Server Manager, Class Manager, and Virtual Server Manager pages, 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.
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:

- "sun-web-server_6_1.dtd File" on page 27
- "Elements in the server.xml File" on page 29
- "Core Server Elements" on page 30
- "Listener Elements" on page 33
- "WebDAV Elements" on page 43
- "Search Elements" on page 46
- "Web Application Elements" on page 49
- "Java Configuration Elements" on page 50
- "Resource Elements" on page 56
- "LOG" on page 65
- "User Database Selection" on page 66
- "Sun Java System LDAP Schema" on page 67
- "Variables" on page 69
- "Sample server.xml File" on page 71

**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 might be present in the corresponding XML file) can contain the following:

- "Subelements" on page 28
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:

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

For example:

```xml
<!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:
Attributes

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

```xml
<!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 Java System 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” on page 30
- “Listener Elements” on page 33
- “WebDAV Elements” on page 43
- “Search Elements” on page 46
- “Web Application Elements” on page 49
- “Java Configuration Elements” on page 50
- “Resource Elements” on page 56

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 “Appendix D, Alphabetical List of Server Configuration Elements.”
Core Server Elements

General elements are as follows:

- “SERVER” on page 30
- “PROPERTY” on page 31
- “DESCRIPTION” on page 32
- “VARS” on page 32

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>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“VARS” on page 32</td>
<td>zero or one</td>
<td>Defines variables that can be given values in server.xml and referenced in obj.conf</td>
</tr>
<tr>
<td>“PROPERTY” on page 31</td>
<td>zero or more</td>
<td>Specifies a property of the serve.</td>
</tr>
<tr>
<td>“LS” on page 33</td>
<td>one or more</td>
<td>Defines one or more HTTP listen sockets</td>
</tr>
<tr>
<td>“MIME” on page 36</td>
<td>zero or more</td>
<td>Defines the mime type</td>
</tr>
<tr>
<td>“ACLFILE” on page 37</td>
<td>zero or more</td>
<td>References one or more ACL files</td>
</tr>
<tr>
<td>“VSCLASS” on page 38</td>
<td>one or more</td>
<td>Defines a virtual server class</td>
</tr>
<tr>
<td>“QOSPARAMS” on page 41</td>
<td>zero or one</td>
<td>Defines quality of service parameters</td>
</tr>
<tr>
<td>“JAVA” on page 50</td>
<td>zero or one</td>
<td>Configures Java™ Virtual Machine (JVM) parameters</td>
</tr>
<tr>
<td>“LOG” on page 65</td>
<td>zero or one</td>
<td>Configures the system logging service</td>
</tr>
</tbody>
</table>

Attributes

The following table describes attributes for the SERVER element.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>qosactive</td>
<td>no</td>
<td>Enables quality of service features, which let you set limits on server entities or view server statistics for bandwidth and connections. Allowed values are yes, no, on, off, true, false, 1 or 0.</td>
</tr>
<tr>
<td>qosmetricsinterval</td>
<td>30</td>
<td>(optional) The interval in seconds during which the traffic is measured.</td>
</tr>
<tr>
<td>qosrecomputeinterval</td>
<td>100</td>
<td>(optional) The period in milliseconds in which the bandwidth gets recomputed for all server entities.</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" on page 69.

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 Java System Web Server
- Needed by a system or object that Sun Java System Web Server does not 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>
```

The properties that 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 2–4 PROPERTY Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;DESCRIPTION&quot; on page 32</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.

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 69.

Subelements
none

Attributes
none
Listener Elements

The listener elements are as follows:

- "LS" on page 33
- "SSLPARAMS" on page 35
- "MIME" on page 36
- "ACLFILE" on page 37
- "VSCLASS" on page 38
- "VS" on page 39
- "QOSPARAMS" on page 41
- "USERDB" on page 42

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 Java System 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>&quot;DESCRIPTION&quot; on page 32</td>
</tr>
<tr>
<td>&quot;SSLPARAMS&quot; on page 35</td>
</tr>
</tbody>
</table>

Attributes

The following table describes attributes for the LS element.
### 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. When you create a secure listen socket in the <code>server.xml</code> file, security must be turned on in <code>magnus.conf</code>. When you create a secure listen socket in the Server Manager, security is automatically turned on globally in <code>magnus.conf</code>.</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 <code>INADDR_ANY</code>.</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. The Security setting in the <code>magnus.conf</code> file globally enables or disables SSL by making certificates available to the server instance. Therefore, Security in <code>magnus.conf</code> must be on or security in <code>server.xml</code> does not work. For more information, see Chapter 3, Syntax and Use of <code>magnus.conf</code>.</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>
<tr>
<td>family</td>
<td>none</td>
<td>(Optional) The socket family type. Legal values are inet, inet6, and nca. Use the value inet6 for IPv6 listen sockets. When using the value of inet6, IPv4 addresses are prefixed with <code>::ffff:</code> in the log file. Specify nca to use the Solaris Network Cache and Accelerator.</td>
</tr>
</tbody>
</table>
### LS Attributes (Continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 does not 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>
</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>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 <code>tokenname:nickname</code>. Including the <code>tokenname</code> part of the name in this attribute is optional.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</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>
<tr>
<td>tlsrollback</td>
<td>true</td>
<td>(optional) Determines whether TLS rollback is enabled. Legal values are on, off, yes, no, 1, 0, true, and false. TLS rollback should be enabled for Microsoft Internet Explorer 5.0 and 5.5.</td>
</tr>
<tr>
<td>clientauth</td>
<td>false</td>
<td>(optional) Determines whether SSL3 client authentication is performed on every request, independent of ACL-based access control. Legal values are on, off, yes, no, 1, 0, true, and false.</td>
</tr>
</tbody>
</table>

**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" on page 30 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 &quot;VS&quot; on page 39 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 Chapter 5, MIME Types.</td>
</tr>
</tbody>
</table>

ACLFILE

References one or more ACL files

Subelements

The following table describes subelements for the ACLFILE element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;DESCRIPTION&quot; on page 32</td>
<td>Zero or one</td>
<td>Contains a text description of the ACLFILE element</td>
</tr>
</tbody>
</table>

Attributes

The following table describes attributes for the ACLFILE element.
TABLE 2–11  ACLFILE Attributes

<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 ACL file listing. Used in a &quot;VS&quot; on page 39 element to define the ACL file used by the virtual server. An ACL file listing name cannot begin with a number.</td>
</tr>
<tr>
<td>file</td>
<td>none</td>
<td>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 Java System Web Server 6.1 SP9 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.</td>
</tr>
</tbody>
</table>

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>&quot;DESCRIPTION&quot; on page 32</td>
<td>zero or one</td>
<td>Contains a text description of the VSCLASS.</td>
</tr>
<tr>
<td>&quot;VARS&quot; on page 32</td>
<td>zero or one</td>
<td>Specifies a property of the VSCLASS.</td>
</tr>
<tr>
<td>&quot;PROPERTY&quot; on page 31</td>
<td>zero or more</td>
<td>Specifies a property of the VSCLASS.</td>
</tr>
<tr>
<td>&quot;VS&quot; on page 39</td>
<td>zero or more</td>
<td>Defines a virtual server.</td>
</tr>
<tr>
<td>&quot;QOSPARAMS&quot; on page 41</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>
<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 does not report an error until a client tries to retrieve a document. The Server Manager assumes the default to be the object named default. Do not 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>

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 2–14  VS Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;DESCRIPTION&quot; on page 32</td>
<td>zero or one</td>
<td>Contains a text description of this element.</td>
</tr>
<tr>
<td>&quot;VARS&quot; on page 32</td>
<td>zero or one</td>
<td>Specifies a property or a variable of the VS.</td>
</tr>
<tr>
<td>&quot;PROPERTY&quot; on page 31</td>
<td>zero or more</td>
<td>Specifies a property or a variable of the VS.</td>
</tr>
<tr>
<td>&quot;QOSPARAMS&quot; on page 41</td>
<td>zero or one</td>
<td>Defines quality of service parameters.</td>
</tr>
<tr>
<td>&quot;USERDB&quot; on page 42</td>
<td>zero or more</td>
<td>Defines the user database for the virtual server.</td>
</tr>
<tr>
<td>&quot;DAV&quot; on page 43</td>
<td>zero or one</td>
<td>Defines the WebDAV configuration for the virtual server.</td>
</tr>
<tr>
<td>&quot;SEARCH&quot; on page 47</td>
<td>zero or one</td>
<td>Defines the search configuration for the virtual server.</td>
</tr>
<tr>
<td>&quot;WEBAPP&quot; on page 49</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 $id in an obj.conf 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>
<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>Attribute</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</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. 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” on page 36 element used by the virtual server.</td>
</tr>
<tr>
<td>aclids</td>
<td>none</td>
<td>(optional) One or more id attributes of “ACLFILE” on page 37 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” on page 65 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>
**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 “SERVER” on page 30, “VSCLASS” on page 38, or “VS” on page 39.</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” on page 39 element.

**Subelements**

The following table describes subelements for the USERDB element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;DESCRIPTION&quot; on page 32</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 USERDB element.
### WebDAV Elements

The WebDAV elements are as follows:

- “DAV” on page 43
- “DAVCOLLECTION” on page 45

#### DAV

Defines the WebDAV (Web-based Distributed Authoring and Versioning) configuration for the “VS” on page 39 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” on page 31</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 DAV element.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lockdb</td>
<td><code>server-instance\lock-db\vs</code></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>
<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>
</tbody>
</table>
TABLE 2–20  DAV Attributes  (Continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>

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 2–21  DAVCOLLECTION Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;DESCRIPTION&quot; on page 32</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 &quot;DAV&quot; on page 43 element</td>
<td>(optional) Specifies the directory where the locking database will be maintained.</td>
</tr>
</tbody>
</table>
### TABLE 2–22  DAVCOLLECTION Attributes  (Continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lockdbupdateinterval</td>
<td>The value specified in the “DAV” on page 43 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” on page 43 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” on page 43 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>
<tr>
<td>maxpropdbsize</td>
<td>The value specified in the “DAV” on page 43 element.</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>The value specified in the “DAV” on page 43 element.</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>The value specified in the “DAV” on page 43 element.</td>
<td>(optional) The maximum depth permitted for a DAV PROPFIND request. Allowed values are 0, 1, and infinity.</td>
</tr>
<tr>
<td>enabled</td>
<td>true</td>
<td>(optional) Specifies if DAV functionality is enabled for this collection.</td>
</tr>
</tbody>
</table>

### Search Elements

Search elements are as follows:

- “SEARCH” on page 47
- “SEARCHCOLLECTION” on page 47
- “DISPLAYNAME” on page 48
SEARCH

Defines search related configuration parameters for a given “VS” on page 39.

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” on page 49</td>
<td>zero or one</td>
</tr>
<tr>
<td>“SEARCHCOLLECTION” on page 47</td>
<td>zero or more</td>
</tr>
<tr>
<td>“PROPERTY” on page 31</td>
<td>zero or more</td>
</tr>
</tbody>
</table>

Attributes

The following table describes attributes for the SEARCH element.

<table>
<thead>
<tr>
<th>TABLE 2–24</th>
<th>SEARCH Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>Default</td>
</tr>
<tr>
<td>maxhits</td>
<td>none</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>TABLE 2–25</th>
<th>SEARCHCOLLECTION Subelements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Required</td>
</tr>
<tr>
<td>“DISPLAYNAME” on page 48</td>
<td>zero or one</td>
</tr>
</tbody>
</table>
Attributes

The following table describes attributes for the SEARCHCOLLECTION 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 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:

```xml
<DISPLAYNAME> Omega Manual </DISPLAYNAME>
```

Subelements

none

Attributes

none
Web Application Elements

The Web application elements are as follows:

- "WEBAPP" on page 49

WEBAPP

Defines a Java web application rooted at a given URI within a "VS" on page 39.

Subelements

The following table describes subelements for the WEBAPP element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;DESCRIPTION&quot; on page 32</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 WEBAPP element.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>empty string</td>
<td>This is the context path at which the web application is installed (Section 5.4 of the Servlet 2.3 specification). If this attribute is &quot;/&quot; then this web application is designated to be the default web application for the virtual server. The default web application for a virtual server responds to all requests that cannot be resolved to other web applications deployed to the virtual server. Every virtual server has a default web application.</td>
</tr>
<tr>
<td>path</td>
<td>none</td>
<td>A fully qualified or relative path to the directory in which the contents of the .war file have been extracted.</td>
</tr>
</tbody>
</table>
Java Configuration Elements

The Java configuration elements are as follows:

- “JAVA” on page 50
- “JVMOPTIONS” on page 52
- “PROFILER” on page 53
- “SECURITY” on page 54
- “AUTHREALM” on page 55

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>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;PROPERTY&quot; on page 31</td>
<td>zero or more</td>
<td>Specifies a property or a variable.</td>
</tr>
<tr>
<td>&quot;JVMOPTIONS&quot; on page 52</td>
<td>zero or more</td>
<td>Contains JVM command line options.</td>
</tr>
<tr>
<td>&quot;PROFILER&quot; on page 53</td>
<td>zero or one</td>
<td>Configures a profiler for use with the server.</td>
</tr>
<tr>
<td>&quot;SECURITY&quot; on page 54</td>
<td>zero or one</td>
<td>Defines parameters and configuration information needed by the security service.</td>
</tr>
<tr>
<td>&quot;RESOURCES&quot; on page 56</td>
<td>zero or one</td>
<td>Specifies configured resources.</td>
</tr>
</tbody>
</table>

Attributes

The following table describes attributes for the JAVA element.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enabled</td>
<td>true</td>
<td>This attribute can be used to temporarily disable the web application from servicing requests without removing the contents of the web application (on disk). Legal values are on, off, yes, no, 1, 0, true, false.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>javahome</td>
<td><code>&lt;install-root&gt;/bin/https/jdk</code></td>
<td>The path to the directory where the JDK is installed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For SVR 4 package-based installation for Solaris: <code>/usr/java</code></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><code>-Xdebug -Xrunjdwp:transport=dt_socket, server=y, suspend=n</code></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>
<tr>
<td>nativelibraryprefix</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 &quot;PROFILER&quot; on page 53 element. Since this is synthesized, it does not appear explicitly in the server configuration.</td>
</tr>
<tr>
<td>nativelibrarysuffix</td>
<td>none</td>
<td>(optional) Specifies a suffix for the native library path.</td>
</tr>
</tbody>
</table>
### TABLE 2-30 JAV A Attributes (Continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>envclasspathignored</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 classpath-suffix, 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>bytecodepreprocessors</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>dynamicreloadinterval</td>
<td>2</td>
<td>Specifies the interval, in seconds, after which a deployed application is reloaded.</td>
</tr>
<tr>
<td>loglevel</td>
<td>Value of level attribute of &quot;LOG&quot; on page 65 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 &quot;LOG&quot; on page 65 element.</td>
</tr>
</tbody>
</table>

#### JVMOPTIONS

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

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

In addition, web server looks for a system property, -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 http://java.sun.com/docs/hotspot/VMOptions.html.
Note – (On Windows only). Logging out of the machine where the web server is installed might abort the web server process. To avoid this problem, perform these steps:

1. Add the following line to the server.xml file of the web server instance.
   
   `<JVMOPTIONS>-Xrs</JVMOPTIONS>`

2. Restart the instance.

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>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;PROPERTY&quot; on page 31</td>
<td>zero or more</td>
<td>Specifies a property.</td>
</tr>
<tr>
<td>&quot;JVMOPTIONS&quot; on page 52</td>
<td>zero or more</td>
<td>Contains profiler-specific JVM command line options.</td>
</tr>
</tbody>
</table>

Attributes
The following table describes attributes for the 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>
</tbody>
</table>
## TABLE 2–32  PROFILER Attributes (Continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 SECURITY element.

### TABLE 2–33  SECURITY Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;PROPERTY&quot; on page 31</td>
<td>zero or more</td>
<td>Specifies a property or a variable.</td>
</tr>
<tr>
<td>&quot;AUTHREALM&quot; on page 55</td>
<td>one or more</td>
<td>Defines a realm for authentication.</td>
</tr>
</tbody>
</table>

### Attributes

The following table describes attributes for the SECURITY element.

### 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 &quot;AUTHREALM&quot; on page 55 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>
</tbody>
</table>
### ATTRIBUTE Default Description

- **audit**
  - **false**
  - (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:
    - Authentication success and failure events
    - Servlet access grants and denials

- **loglevel**
  - Value of `level` attribute of "LOG" on page 65 element
  - (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" on page 65 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:

```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 AUTHREALM element.
TABLE 2–35  AUTHREALM Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;PROPERTY&quot; on page 31</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 AUTHREALM element.

TABLE 2–36  AUTHREALM 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 this realm.</td>
</tr>
<tr>
<td>classname</td>
<td>none</td>
<td>Specifies the Java class that implements this realm.</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 Java System Web Server 6.1 Programmer’s Guide to Web Applications.

Resource Elements

Resource elements are as follows:

- "RESOURCES" on page 56
- "CUSTOMRESOURCE" on page 57
- "EXTERNALJNDIRESOURCE" on page 58
- "JDBCRESOURCE" on page 59
- "JDBCCONNECTIONPOOL" on page 59
- "CONNECTIONPROPERTY" on page 63
- "MAILRESOURCE" on page 64

**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>&quot;CUSTOMRESOURCE&quot; on page 57</td>
<td>zero or more</td>
<td>Defines a custom resource.</td>
</tr>
<tr>
<td>&quot;EXTERNALJNDIRESOURCE&quot; on page 58</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>&quot;JDBCRESOURCE&quot; on page 59</td>
<td>zero or more</td>
<td>Defines a JDBC (Java Database Connectivity) resource.</td>
</tr>
<tr>
<td>&quot;JDBCCONNECTIONPOOL&quot; on page 59</td>
<td>zero or more</td>
<td>Defines the properties that are required for creating a JDBC connection pool.</td>
</tr>
<tr>
<td>&quot;MAILRESOURCE&quot; on page 64</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>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;DESCRIPTION&quot; on page 32</td>
<td>zero or one</td>
<td>Contains a text description of this element.</td>
</tr>
<tr>
<td>&quot;PROPERTY&quot; on page 31</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 CUSTOMRESOURCE element.
### CUSTOMRESOURCE Attributes

<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>restype</td>
<td>none</td>
<td>Specifies the fully qualified type of the resource.</td>
</tr>
<tr>
<td>factoryclass</td>
<td>none</td>
<td>Specifies the fully qualified name of the user-written factory class, which implements javax.naming.spi.ObjectFactory.</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>

### 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>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;DESCRIPTION&quot; on page 32</td>
<td>zero or one</td>
<td>Contains a text description of this element.</td>
</tr>
<tr>
<td>&quot;PROPERTY&quot; on page 31</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 EXTERNALJNDIRESOURCE 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>jndilookupname</td>
<td>none</td>
<td>Specifies the JNDI lookup name for the resource.</td>
</tr>
<tr>
<td>restype</td>
<td>none</td>
<td>Specifies the fully qualified type of the resource.</td>
</tr>
</tbody>
</table>
### TABLE 2–41  EXTERNALJNDIRESOURCE Attributes  (Continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>factoryclass</td>
<td>none</td>
<td>Specifies the fully qualified name of the factory class, which implements <code>javax.naming.spi.InitialContextFactory</code>.</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>

### JDBCRESOURCE

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

#### Subelements

The following table describes subelements for the JDBCRESOURCE element.

#### TABLE 2–42  JDBCRESOURCE Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;DESCRIPTION&quot; on page 32</td>
<td>zero or one</td>
<td>Contains a text description of this element.</td>
</tr>
<tr>
<td>&quot;PROPERTY&quot; on page 31</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 2–43  JDBCRESOURCE Attributes

<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 &quot;JDBCCONNECTIONPOOL&quot; on page 59 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 JDBC.getConnectionPool element is reserved and ignored in Sun Java System Web Server 6.1. Any value set for this attribute is ignored by the server.

## Subelements

The following table describes subelements for the JDBC.getConnectionPool element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;DESCRIPTION&quot; on page 32</td>
<td>zero or one</td>
<td>Contains a text description of this element.</td>
</tr>
<tr>
<td>&quot;PROPERTY&quot; on page 31</td>
<td>zero or more</td>
<td>Specifies a property or a variable.</td>
</tr>
<tr>
<td>&quot;CONNECTIONPROPERTY&quot; on page 63</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 JDBC.getConnectionPool 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 &quot;JDBCRESOURCE&quot; on page 59 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.XDataSource 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>
<tr>
<td>Attribute</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<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 Java System 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>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>Specifies the user name for this connection pool.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the password for this connection pool.</td>
</tr>
<tr>
<td>databaseName</td>
<td>Specifies the database for this connection pool.</td>
</tr>
</tbody>
</table>
### TABLE 2-46  JDBC CONNECTION POOL Properties  (Continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverName</td>
<td>Specifies the database server for this connection pool.</td>
</tr>
<tr>
<td>port</td>
<td>Specifies the port on which the database server listens for requests.</td>
</tr>
<tr>
<td>networkProtocol</td>
<td>Specifies the communication protocol.</td>
</tr>
<tr>
<td>roleName</td>
<td>Specifies the initial SQL role name.</td>
</tr>
<tr>
<td>dataSourceName</td>
<td>Specifies an underlying XADataSource, or a ConnectionPoolDataSource if connection pooling is done.</td>
</tr>
<tr>
<td>description</td>
<td>Specifies a text description.</td>
</tr>
<tr>
<td>url</td>
<td>Specifies the URL for this connection pool. Although this is not a standard property, it is commonly used.</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 2-47  CONNECTIONPROPERTY Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“DESCRIPTION” on page 32</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.

#### TABLE 2-48  CONNECTIONPROPERTY 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 a name for the connection property.</td>
</tr>
<tr>
<td>value</td>
<td>none</td>
<td>Specifies a value for the connection property.</td>
</tr>
<tr>
<td>invocationfrequency</td>
<td>at-creation</td>
<td>(optional) Specifies the frequency with which the connection property is invoked. Legal values are at-creation and every-lease.</td>
</tr>
</tbody>
</table>
MAILRESOURCE

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

Subelements

The following table describes subelements for the MAILRESOURCE element.

**TABLE 2-49 MAILRESOURCE Subelements**

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“DESCRIPTION” on page 32</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 2-50 MAILRESOURCE Attributes**

<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>
<tr>
<td>storeprotocolclass</td>
<td>com.sun.mail.imap.IMAP Store</td>
<td>(optional) Specifies the service provider implementation class for storage. You can find this class at:</td>
</tr>
<tr>
<td>transportprotocol</td>
<td>smtp</td>
<td>(optional) Specifies the transport protocol service, which sends messages.</td>
</tr>
<tr>
<td>transportprotocolclass</td>
<td>com.sun.mail.smtp.SMTP Transport</td>
<td>(optional) Specifies the service provider implementation class for transport. You can find this class at:</td>
</tr>
<tr>
<td>host</td>
<td>none</td>
<td>The mail server host name</td>
</tr>
</tbody>
</table>
LOG

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 log file attribute is explicitly specified in the “VSCLASS” on page 38 or “VS” on page 39 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" on page 38 or "VS" on page 39 element that has an explicitly specified log-file attribute.

Subelements

The following table describes subelements for the LOG element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“PROPERTY” on page 31</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.
### User Database Selection

A "USERDB" on page 42 object selects a user database for the parent virtual server. This selection occurs in the following manner:

- The "USERDB" on page 42 element’s id attribute maps to an ACL file’s database attribute.
- The "USERDB" on page 42 element’s database attribute maps to a dbswitch.conf entry.

#### 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 error log attribute is explicitly specified in the &quot;VS&quot; on page 39 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 &quot;VS&quot; on page 39 elements share the same log file. Legal values are on, off, yes, no, 1, 0, true, false.</td>
</tr>
<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>
This layer between the ACL file and the `dbswitch.conf` file gives the server administrator full control over the databases to which virtual server administrators and users have access. 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" on page 42 tag.

- 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. If the database is schema compliant you can use dcsuffix. Requirements for schema compliance are listed in “Sun Java System LDAP Schema” on page 67.

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

- If a “VS” on page 39 has no “USERDB” on page 42 subelement, user- or group-based ACLs fail.

- When no database attribute is present in a virtual server’s ACL definition, the “VS” on page 39 must have a “USERDB” on page 42 subelement with an id attribute of `default`. The database attribute of the “USERDB” on page 42 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” on page 30 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 basedn attribute of the “USERDB” on page 42 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 Java System Web Server versions.

**Sun Java System LDAP Schema**

This section describes the Sun Java System 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 211.

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, o=root may have o=isp and o=internet under it), or have two separate roots, one for the convergence tree and one for the dc tree.

**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 organizationalUnit entries: ou=People and ou=Groups. A third, ou=Devices, can be present if device data is to be stored for the organization.

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

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

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

Groups are located under the Groups subtree and consist of LDAP entries of type groupOfUniqueNames.

**Domain Component (dc) Tree**

The dc tree contains hierarchical 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 inetDomain. For example, the two LDAP entries
dc=customer1,dc=com,o=Internet,o=root and dc=customer2,dc=com,o=Internet,o=root
contain the inetDomain class, but 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 contains the attribute inetDomainBaseDN with value o=Cust2,o=isp,o=root.

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

Variables

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

```
<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:

```
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.

Format of a Variable

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.

The id Variable

A special variable, id, is always available within a "VS" on page 39 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:

```
<PROPERTY name=docroot value="/export/$id" >
```

If the id attribute of the parent "VS" on page 39 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>docroot</td>
<td>The document root of the virtual server. Typically evaluated as the parameter to the document-root function in the obj.conf file.</td>
</tr>
<tr>
<td>accesslog</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>user</td>
<td>The value of the user CGI parameter.</td>
</tr>
<tr>
<td>group</td>
<td>The value of the group CGI parameter.</td>
</tr>
<tr>
<td>chroot</td>
<td>The value of the chroot CGI parameter.</td>
</tr>
<tr>
<td>dir</td>
<td>The value of the dir CGI parameter.</td>
</tr>
<tr>
<td>nice</td>
<td>The value of the nice CGI parameter.</td>
</tr>
</tbody>
</table>

For more information about the send-cgi function, see the Sun Java System Web Server 6.1 SP9 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:
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.
  -->
<!DOCTYPE SERVER PUBLIC "-//Sun Microsystems Inc.//DTD Sun Java System 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"/>
```

Chapter 2 • Server Configuration Elements in server.xml 71
defaultvs="useradmin">

<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>
    </WEBAPP>
</USERDB>

<PROPERTY name="docroot" value="/home/nb136819/space/servers/s1ws61/docs">
    <USERDB id="default">
        <WEBAPP uri="/user-app" path="/home/nb136819/space/servers/s1ws61/bin/https/webapps/user-app">
        </VS>
    </WEBAPP>
</USERDB>

</VSCLASS>

</JAVA>

</JVMOPTIONS>

<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>-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/s1ws61/https-admserv/config/server.policy</JVMOPTIONS>

<JVMOPTIONS>-Djava.security.auth.login.config=/home/nb136819/space/servers/s1ws61/https-admserv/config/login.conf</JVMOPTIONS>

<JVMOPTIONS>-Djava.util.logging.manager=com.iplanet.ias.server.logging.ServerLogManager</JVMOPTIONS>

<JVMOPTIONS>-Xms128m -Xmx256m</JVMOPTIONS>

<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/keyfile" />
    <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>
Syntax and Use of magnus.conf

When the Sun Java™ System Web Server starts up, it looks in a file called magnus.conf in the server-id/conf directory to establish a set of global variable settings that affect the server’s behavior and configuration. Sun Java System 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 Java System Web Server 6.1.

The categories are:

- “Init Functions” on page 76
- “Server Information” on page 76
- “Language Issues” on page 78
- “DNS Lookup” on page 78
- “Threads, Processes, and Connections” on page 79
- “Native Thread Pools” on page 86
- “CGI” on page 87
- “Error Logging and Statistic Collection” on page 89
- “ACL” on page 90
- “Security” on page 91
- “Chunked Encoding” on page 94
- “Miscellaneous Directives” on page 95
Init Functions

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 Java System Web Server 6.1, see " Deprecated Directives" on page 96.

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 215.

Init Functions

The Init functions load and initialize server modules and plug-ins, and initialize log files. For more information about these functions, see the Sun Java System Web Server 6.1 SP9 NSAPI Programmer’s Guide.

Server Information

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

- “ExtraPath” on page 76
- “TempDir” on page 76
- “TempDirSecurity” on page 77
- “User” on page 77

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” on page 77 and “TempDirSecurity” on page 77.

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 Unix user account.
**Language Issues**

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

- **DefaultLanguage** on page 78

**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” on page 78
- “DNS” on page 79

**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, JSP, Servlet, 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.

Threads, Processes, and Connections

In Sun Java System 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” on page 80
- “ConnQueueSize” on page 81
- “HeaderBufferSize” on page 81
- “KeepAliveQueryMaxSleepTime” on page 81
- “KeepAliveQueryMeanTime” on page 81
- “KeepAliveThreads” on page 82
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 Java System Web Server 6.1 SP9 Performance Tuning, Sizing, and Scaling Guide*.

**Default**
The default value is 4096.

**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" on page 84 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” on page 83) 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 Java System 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 (Windows), 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 1048576 for the 64-bit server.

**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 Java System 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 Java System Web Server 6.1 *Performance Tuning, Sizing, and Scaling Guide*.

**Default**

1

**PostThreadsEarly**

If this directive is set to 1 (on), the server checks 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 Java System Web Server 6.1 SP9 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).

**StackSize**

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**

off

**TerminateTimeout**

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

**Default**

30 seconds

**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 Java System 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" on page 86
- "NativePoolMaxThreads" on page 86
- "NativePoolMinThreads" on page 86
- "NativePoolQueueSize" on page 87

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.

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

**CGI**

This section lists the directives in `magnus.conf` that affect requests for CGI programs. The directives are:

- “CGIExpirationTimeout” on page 87
- “CGIStubIdleTimeout” on page 88
- “CGIWaitPid (UNIX Only)” on page 88
- “MaxCGIStubs” on page 88
- “MinCGIStubs” on page 88

**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 plug-in 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” on page 89
- “LogFlushInterval” on page 89
- “PidLog” on page 90

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

2
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

PidLog /tmp/ns-server.pid

ACL

This section lists the directives in magnus . conf relevant to access control lists (ACLs). They are:

- “ACLCacheLifetime” on page 90
- “ACLUserCacheSize” on page 91
- “ACLGroupCacheSize” on page 91

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 Java System Web Server when you make changes to the LDAP entries. For example, if this value is set to 120 seconds, the Sun Java System 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 Java System Web Server. They are:

- “Security” on page 91
- “ServerString” on page 92
- “SSLCacheEntries” on page 92
- “SSLClientAuthDataLimit” on page 93
- “SSLClientAuthTimeout” on page 93
- “SSLSessionTimeout” on page 93
- “SSL3SessionTimeout” on page 93

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 Java System Web Server 6.1 SP9 NSAPI Programmer's Guide*.

- “UseOutputStreamSize” on page 94
- “ChunkedRequestBufferSize” on page 94
- “ChunkedRequestTimeout” on page 95

These directives have equivalent Service SAF parameters in `obj.conf`. The `obj.conf` parameters override these directives. For more information, see the *Sun Java System Web Server 6.1 SP9 NSAPI Programmer's Guide*.

**UseOutputStreamSize**

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

**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” on page 95
- “Favicon” on page 96
- “HTTPVersion” on page 96
- “MaxRqHeaders” on page 96
- “Umask (UNIX only)” on page 96

Note – Directives noted with boolean values have the following equivalent values: on/yes/true and off/no/false.

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 directive
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.

For more information on these functions, see the Sun Java System Web Server 6.1 SP9 NSAPI Programmer’s Guide.

Deprecated Directives

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

- AdminLanguage
- ClientLanguage
- NetsiteRoot
- ServerID
- ServerName
- ServerRoot
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 “Init Functions” on page 97, functions are in bold to distinguish them from parameters.

Directive value

See Also
Sun Java System Web Server 6.1 SP9 NSAPI Programmer’s Guide

Init Functions
The following table lists the Init functions available in the magnus.conf file:
### 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>cindex-init</td>
<td></td>
<td></td>
<td>Changes the default characteristics for fancy indexing.</td>
</tr>
</tbody>
</table>
| opts               | s              | (None)        | (optional) is a string of letters specifying the options to activate. Currently there is only one possible option:  
  - s tells the server to scan each HTML file in the directory being indexed for the contents of the HTML <TITLE> tag to display in the description field. The <TITLE> tag must be within the first 255 characters of the file. |
| widths             | Comma separated numbers of characters | Minimums required to display column titles | (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. |
| timezone           | GMT or local   | local         | (optional, iPlanet Web Server 4.x only) Indicates whether the last-modified time is shown in local time or in Greenwich Mean Time. |
| format             | Format for the UNIX function strftime() | %d-%b-%Y %H:%M | (optional, iPlanet Web Server 4.x only) Determines the format of the last modified date display. |
| ignore             | Wildcard pattern | .*     | (optional) Specifies a wildcard pattern for file names the server should ignore while indexing. File names starting with a period (.) are always ignored. |
### TABLE 3–1  magnus.conf Init Functions  
(Continued)

<table>
<thead>
<tr>
<th>Function/Parameter</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>icon-uri</td>
<td>/mc-icons/</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 prefix2dir function in the NameTrans directive must be changed so that the server can find these icons.</td>
<td></td>
</tr>
<tr>
<td>define-perf-bucket</td>
<td></td>
<td></td>
<td>Creates a performance bucket, which you can use to measure the performance of SAFs in obj.conf (see the Sun Java System Web Server 6.1 SP9 NSAPI Programmer’s Guide). This function works only if the perf-init function is enabled.</td>
</tr>
<tr>
<td>name</td>
<td></td>
<td></td>
<td>A name for the bucket, for example cgi-bucket.</td>
</tr>
<tr>
<td>description</td>
<td></td>
<td></td>
<td>A description of what the bucket measures, for example CGI Stats.</td>
</tr>
<tr>
<td>dns-cache-init</td>
<td></td>
<td></td>
<td>Configures DNS caching.</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)</td>
<td>(optional) specifies how long (in seconds) it takes for a cache entry to expire.</td>
</tr>
<tr>
<td>flex-init</td>
<td></td>
<td></td>
<td>Initializes the flexible logging system.</td>
</tr>
<tr>
<td>logFileName</td>
<td>A path or file name</td>
<td></td>
<td>The full path to the log file or a file name relative to the server’s logs directory. In this example, the log file name is access and the path is /logdir/access: access=&quot;/logdir/access&quot;</td>
</tr>
</tbody>
</table>
### TABLE 3–1 magnus.conf Init Functions (Continued)

<table>
<thead>
<tr>
<th>Function/Parameter</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>format.logFileName</code></td>
<td></td>
<td></td>
<td>Specifies the format of each log entry in the log file. See the Sun Java System Web Server 6.1 SP9 NSAPI Programmer's Guide for more information.</td>
</tr>
<tr>
<td><code>buffer-size</code></td>
<td>Number of bytes</td>
<td>8192</td>
<td>Specifies the size of the global log buffer.</td>
</tr>
<tr>
<td><code>buffers-per-file</code></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><code>num-buffers</code></td>
<td></td>
<td>1000</td>
<td>Specifies the maximum number of logging buffers to use.</td>
</tr>
<tr>
<td><code>flex-rotate-init</code></td>
<td></td>
<td></td>
<td>Enables rotation for logs.</td>
</tr>
<tr>
<td><code>rotate-start</code></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><code>rotate-interval</code></td>
<td>Number of minutes</td>
<td></td>
<td>Indicates the number of minutes to elapse between each log rotation.</td>
</tr>
<tr>
<td><code>rotate-access</code></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 Java System Web Server 6.1 SP9 NSAPI Programmer's Guide.</td>
</tr>
<tr>
<td><code>rotate-error</code></td>
<td>yes, no</td>
<td>yes</td>
<td>(optional) determines whether error logs are rotated.</td>
</tr>
<tr>
<td>Function/Parameter</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>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 Java System Web Server 6.1 SP9 NSAPI Programmer’s Guide.</td>
</tr>
<tr>
<td>env-variable</td>
<td></td>
<td></td>
<td>(optional) specifies the name and value for an environment variable that the server places into the environment for the CGI.</td>
</tr>
<tr>
<td>init-clf</td>
<td></td>
<td></td>
<td>Initializes the Common Log subsystem.</td>
</tr>
<tr>
<td>logFileName</td>
<td>A path or file name</td>
<td></td>
<td>Specifies either the full path to the log file or a file name relative to the server’s logs directory.</td>
</tr>
<tr>
<td>init-uhome</td>
<td></td>
<td></td>
<td>Loads user home directory information.</td>
</tr>
<tr>
<td>Function/Parameter</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>pwfile</td>
<td></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>
</tr>
<tr>
<td>load-modules</td>
<td></td>
<td>Loads shared libraries into the server.</td>
<td></td>
</tr>
<tr>
<td>shlib</td>
<td></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>
</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>
<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>The name of a custom thread pool, as specified in thread-pool-init.</td>
<td></td>
</tr>
<tr>
<td>nt-console-init</td>
<td></td>
<td>Enables the NT console, which is the command-line shell that displays standard output and error streams.</td>
<td></td>
</tr>
<tr>
<td>stderr</td>
<td>console</td>
<td>Directs error messages to the NT console.</td>
<td></td>
</tr>
<tr>
<td>stdout</td>
<td>console</td>
<td>Directs output to the NT console.</td>
<td></td>
</tr>
<tr>
<td>perf-init</td>
<td></td>
<td>Enables system performance measurement via performance buckets.</td>
<td></td>
</tr>
<tr>
<td>Function/Parameter</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>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>block-size</td>
<td></td>
<td></td>
<td>(optional) size (in bytes) of the memory blocks allocated by the internal pooled memory allocator. Default value is 32768.</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>
<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>
<tr>
<td>thread-pool-init</td>
<td></td>
<td></td>
<td>Configures an additional thread pool.</td>
</tr>
<tr>
<td>name</td>
<td></td>
<td></td>
<td>Name of the thread pool.</td>
</tr>
</tbody>
</table>
TABLE 3–1  magnus.conf Init Functions (Continued)

<table>
<thead>
<tr>
<th>Function/Parameter</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxthreads</td>
<td></td>
<td></td>
<td>Maximum number of threads in the pool. You can use only five threads at a time in the FastTrack edition of iPlanet Web Server.</td>
</tr>
<tr>
<td>minthreads</td>
<td></td>
<td></td>
<td>Minimum number of threads in the pool.</td>
</tr>
<tr>
<td>queueSize</td>
<td>Number of bytes</td>
<td></td>
<td>Size of the queue for the pool.</td>
</tr>
<tr>
<td>stackSize</td>
<td>Number of bytes</td>
<td></td>
<td>Stack size of each thread in the native (kernel) thread pool.</td>
</tr>
</tbody>
</table>

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</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>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>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>CGISTubIdleTimeout</td>
<td>Any number of seconds</td>
<td>30</td>
<td>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.</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>
<tr>
<td>ChildRestartCallback</td>
<td>on, off, yes, no, true, false</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>ChunkedRequestBufferSize</td>
<td>Any number of bytes</td>
<td>8192</td>
<td>Determines the default buffer size for &quot;un-chunking&quot; 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 &quot;un-chunking&quot; request data.</td>
</tr>
<tr>
<td>ConnQueueSize</td>
<td>Any number of connections (including 0)</td>
<td>4096</td>
<td>Specifies the number of outstanding (yet to be serviced) connections that the web server can have. This setting can have performance implications. For more information, see the Sun Java System Web Server 6.1 SP9 Performance Tuning, Sizing, and Scaling Guide.</td>
</tr>
<tr>
<td>DefaultLanguage</td>
<td>en (English), fr (French), de (German), ja (Japanese)</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>
</tbody>
</table>
### TABLE 3-2 magnus.conf Directives (Continued)

<table>
<thead>
<tr>
<th>Directive</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 for the C library function strftime</td>
<td>%d/%b/%Y:%H:%M:%S</td>
<td>The date format for the error log.</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>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>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>KeepAliveQueryMaxSleepTime</td>
<td>100</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>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This directive specifies an upper limit to the time slept (in milliseconds) after polling keep-alive connections for further requests.</td>
</tr>
<tr>
<td>KeepAliveQueryMeanTime</td>
<td>100 is appropriate for almost all installations. Note that CPU usage will increase with lower KeepAliveQueryMeanTime values.</td>
<td></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>
</tbody>
</table>
### Summary of Init Functions and Directives in magnus.conf

<table>
<thead>
<tr>
<th>Directive</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 CGIStub processes the server can spawn. This is the maximum concurrent CGIStub processes in execution, not the maximum number of pending requests.</td>
</tr>
<tr>
<td>MaxKeepAliveConnections</td>
<td>0 - 1048576 (for 64-bit server)</td>
<td>25616</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>Any number of processes</td>
<td>1</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>1285</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>NativeQueueSize</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>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>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>
<tr>
<td>RqThrottle</td>
<td>Any number of requests (including 0)</td>
<td>51216</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 Java System Web Server 6.1 SP9 Performance Tuning, Sizing, and Scaling Guide.</td>
</tr>
<tr>
<td>RqThrottleMin</td>
<td>Any number less than RqThrottle</td>
<td>484</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</td>
<td>86400 (24 hours)</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>
</tbody>
</table>
### TABLE 3-2 magnus.conf Directives (Continued)

<table>
<thead>
<tr>
<th>Directive</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
<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></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></td>
<td></td>
<td>TEMP (environment variable for Windows NT)</td>
<td></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>
<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>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>
Predefined SAFs in obj.conf

The obj.conf configuration file contains directives that instruct the Sun Java System Web Server how to handle HTTP and HTTPS requests from clients and service web server content such as native server plug-ins 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 Java System 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 Java System 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:
For an alphabetical list of predefined SAFs, see Appendix E, Alphabetical List of Predefined SAFs

### List of SAFs

The following table lists the Server Application Functions (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” on page 116</td>
<td>“basic-auth” on page 117 “basic-ncsa” on page 118”get-sslid” on page 119 “match-browser” on page 120”qos-handler” on page 121”set-variable” on page 122</td>
</tr>
<tr>
<td>“NameTrans” on page 126</td>
<td>“assign-name” on page 126”document-root” on page 128”home-page” on page 129”match-browser” on page 130”nttrans-dav” on page 131”pfx2dir” on page 131”redirect” on page 133”set-variable” on page 132”strip-params” on page 134”unix-home” on page 135</td>
</tr>
<tr>
<td>“PathCheck” on page 136</td>
<td>“check-acl” on page 136”deny-existence” on page 139”find-index” on page 139”find-links” on page 140”find-pathinfo” on page 141”get-client-cert” on page 142”load-config” on page 143”match-browser” on page 142”nt-uri-clean” on page 145”ntcgicheck” on page 146”require-auth” on page 147”set-variable” on page 148”set-virtual-index” on page 149”ssl-check” on page 150”ssl-password” on page 150</td>
</tr>
<tr>
<td>“ObjectType” on page 151</td>
<td>“force-type” on page 152”match-browser” on page 152”set-default-type” on page 153”set-variable” on page 154”set-virtual-index” on page 155”type-by-extension” on page 156</td>
</tr>
</tbody>
</table>
TABLE 4–1  Available SAFs per Directive (Continued)

<table>
<thead>
<tr>
<th>Directive</th>
<th>Server Application Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Input” on page 157</td>
<td>“insert-filter” on page 158 “match-browser” on page 120 “remove-filter” on page 159 “set-variable” on page 122</td>
</tr>
<tr>
<td>“Output” on page 160</td>
<td>“insert-filter” on page 160 “match-browser” on page 120 “remove-filter” on page 161 “set-variable” on page 122</td>
</tr>
<tr>
<td>“AddLog” on page 195</td>
<td>“common-log” on page 195 “flex-log” on page 196 “match-browser” on page 120 “record-useragent” on page 197 “set-variable” on page 122</td>
</tr>
<tr>
<td>“Error” on page 198</td>
<td>“error-j2ee” on page 199 “match-browser” on page 120 “qos-error” on page 200 “query-handler” on page 200 “remove-filter” on page 201 “send-error” on page 202 “set-variable” on page 122</td>
</tr>
</tbody>
</table>

The bucket Parameter

The following performance buckets are predefined in Sun Java System 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” on page 188 Service function.

As an alternative, you can use the “stats-xml” on page 192 Service function to generate performance statistics; use of buckets is optional.

For more information about performance buckets, see the Sun Java System Web Server 6.1 SP9 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 username 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” on page 116 validates authorization information sent by the client in the Authorization header.
- “PathCheck” on page 136 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” on page 117 calls a custom function to verify user name and password. Optionally determines the user’s group.
- “basic-ncsa” on page 118 verifies user name and password against an NCSA-style or system DBM database. Optionally determines the user’s group.
- “get-sslid” on page 119 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” on page 120 matches specific strings in the User-Agent string supplied by the browser, and then modifies the behavior of Sun Java System Web Server based upon the results by setting values for specified variables.
- “qos-handler” on page 121 handles the current quality of service statistics.
- “set-variable” on page 122 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” on page 147.

**Parameters**

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

<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 <code>load-modules</code>. 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 <code>load-modules</code>. 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:

Init fn=load-modules shlib=/path/to/mycustomauth.so
  funcs=hardcoded_auth

In obj.conf:

AuthTrans fn=basic-auth auth-type=basic
  userfn=hardcoded_authPathCheck fn=require-auth
  auth-type=basic realm="Marketing Plans"

See Also
“require-auth” on page 147

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"
on page 147.

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>auth-type</td>
<td>Specifies the type of authorization to be used. This should always be basic.</td>
</tr>
<tr>
<td>dbm</td>
<td>(Optional) Specifies the full path and base file name of the user database in</td>
</tr>
<tr>
<td></td>
<td>the server’s native format. The native format is a system DBM file, which is a</td>
</tr>
<tr>
<td></td>
<td>hashed file format allowing instantaneous access to billions of users. If you</td>
</tr>
<tr>
<td></td>
<td>use this parameter, don’t use the userfile parameter as well.</td>
</tr>
<tr>
<td>userfile</td>
<td>(Optional) Specifies the full path name of the user database in the NCSA-style</td>
</tr>
<tr>
<td></td>
<td>HTTPD user file format. This format consists of lines using the format name:password, where password is encrypted. If you use this parameter, don’t use dbm.</td>
</tr>
</tbody>
</table>
TABLE 4–3  basic-auth Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grpfile</td>
<td>(Optional) Specifies the NCSA-style HTTPD group file to be used. Each line of a group file consists of group: user1 user2 ... userN where each user is separated by spaces.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</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” on page 147

**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 4-4  get-sslid 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>

**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 Java System Web Server based upon the results by setting values for specified variables.

**Syntax**

\[
\text{stage fn="match-browser" browser="string" name="value" [name="value" ...]}\]

**Parameters**

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

TABLE 4-5  match-browser Parameter Values

<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 &quot;set-variable&quot; on page 122 SAF. For a list of valid variables, see &quot;set-variable&quot; on page 122.</td>
</tr>
<tr>
<td>value</td>
<td>New value for the specified variable.</td>
</tr>
</tbody>
</table>

**Example**

The following AuthTrans directive instructs Sun Java System 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 "set-variable" on page 122).
- Not honor requests for HTTP Keep-Alive (see "set-variable" on page 122)
- Use the HTTP/1.0 protocol rather than HTTP/1.1 (see "set-variable" on page 122).
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 Java System Web Server 6.1 NSAPI Programmer’s Guide.

For more information, see the Sun Java System Web Server 6.1 SP9 Performance Tuning, Sizing, and Scaling Guide.

### Parameters

The following table describes parameters for the `qos-handler` 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

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

### See Also

“`qos-error`” on page 200
**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`.

---

**Note** – For more information about parameter blocks, see the *Sun Java System Web Server 6.1 SP9 NSAPI Programmer’s Guide*.

---

**Syntax**

```
stage fn="set-variable" [{insert|set|remove}·pblock="name=value" . . .] [name="value" . . .]
```

**Parameters**

The following table describes parameter values for the `set-variable` function.
### TABLE 4–7 set-variable Parameter Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| pblock | One of the following Session/Request parameter block names:  
- `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->client` function in the "Data Structure Reference" chapter of the *Sun Java System Web Server 6.1 SP9 NSAPI Programmer’s Guide*.  
- `vars`: Contains the server’s working variables, which includes anything not specifically found in the `reqpb`, `headers`, or `srvhdrs` pblocks. The contents of this pblock differ, depending upon the specific request and the type of SAF.  
- `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.  
- `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 Java System Web Server 6.1 SP9 NSAPI Programmer’s Guide*.  
- `srvhdrs`: Contains elements of 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 Java System Web Server 6.1 SP9 NSAPI Programmer’s Guide*. |

*Note* – For more information about parameter blocks, see the *Sun Java System Web Server 6.1 SP9 NSAPI Programmer’s Guide*.  

| name | The variable to set. |
| value | The string assigned to the variable specified by `name`. |

### Variables

The following tables lists variables supported by the `set-variable` SAF.
## 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 <em>Sun Java System Web Server 6.1 SP9 NSAPI Programmer’s Guide</em>.</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 <em>Sun Java System Web Server 6.1 SP9 NSAPI Programmer’s Guide</em>.</td>
</tr>
<tr>
<td>find-pathinfo-forward</td>
<td>Path information after the file name in a URI. See “find-pathinfo” on page 141.</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” on page 126.</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 <em>Sun Java System Web Server 6.1 SP9 NSAPI Programmer’s Guide</em>.</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” on page 126.</td>
</tr>
<tr>
<td>senthdrs</td>
<td>A boolean value that indicates whether HTTP response headers have been sent to the client.</td>
</tr>
<tr>
<td>ssl-unclean-shutdown</td>
<td>A boolean value that can be used to alter the way SSL3 connections are closed. As this violates the SSL3 RFCs, you should only use this with great caution if you know that you are experiencing problems with SSL3 shutdowns.</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 URLs 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:

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

### See Also

"match-browser" on page 120
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” on page 126 tells the server to process directives in a named object.
- “document-root” on page 128 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” on page 129 translates a request for the server’s root home page (/) to a specific file.
- “match-browser” on page 120 matches specific strings in the User-Agent string supplied by the browser, and then modifies the behavior of Sun Java System Web Server based upon the results by setting values for specified variables.
- “ntrans-dav” on page 130 determines whether a request should be handled by the WebDAV subsystem and if so, creates a dav objectset.
- “ntrans-j2ee” on page 131 determines whether a request maps to a Java™ technology-based web application context.
- “pfx2dir” on page 131 translates any URL beginning with a given prefix to a file system directory and optionally enables directives in an additional named object.
- “redirect” on page 133 redirects the client to a different URL.
- “set-variable” on page 122 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” on page 134 removes embedded semicolon-delimited parameters from the path.
- “unix-home” on page 135 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>from</td>
<td>Wildcard pattern that specifies the path to be affected.</td>
</tr>
<tr>
<td>name</td>
<td>Specifies an additional named object in <code>obj.conf</code> whose directives will be applied to this request.</td>
</tr>
<tr>
<td>find-pathinfo-forward</td>
<td>(Optional) Makes the server look for the <code>PATHINFO</code> forward in the path right after the <code>ntrans-base</code> instead of backward from the end of path as the server function <code>assign-name</code> does by default.</td>
</tr>
</tbody>
</table>

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->vars`. By default, `ntrans-base` is set.

This feature can improve performance for certain URLs by reducing the number of stats performed.
### TABLE 4–9 assign-name Parameters (Continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nostat</td>
<td>(Optional) Prevents the server from performing a stat on a specified URL whenever possible. The effect of nostat=&quot;virtual-path&quot; in the NameTrans function assign-name is that the server assumes that a stat on the specified virtual-path will fail. Therefore, use nostat only when the path of the virtual-path does not exist on the system, for example, for NSAPI plug-in 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 /ntrans-base/virtual-path and /ntrans-base/virtual-path/* if ntrans-base is set (the default condition); it does not stat for the URLs /virtual-path and /virtual-path/* if ntrans-base is not set.</td>
</tr>
</tbody>
</table>

| bucket    | (Optional) Common to all 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/webservlet61/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/webservlet61/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" on page 131 function to set up additional path name translations.

**Parameters**

The following table describes parameters for the document-root function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>File system path to the server's root document directory.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</td>
</tr>
</tbody>
</table>

**Examples**

NameTrans fn=document-root root=/usr/sun/webserver61/server1/docs

NameTrans fn=document-root root=$docroot

**See Also**

"pfx2dir" on page 131

**home-page**

Applicable in NameTrans-class directives.

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 Java System Web Server based upon the results by setting values for specified variables. See “match-browser” on page 120.

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 4–12  ntrans-dav Parameters

<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" on page 187

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" on page 189, "error-j2ee" on page 199

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" on page 126 for details of using named objects.)
Parameters

The following table describes parameters for the \texttt{pfx2dir} function.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
Parameter & Description \\
\hline
\texttt{from} & URI prefix to convert. It should not have a trailing slash (/). \\
\texttt{dir} & Local file system directory path that the prefix is converted to. It should not have a trailing slash (/). \\
\texttt{name} & (Optional) Specifies an additional named object in \texttt{obj.conf} whose directives will be applied to this request. \\
\texttt{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{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 \texttt{find-pathinfo-forward} parameter is ignored if the \texttt{ntrans-base} parameter is not set in \texttt{rq->vars} when the server function \texttt{find-pathinfo} is called. By default, \texttt{ntrans-base} is set. \\
& This feature can improve performance for certain URLs by reducing the number of stats performed in the server function \texttt{find-pathinfo}. \\
& On Windows, this feature can also be used to prevent the PATHINFO from the server URL normalization process (changing \texttt{"\ to "} ) when the PathCheck server function \texttt{find-pathinfo} is used. Some double-byte characters have hexadecimal values that may be parsed as URL separator characters such as \texttt{"\ or ". Using the \texttt{find-pathinfo-forward} parameter can sometimes prevent incorrect parsing of URLs containing double-byte characters. \\
\texttt{bucket} & (Optional) Common to all \texttt{obj.conf} functions. \\
\hline
\end{tabular}
\end{table}

Examples

In the first example, the URL \texttt{http://server-name/cgi-bin/resource} (such as \texttt{http://x.y.z/cgi-bin/test.cgi}) is translated to the physical path name \texttt{/httpd/cgi-local/resource} (such as \texttt{/httpd/cgi-local/test.cgi}), and the server also starts processing the directives in the object named \texttt{cgi}.

\texttt{NameTrans fn=pfx2dir from=/cgi-bin dir=/httpd/cgi-local name=cgi}
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).

NameTrans fn=pfx2dir from=/icons/happy dir=/users/nikki/images

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.

NameTrans fn="pfx2dir" find-pathinfo-forward="" from="/cgi-bin" dir="/export/home/cgi-bin" name="cgi"

**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>
<tr>
<td>url-prefix</td>
<td>(Maybe optional) The new URL prefix to return to the client. The from prefix is simply replaced by this URL prefix. If you use this parameter, don’t use url (and vice versa).</td>
</tr>
<tr>
<td>escape</td>
<td>(Optional) Flag that tells the server to util_uri_escape the URL before sending it. It should be yes or no. The default is yes. For more information about util_uri_escape, see the Sun Java System Web Server 6.1 SP9 NSAPI Programmer’s Guide.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all obj.conf functions.</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” on page 122.

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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subdir</td>
<td>Subdirectory within the user’s home directory that contains their web documents.</td>
</tr>
<tr>
<td>pwfile</td>
<td>(Optional) Full path and file name of the password file if it is different from /etc/passwd.</td>
</tr>
<tr>
<td>name</td>
<td>(Optional) Specifies an additional named object 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>

Examples

NameTrans fn=unix-home from=/~ subdir=public_html
NameTrans fn=unix-home from /~ pwfile=/mydir/passwd subdir=public_html

See Also

“find-links” on page 140
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 //, 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" on page 136 checks an access control list for authorization.
- "deny-existence" on page 139 indicates that a resource was not found.
- "find-index" on page 139 locates a default file when a directory is requested.
- "find-links" on page 140 denies access to directories with certain file system links.
- "find-pathinfo" on page 141 locates extra path info beyond the file name for the PATH_INFO CGI environment variable.
- "get-client-cert" on page 142 gets the authenticated client certificate from the SSL3 session.
- "load-config" on page 143 finds and loads extra configuration information from a file in the requested path.
- "match-browser" on page 120 matches specific strings in the User-Agent string supplied by the browser, and then modifies the behavior of Sun Java System Web Server based upon the results by setting values for specified variables.
- "nt-uri-clean" on page 145 denies access to requests with unsafe path names by indicating not found.
- "ntcgicheck" on page 146 looks for a CGI file with a specified extension.
- "pcheck-dav" on page 147 inserts a DAV-specific service function.
- "require-auth" on page 147 denies access to unauthorized users or groups.
- "set-variable" on page 122 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" on page 148 specifies a virtual index for a directory.
- "ssl-check" on page 149 checks the secret keysize.
- "ssl-logout" on page 150 invalidates the current SSL session in the server’s SSL session cache.
- "unix-uri-clean" on page 150 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 <code>obj.conf</code> functions.</td>
</tr>
</tbody>
</table>

**Example**

```
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 `/http/docs/index.html` would be named `/http/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\textsuperscript{TM} (JSP\textsuperscript{TM}) technology that need to be interpreted by the server. To compress the dynamic content generated by these types of files, use the \texttt{http-compression} filter.

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

**Parameters**

The following table describes parameters for the \texttt{find-compressed} function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| \texttt{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. |
| \texttt{vary} | Specifies whether to insert a \texttt{Vary: Accept-Encoding} header. Possible values are yes or no.  
  - If set to yes, a \texttt{Vary: Accept-Encoding} header is always inserted when a compressed version of a file is selected.  
  - If set to no, a \texttt{Vary: Accept-Encoding} header is never inserted.  
  By default, the value is set to yes. |
| \texttt{bucket} | (Optional) Common to all \texttt{obj.conf} functions. |

**Example**

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

**See Also**

\texttt{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 REQ_NOACTION. 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 obj.conf 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” on page 129, 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 4-21  find-index Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index-names</td>
<td>Comma-separated list of index filenames to look for. Use spaces only if they are part of a filename. 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 4-22  find-links Parameters**

<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>- <code>h</code> is hard links</td>
</tr>
<tr>
<td></td>
<td>- <code>s</code> is soft links</td>
</tr>
<tr>
<td></td>
<td>- <code>o</code> allows symbolic links from user home directories only if the user owns the target of the link</td>
</tr>
</tbody>
</table>
find-links

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.

checkFileExistence
Checks linked file for existence and aborts request with 403 (forbidden) if this check fails.

bucket
(Optional) Common to all obj.conf functions.

Examples

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

See Also

"unix-home" on page 135

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. 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 auth-cert in the Request-&gt;vars pblock, and the function returns REQ_PROCEED, 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 PROTOCOL_ FORBIDDEN, and the function returns REQ_ABORTED.</td>
</tr>
<tr>
<td></td>
<td>• 0 tells the function to return REQ_NOACTION if the client certificate is not present after dorequest is handled.</td>
</tr>
</tbody>
</table>
### TABLE 4–24  get-client-cert Parameters  
(Continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 obj.conf functions.</td>
</tr>
</tbody>
</table>

### Example

```
# 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 `p` 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`.

```
<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>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `file`    | (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`.
| `disable-types` | (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. |
| `descend` | (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. |
TABLE 4–25  load-config Parameters  (Continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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.

```xml
<Object ppath="*secret*">
  PathCheck fn="load-config" file="checkaccess.nsconfig"
  basedir="D:/Sun/WebServer61/server1/docs/nikki" descend="1"
</Object>
```

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 Java System Web Server based upon the results by setting values for specified variables. See “match-browser” on page 120.

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>//</code> sequences are allowed.</td>
</tr>
<tr>
<td>dotdirok</td>
<td>If present, <code>//</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

See Also

“unix-uri-clean” on page 150

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 obj.conf functions.</td>
</tr>
</tbody>
</table>
Example
PathCheck fn=ntcgicheck extension=pl

See Also
“send-cgi” on page 180, “send-wincgi” on page 186, “send-shellcgi” on page 185

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>
<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>

See Also
“ntrans-dav” on page 130, “service-dav” on page 187

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>
<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>
<tr>
<td>auth-user</td>
<td>(Optional) Specifies a wildcard list of users who are allowed access. If this parameter is not provided, any user authorized by the authorization function is allowed access.</td>
</tr>
<tr>
<td>auth-group</td>
<td>(Optional) Specifies a wildcard list of groups that are allowed access.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all <code>obj.conf</code> functions.</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” on page 117, “basic-ncsa” on page 118

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” on page 122.

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™ Java System 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 applicationPathCheck 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>
<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 obj.conf 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>
<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>

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.
### TABLE 4–33  unix-uri-clean Parameters

<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” on page 145

### 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” on page 156 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” on page 152 sets the Content-Type header for the response to a specific type.
- “match-browser” on page 120 matches specific strings in the User-Agent string supplied by the browser, and then modifies the behavior of Sun Java System Web Server based upon the results by setting values for specified variables.
- "set-default-type" on page 153 allows you to define a default charset, content-encoding, and content-language for the response being sent back to the client.
“set-variable” on page 122 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” on page 154 requests that .htm and .html files are parsed for server-parsed HTML commands.

“type-by-exp” on page 155 sets the Content-Type header for the response based on the requested path.

“type-by-extension” on page 156 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>
<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>
</tbody>
</table>
TABLE 4–34  force-type Parameters (Continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 &quot;; charset=charset&quot; 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" on page 156, "type-by-exp" on page 155

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 Java System Web Server based upon the results by setting values for specified variables. See "match-browser" on page 120.

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 4-35  set-default-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 &quot;; charset=charset&quot; 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="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" on page 122.

### 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 4–36  shtml-hacktype Parameters

<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" on page 156, 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>
</tbody>
</table>
TABLE 4–37 type-by-exp Parameters (Continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 &quot;; charset=charset&quot; 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” on page 156, “force-type” on page 152

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 Java System Web Server 6.1 SP9 Administrator’s Configuration File Reference.

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

type=text/html exts=htm,html

tag=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 \texttt{type-by-extension} function.

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

Example

\texttt{ObjectType fn=type-by-extension}

See Also

“\texttt{type-by-exp}” on page 155, “\texttt{force-type}” on page 152

\textbf{Input}

All Input directives are executed when the server or a plug-in 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 Java System 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 \texttt{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 \texttt{filter_create} function to define the filter's position in the filter stack, and the \texttt{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 \texttt{Input fn="insert-filter"} and \texttt{Output fn="insert-filter"} directives in \texttt{obj.conf} becomes important.

The following Input-class functions are described in detail in this section:
“insert-filter” on page 158 adds a filter to the filter stack to process incoming data.

“match-browser” on page 120 matches specific strings in the User-Agent string supplied by the browser, and then modifies the behavior of Sun Java System Web Server based upon the results by setting values for specified variables.

“remove-filter” on page 159 removes a filter from the filter stack.

“set-variable” on page 122 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 are 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 Java System Web Server based upon the results by setting values for specified variables. See “match-browser” on page 120.

**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 then 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 <em>obj.conf</em> 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” on page 122.
All Output directives are executed when the server or a plug-in 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 position of a 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” on page 160** adds a filter to the filter stack to process outgoing data.
- **“match-browser” on page 120** matches specific strings in the `User-Agent` string supplied by the browser, and then modifies the behavior of Sun Java System Web Server based upon the results by setting values for specified variables.
- **“remove-filter” on page 161** removes a filter from the filter stack.
- **“set-variable” on page 122** 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

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 Java System Web Server based upon the results by setting values for specified variables. See “match-browser” on page 120.

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 4–42 remove-filter Parameters

<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 obj.conf functions.</td>
</tr>
</tbody>
</table>

**Example**

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" on page 122.

**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, or query) is executed.

For more information about the UseOutputStreamSize, flushTimer, ChunkedRequestBufferSize, and ChunkedRequestTimeout parameters, see Buffered Streams in the Sun Java System Web Server 6.1 SP9 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 Java System Web Server 6.1 SP9 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” on page 183 function. The directive that sets the default is:
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 Java System Web Server 6.1 SP9 NSAPI Programmer’s Guide.

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

- "add-footer" on page 165 appends a footer specified by a file name or URL to an HTML file.
- "add-header" on page 166 prepends a header specified by a file name or URL to an HTML file.
- "append-trailer" on page 167 appends text to the end of an HTML file.
- "imagemap" on page 169 handles server-side image maps.
- "index-common" on page 169 generates a fancy list of the files and directories in a requested directory.
- "index-simple" on page 171 generates a simple list of files and directories in a requested directory.
- "key-too-small" on page 172 indicates to the client that the provided certificate key size is too small to accept.
- "list-dir" on page 173 lists the contents of a directory.
- "make-dir" on page 175 creates a directory.
- "match-browser" on page 120 matches specific strings in the User-Agent string supplied by the browser, and then modifies the behavior of Sun Java System Web Server based upon the results by setting values for specified variables.
- "query-handler" on page 176 handles the HTML ISINDEX tag.
- "remove-dir" on page 176 deletes an empty directory.
- "delete-file" on page 177 deletes a file.
- "remove-filter" on page 178 removes a refilter from the filter stack.
- "rename-file" on page 179 renames a file.
- "send-cgi" on page 180 sets up environment variables, launches a CGI program, and sends the response to the client.
- "send-error" on page 182 sends an HTML file to the client in place of a specific HTTP response status.
- "send-file" on page 183 sends a local file to the client.
- "send-range" on page 185 sends a range of bytes of a file to the client.
- "send-shell-cgi" on page 185 sets up environment variables, launches a shell CGI program, and sends the response to the client.
- "send-wincgi" on page 186 sets up environment variables, launches a WinCGI program, and sends the response to the client.
- "service-dav" on page 187 services static content and restarts the request with the source uri for dynamic content.
- "service-dump" on page 188 creates a performance report based on collected performance bucket data.
- "service-j2ee" on page 189 services requests made to Java web applications.
- "service-trace" on page 190 services TRACE requests.
- "set-variable" on page 122 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" on page 191 parses an HTML file for server-parsed HTML commands.
- "stats-xml" on page 192 creates a performance report in XML format.
- "upload-file" on page 194 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” on page 167 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 NSIntAbsFilePath 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>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>
</tbody>
</table>
add-footer Parameters (Continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;type&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;method&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;query&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;UseOutputStreamSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;flushTimer&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestBufferSize&quot; on page 163</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"
NSIntAbsFilePath="yes"

See Also

"append-trailer" on page 167, "add-header" on page 166

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 4–44  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. By default, the path name is relative. If the path name is absolute, pass the <strong>NSNotificationFilePath</strong> 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><strong>NSNotificationFilePath</strong></td>
<td>(Optional) If the file parameter is specified, the <strong>NSNotificationFilePath</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>“type” on page 162</td>
<td>(Optional) Common to all <strong>Service</strong>-class functions.</td>
</tr>
<tr>
<td>“method” on page 162</td>
<td>(Optional) Common to all <strong>Service</strong>-class functions.</td>
</tr>
<tr>
<td>“query” on page 162</td>
<td>(Optional) Common to all <strong>Service</strong>-class functions.</td>
</tr>
<tr>
<td>“UseOutputStreamSize” on page 163</td>
<td>(Optional) Common to all <strong>Service</strong>-class functions.</td>
</tr>
<tr>
<td>“flushTimer” on page 163</td>
<td>(Optional) Common to all <strong>Service</strong>-class functions.</td>
</tr>
<tr>
<td>“ChunkedRequestBufferSize” on page 163</td>
<td>(Optional) Common to all <strong>Service</strong>-class functions.</td>
</tr>
<tr>
<td>bucket</td>
<td>(Optional) Common to all <strong>obj.conf</strong> 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"  
NSNotificationFilePath="yes"

### See Also

“add-footer” on page 165, “append-trailer” on page 167

### 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 4-45 append-trailer Parameters**

<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>&quot;type&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;method&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;query&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;UseOutputStreamSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;flushTimer&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestBufferSize&quot; on page 163</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=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:"
See Also
“add-footer” on page 165, “add-header” on page 166

imagemap
Applicable in Service-class directives.

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>“type” on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“method” on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“query” on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“UseOutputStreamSize” on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“flushTimer” on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“ChunkedRequestBufferSize” on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“ChunkedRequestTimeout” on page 163</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

index-common
Applicable in Service-class directives.
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” on page 171, 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” on page 139 or “home-page” on page 129.

The icons displayed are `.gif` files dependent on the `content-type` of the file, as listed in the following table:

<table>
<thead>
<tr>
<th>Content-type</th>
<th>Icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;text/*&quot;</td>
<td><code>text.gif</code></td>
</tr>
<tr>
<td>&quot;image/*&quot;</td>
<td><code>image.gif</code></td>
</tr>
<tr>
<td>&quot;audio/*&quot;</td>
<td><code>sound.gif</code></td>
</tr>
<tr>
<td>&quot;video/*&quot;</td>
<td><code>movie.gif</code></td>
</tr>
<tr>
<td>&quot;application/octet-stream&quot;</td>
<td><code>binary.gif</code></td>
</tr>
<tr>
<td>directory</td>
<td><code>menu.gif</code></td>
</tr>
<tr>
<td>all others</td>
<td><code>unknown.gif</code></td>
</tr>
</tbody>
</table>

**Parameters**

The following table describes parameters for the `index-common` function.
<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 .html 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 .html and incorporated as preformatted plain text (bracketed by &lt;PRE&gt; and).</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 .html 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 .html and incorporated as preformatted plain text (enclosed by &lt;PRE&gt; and &lt;/PRE&gt;).</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=index-common type=magnus-internal/directory method=(GET|HEAD) header=hdr readme=rdme.txt

**See Also**

“index-simple” on page 171, “find-index” on page 139, “home-page” on page 129

**index-simple**

Applicable in Service-class directives.
The **index-simple** function generates a simple index of the files in the requested directory. This function 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" on page 139 or "home-page" on page 129.

**Parameters**

The following table describes parameters for the **index-simple** function.

**TABLE 4–49  index-simple Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;type&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;method&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;query&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;UseOutputStreamSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;FlushTimer&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestBufferSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestTimeout&quot; on page 163</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" on page 169

**key-tooosmall**

Applicable in Service-class directives.
Note – This function is provided for backward compatibility only and was deprecated in Sun Java System Web Server 4.x. It is replaced by the PathCheck-class SAF “ssl-check” on page 149.

The key-too small 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-too small function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“type” on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“method” on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“query” on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“UseOutputStreamSize” on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“flushTimer” on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“ChunkedRequestBufferSize” on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“ChunkedRequestTimeout” on page 163</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

```
<Object ppath=/mydocs/secret/>Service fn=key-too small</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>&quot;type&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;method&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;query&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;UseOutputStreamSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;flushTimer&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestBufferSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestTimeout&quot; on page 163</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=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>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;type&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;method&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;query&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;UseOutputStreamSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;flushTimer&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestBufferSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestTimeout&quot; on page 163</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="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 Java System Web Server based upon the results by setting values for specified variables. See "match-browser" on page 120.
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>&quot;type&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;method&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;query&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;UseOutputStreamSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;flushTimer&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestBufferSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestTimeout&quot; on page 163</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 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><code>&quot;type&quot;</code> on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td><code>&quot;method&quot;</code> on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td><code>&quot;query&quot;</code> on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td><code>&quot;UseOutputStreamSize&quot;</code> on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td><code>&quot;flushTimer&quot;</code> on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td><code>&quot;ChunkedRequestBufferSize&quot;</code> on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td><code>&quot;ChunkedRequestTimeout&quot;</code> on page 163</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-dir" method="RMDIR"
```

### delete-file

Applicable in Service-class directives.

The `delete-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 `delete-file` when the request method is `DELETE`. 
Parameters
The following table describes parameters for the delete-file function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;type&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;method&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;query&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;UseOutputStreamSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;flushTimer&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestBufferSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestTimeout&quot; on page 163</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="delete-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 4–56  remove-filter Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>Specified the name of the filter to remove.</td>
</tr>
<tr>
<td>&quot;type&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;method&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;query&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;UseOutputStreamSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;flushTimer&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestBufferSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestTimeout&quot; on page 163</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. This file 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 4–57 rename-file Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;type&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;method&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;query&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;UseOutputStreamSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;flushTimer&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestBufferSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestTimeout&quot; on page 163</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="rename-file" method="MOVE"

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 more information about the CGI environment variables and their NSAPI equivalents, see the Sun Java System Web Server 6.1 SP9 NSAPI Programmer’s Guide.

For more information about CGI, see the Sun Java System Web Server 6.1 SP9 Administrator’s Guide and the Sun Java System Web Server 6.1 SP9 Programmer’s Guide.

There are three ways to change the timing used to flush the CGI buffer:

- Adjust the interval between flushes using the “Service” on page 162 parameter.
- Adjust the buffer size using the “Service” on page 162 parameter.
- Force Sun Java System 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>
<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 running 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>&quot;type&quot;</td>
<td>(Optional) Common to all <code>Service</code>-class functions.</td>
</tr>
<tr>
<td>&quot;method&quot;</td>
<td>(Optional) Common to all <code>Service</code>-class functions.</td>
</tr>
<tr>
<td>&quot;query&quot;</td>
<td>(Optional) Common to all <code>Service</code>-class functions.</td>
</tr>
<tr>
<td>&quot;UseOutputStreamSize&quot;</td>
<td>(Optional) Common to all <code>Service</code>-class functions.</td>
</tr>
<tr>
<td>&quot;flushTimer&quot;</td>
<td>(Optional) Common to all <code>Service</code>-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestBufferSize&quot;</td>
<td>(Optional) Common to all <code>Service</code>-class functions.</td>
</tr>
</tbody>
</table>
TABLE 4–58  send-cgi Parameters (Continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;ChunkedRequestTimeout&quot;</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>on page 163</td>
<td></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 Java System Web Server 6.1 SP9 Administrator's Configuration File Reference.

```xml
<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 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 4–59  send-error Parameters

<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>
</tbody>
</table>
**TABLE 4-59 send-error Parameters (Continued)**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;type&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;method&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;query&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;UseOutputStreamSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;flushTimer&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestBufferSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestTimeout&quot; on page 163</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

**send-file**

Applicable in Service-class directives.

The send-file function sends the contents of the requested file to the client. This function 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 Java System Web Server 6.1 SP9 NSAPI Programmer’s Guide.

**Parameters**

The following table describes parameters for the send-file function.
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>&quot;type&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;method&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;query&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;UseOutputStreamSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;flushTimer&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestBufferSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestTimeout&quot; on page 163</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/*" 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.

<Object name=default>
  ...
  NameTrans fn="pfx2dir" from="/myurl" dir="/export/mydir", name="myname"
  ...
  Service method=(GET|HEAD|POST) type="*-magnus-internal/*" fn=send-file
  ...
</Object>
<Object name="myname">
  Service method=(GET|HEAD) type="*-magnus-internal/*" fn=send-file
  nocache=""
</Object>
**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 the specified 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>&quot;type&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;method&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;query&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;UseOutputStreamSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;flushTimer&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestBufferSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestTimeout&quot; on page 163</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, see the *Sun Java System Web Server 6.1 SP9 Administrator's Guide.*
Parameters
The following table describes parameters for the `send-shellcgi` function.

**TABLE 4–62  send-shellcgi Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;type&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;method&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;query&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;UseOutputStreamSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;flushTimer&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestBufferSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestTimeout&quot; on page 163</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, see *Sun Java System Web Server 6.1 SP9 Administrator’s Guide*.

Parameters
The following table describes parameters for the `send-wincgi` function.
TABLE 4–63  send-wincgi Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“type” on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“method” on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“query” on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“UseOutputStreamSize” on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“flushTimer” on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“ChunkedRequestBufferSize” on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“ChunkedRequestTimeout” on page 163</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` are 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 Java System Web Server 6.1 SP9 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>&quot;method&quot; on page 162</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">
  ...
  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" on page 192

**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 115)

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>&quot;type&quot; on page 162</td>
<td>Must be perf for this function.</td>
</tr>
<tr>
<td>&quot;method&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;query&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;UseOutputStreamSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;flushTimer&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestBufferSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestTimeout&quot; on page 163</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" on page 192

**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.
### TABLE 4–66  service-j2ee Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;type&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;method&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;query&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;UseOutputStreamSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;flushTimer&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestBufferSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestTimeout&quot; on page 163</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

```
<Object name=default>
  NameTrans fn="ntrans-j2ee" name="j2ee"
  ...
</Object>

<Object name=j2ee>
  Service fn="service-j2ee"
</Object>
```

### See Also

“ntrans-j2ee” on page 131, “error-j2ee” on page 199

### service-trace

Applicable in Service-class directives.

The service-trace function services TRACE requests. TRACE requests are 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 4–67 service-trace Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“type” on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“method” on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“UseOutputStreamSize” on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“flushTimer” on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“ChunkedRequestBufferSize” on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“ChunkedRequestTimeout” on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>“query” on page 162</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” on page 122.

**shtml_send**

Applicable in Service-class directives.

The `shtml_send` function parses an HTML document and scans 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 plug-in (libShtml.so on UNIX libShtml.dll on Windows) is loaded. See the *Sun Java System Web Server 6.1 SP9 Programmer’s Guide* for server-parsed HTML commands.
Parameters

The following table describes parameters for the shtml_send function.

<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>&quot;type&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;method&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;UseOutputStreamSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;flushTimer&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestBufferSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestTimeout&quot; on page 163</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 the 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 Java System Web Server 6.1 SP9 Performance Tuning, Sizing, and Scaling Guide

Parameters

The following table describes parameters for the stats-xml function.

TABLE 4–69  stats-xml Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;type&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;method&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;query&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;UseOutputStreamSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;flushTimer&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestBufferSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;ChunkedRequestTimeout&quot; on page 163</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:

Init fn="stats-init" update-interval="5" virtual-servers="2000" profiling="yes"

In obj.conf:

<Object name="default">
  ...
  NameTrans fn="assign-name" from="/stats-xml/*" name="stats-xml"
  ...
</Object>
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 4-70 upload-file Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;type&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;method&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;query&quot; on page 162</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;UseOutputStreamSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;flushTimer&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;chunkedRequestBufferSize&quot; on page 163</td>
<td>(Optional) Common to all Service-class functions.</td>
</tr>
<tr>
<td>&quot;chunkedRequestTimeout&quot; on page 163</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>
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” on page 195** records information about the request in the common log format.
- **“flex-log” on page 196** records information about the request in a flexible, configurable format.
- **“match-browser” on page 120** matches specific strings in the User-Agent string supplied by the browser, and then modifies the behavior of Sun Java System Web Server based upon the results by setting values for specified variables.
- **“record-useragent” on page 197** records the client’s IP address and User-Agent header.
- **“set-variable” on page 122** 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 Java System Web Server.

The common log must have been initialized previously by using the init-clf function. For information about rotating logs, see flex-rotate-init in the *Sun Java System Web Server 6.1 SP9 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 4–71 common-log Parameters**

<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**

```
# Log all accesses to the global log file
AddLog fn=common-log
# Log accesses from outside our subnet (198.93.5.*) to nonlocal log
<Client ip="*~198.93.5.*">
AddLog fn=common-log name=nonlocallog
</Client>
```

**See Also**

"record-useragent" on page 197, "flex-log" on page 196

**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 Java System Web Server.

There are also a number of free statistics generators for the common log format. The log format is specified by using the flex-init function call. For information about rotating logs, see flex-rotate-init in the Sun Java System Web Server 6.1 SP9 NSAPI Programmer's Guide.

**Parameters**

The following table describes parameters for the flex-log function.
### TABLE 4–72  flex-log Parameters

<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</td>
</tr>
<tr>
<td></td>
<td>parameter to the flex-init function in magnus.conf. If no name is given,</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>rather than looking up and logging the DNS name. This will improve</td>
</tr>
<tr>
<td></td>
<td>performance if DNS is off in the magnus.conf file. The value of iponly has</td>
</tr>
<tr>
<td></td>
<td>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>
<tr>
<td>buffers-per-file</td>
<td>Specifies the number of buffers for a given log file. The default value is</td>
</tr>
<tr>
<td></td>
<td>determined by the server.</td>
</tr>
</tbody>
</table>

Access log entries can be logged in strict chronological order by using a single buffer per log file. To accomplish this, add buffers-per-file=”1” to the Init fn=“flex-init” line in magnus.conf. This ensures that requests are logged in chronological order. Note that this approach will result in decreased performance when the server is under heavy load.

### Examples

```plaintext
# Log all accesses to the global log file
AddLog fn=flex-log  
# Log accesses from outside our subnet (198.93.5.*) to
# nonlocallog
<Client ip="!*198.93.5.*">  
AddLog fn=flex-log name=nonlocallog  
</Client>
```

### See Also

"common-log" on page 195, "record-useragent" on page 197

### 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 Java System Web Server based upon the results by setting values for specified variables. See "match-browser" on page 120.

### 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 <code>init-clf</code> function in <code>magnus.conf</code>. 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 <code>obj.conf</code> functions.</td>
</tr>
</tbody>
</table>

**Example**

```
# Record the client ip address and user-agent to browserlog
AddLog fn=record-useragent name=browserlog
```

**See Also**

“`common-log`” on page 195, “`flex-log`” on page 196

**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`" on page 122.

**Error**

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`” on page 199 handles errors that occur during execution of Java™ 2 Platform, Standard Edition (J2SE platform) applications and modules deployed to the Sun Java System Web Server.
• "match-browser" on page 120 matches specific strings in the User-Agent string supplied by the browser, and then modifies the behavior of Sun Java System Web Server based upon the results by setting values for specified variables.
• "qos-error" on page 200 returns an error page stating which quality of service limits caused the error and what the value of the QOS statistic was.
• "query-handler" on page 200 runs a CGI program instead of referencing the path requested.
• "remove-filter" on page 201 removes a filter from the filter stack.
• "send-error" on page 202 sends an HTML file to the client in place of a specific HTTP response status.
• "set-variable" on page 122 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 Java System Web Server individually or as part of full J2SE applications.

Parameters

The following table describes parameters for the error-j2ee function.

<table>
<thead>
<tr>
<th>TABLE 4–74 error-j2ee Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>bucket</td>
</tr>
</tbody>
</table>

See Also

“ntrans-j2ee” on page 131, “service-j2ee” on page 189

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 Java System Web Server based upon the results by setting values for specified variables. See "match-browser" on page 120.
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 Java System Web Server 6.1 NSAPI Programmer's Guide.

For more information, see the Sun Java System Web Server 6.1 SP9 Performance Tuning, Sizing, and Scaling Guide.

Parameters
The following table describes parameters for the qos-error function.

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

Example
Error fn=qos-error code=503

See Also
“qos-handler” on page 121

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 “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. |

**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 obj.conf 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 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>
</tbody>
</table>
### TABLE 4-78 send-error Parameters (Continued)

<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 &quot;Unauthorized&quot; or &quot;Forbidden&quot;). 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. |

### Example

```plaintext
Error fn=send-error code=401 path=/sun/server61/docs/errors/401.html
```

### set-variable

Applicable in all stage directives. The `set-variable` SAF sets the HTTP status code and aborts the request by returning REQ_ABORTED. To set the HTTP status code without aborting the request, use the error parameter in conjunction with the `noaction` parameter.

To rewrite a HTTP status code, use a `<Client>` tag to match the original status code and an Output directive to set the new status code. The following example would rewrite all 302 Moved Temporarily responses to 301 Moved Permanently responses.

### Example

```plaintext
<Client code="302">
Output fn="set-variable" error="301 Moved Permanently" noaction="true"
</Client>
```
MIME Types

This chapter discusses the MIME types file.

The sections are:

- “Introduction” on page 205
- “Determining the MIME Type” on page 206
- “How the Type Affects the Response” on page 206
- “What Does the Client Do with the MIME Type?” on page 207
- “Syntax of the MIME Types File” on page 207
- “Sample MIME Types File” on page 207

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 Java System 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 (lang), encoding (enc), and content-type (type). At least one of these attributes must be present for each type. The most commonly used attribute is type. The server frequently considers the type when deciding how to generate the response to the client. The enc and lang attributes are rarely used.

The default MIME types file is called mime.types.
Determining the MIME Type

During the ObjectType 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 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 obj.conf that tells the server to look up the MIME type according to the extension is:

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

If the server uses a different SAF, such as force-type, to determine the type, then the MIME types table is not used for that particular request.

For more details of the ObjectType step, see the Sun Java System Web Server 6.1 SP9 NSAPI Programmer’s Guide.

How the Type Affects the Response

The server considers the value of the type attribute when deciding which Service directive in obj.conf to use to generate the response to the client.

By default, if the type does not start with magnus-internal/, the server just sends the requested file to the client. The directive in obj.conf that contains this instruction is:

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

By convention, all values of type that require the server to do something other than just send the requested resource to the client start with 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:

```
type=magnus-internal/imagemap exts=map
type=magnus-internal/cgi exts=.cgi,exe,bat
```

If the type starts with magnus-internal/, the server executes the Service directive in obj.conf that 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 the 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, 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
```

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/rtf             exts=rtf
type=application/x-mif           exts=mif,fm
type=application/x-gtar          exts=gtar
type=application/x-shar          exts=shar
```
<table>
<thead>
<tr>
<th>Type</th>
<th>Extensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>application/x-tar</td>
<td>tar</td>
</tr>
<tr>
<td>application/mac-binhex40</td>
<td>hqx</td>
</tr>
<tr>
<td>audio/basic</td>
<td>au,snd</td>
</tr>
<tr>
<td>audio/x-aiff</td>
<td>aif,aiff,aiffc</td>
</tr>
<tr>
<td>audio/x-wav</td>
<td>wav</td>
</tr>
<tr>
<td>image/gif</td>
<td>gif</td>
</tr>
<tr>
<td>image/ief</td>
<td>ief</td>
</tr>
<tr>
<td>image/jpeg</td>
<td>jpeg,jpg,jpe</td>
</tr>
<tr>
<td>image/tiff</td>
<td>tiff,tif</td>
</tr>
<tr>
<td>image/x-rgb</td>
<td>rgb</td>
</tr>
<tr>
<td>image/x-xbitmap</td>
<td>xbm</td>
</tr>
<tr>
<td>image/x-xpixmap</td>
<td>xpm</td>
</tr>
<tr>
<td>image/x-xwindowdump</td>
<td>xwd</td>
</tr>
<tr>
<td>text/html</td>
<td>htm,html</td>
</tr>
<tr>
<td>text/plain</td>
<td>txt</td>
</tr>
<tr>
<td>text/richtext</td>
<td>rtx</td>
</tr>
<tr>
<td>text/tab-separated-values</td>
<td>tsv</td>
</tr>
<tr>
<td>text/x-setext</td>
<td>etx</td>
</tr>
<tr>
<td>video/mpeg</td>
<td>mpeg,mpg,mpe</td>
</tr>
<tr>
<td>video/quicktime</td>
<td>qt,mov</td>
</tr>
<tr>
<td>video/x-msvideo</td>
<td>avi</td>
</tr>
<tr>
<td>enc=x-gzip</td>
<td>gz</td>
</tr>
<tr>
<td>enc=x-compress</td>
<td>z</td>
</tr>
<tr>
<td>enc=x-uuencode</td>
<td>uu,uue</td>
</tr>
<tr>
<td>magnus-internal/imagemap</td>
<td>map</td>
</tr>
<tr>
<td>magnus-internal/parsed-html</td>
<td>shtml</td>
</tr>
<tr>
<td>magnus-internal/cgi</td>
<td>cgi,exe,bat</td>
</tr>
<tr>
<td>magnus-internal/jsp</td>
<td>jsp</td>
</tr>
</tbody>
</table>
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" on page 209
- "dbswitch.conf" on page 211
- "Deployment Descriptors" on page 213
- "generated.instance.acl" on page 214
- "login.conf" on page 214
- "nsfc.conf" on page 215
- "password.conf" on page 216
- "server.policy" on page 217
- "*.clfilter" on page 218

**certmap.conf**

**Purpose**

Configures how a certificate, designated by *name*, is mapped to an LDAP entry, designated by *issuerDN*.

**Location**

```
server_root/bin/https/install/misc
server_root/userdb
```
Syntax

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:

certmap default default

You can use # at the beginning of a line to indicate a comment.

See Also

Sun Java System Web Server 6.1 SP9 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>
<tbody>
<tr>
<td>DNComps</td>
<td>See Description</td>
<td>Commented out</td>
<td>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.</td>
</tr>
<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: ■ Commented out or empty: sets the filter to &quot;objectClass=*&quot;. ■ Comma-separated attributes: forms the filter.</td>
</tr>
<tr>
<td>verifycert</td>
<td>on or off</td>
<td>off (commented out)</td>
<td>Specifies whether certificates are verified.</td>
</tr>
</tbody>
</table>
### TABLE 6-1  certmap.confProperties (Continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Allowed Values</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CmapLdapAttr</td>
<td>LDAP attribute name</td>
<td>certSubjectDN (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 Java System Web Server uses.

**Location**

`server_root/userdb`

**Syntax**

```
directory name LDAP_URLname: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
```
The following table describes properties in the `dbswitch.conf` file. The Property column lists the property names. The Allowed Values column from the left lists allowed values. The Default Value column from the left lists default values. The Description 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>nsesions</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>
<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 <code>server.xml</code> 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 plug-in is required. For information about how to install this plug-in, see the <em>Sun Java System Web Server 6.1 SP9 Administrator's Guide</em>.</td>
</tr>
<tr>
<td>Property</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>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 Java System Web Server 6.1 SP9 Administrator’s Guide.</td>
</tr>
<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 htaccess. For more information about the syntax of the AuthUserFile, see the Sun Java System Web Server 6.1 SP9 Administrator’s Guide.</td>
</tr>
</tbody>
</table>

**Deployment Descriptors**

**Purpose**

Configures features specific to the Sun Java System 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 Java System 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>
<thead>
<tr>
<th>Deployment Descriptor</th>
<th>Where to Find More Information</th>
</tr>
</thead>
</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.

Location

$server_root$/config

See Also

Sun Java System Web Server 6.1 SP9 Administrator’s Guide

login.conf

Purpose

The login module definition configuration file used by the Java Authentication and Authorization Service (JAAS) for client authentication.

Location

$server_root$/config
**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**

*Sun Java System Web Server 6.1 SP9 Performance Tuning, Sizing, and Scaling Guide*

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>
<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>
</tbody>
</table>
### Purpose

By default, the Sun Java System 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 Java System Web Server 6.1 SP9 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 Java System 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 Java System Web Server 6.1 SP9 Programmer’s Guide

*.clfiter

Purpose

The files obj.conf.clfilter, magnus.conf.clfilter, and server.xml.clfilter contain filter specifications for cluster management operations.

Location

server_root/config
Configuration Changes Between iPlanet Web Server 4.1 and Sun Java System Web Server 6.1

This chapter summarizes major configuration file changes between the 4.1 and the 6.1 version of Sun Java System Web Server. The following 4.1 files are described:

- “magnus.conf” on page 219
- “obj.conf” on page 221
- “contexts.properties” on page 222
- “rules.properties” on page 223
- “servlets.properties” on page 224

**magnus.conf**

The following table 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>AccelFileCache</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” on page 38 and ”VS” on page 39 elements in server.xml</td>
</tr>
<tr>
<td>ACLFile</td>
<td>(none)</td>
<td>Maps to the ”ACLFILE” on page 37 element in server.xml</td>
</tr>
<tr>
<td>Address</td>
<td>(none)</td>
<td>Maps to the ”LS” on page 33 element in server.xml.</td>
</tr>
<tr>
<td>AdminLanguage</td>
<td>(none)</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>4.x Directive</td>
<td>6.1 Directive</td>
<td>Comments</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
<td>--------------------------------------------------------------------------</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 &quot;LS&quot; on page 33 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 ssl2ciphers attribute of the &quot;SSLPARAMS&quot; on page 35 element in server.xml.</td>
</tr>
<tr>
<td>ClientLanguage</td>
<td>(none)</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>DaemonStats</td>
<td>(none)</td>
<td>Obsolete due to new performance statistics system. For more information, see the Sun Java System Web Server 6.1 SP9 Performance Tuning, Sizing, and Scaling Guide for further information.</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 &quot;LOG&quot; on page 65 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>LoadObjects</td>
<td>(none)</td>
<td>See the object file attribute in the &quot;VSCLASS&quot; on page 38 element in server.xml.</td>
</tr>
<tr>
<td>LogVerbose</td>
<td>(none)</td>
<td>See the &quot;Attributes&quot; on page 65 attribute in server.xml.</td>
</tr>
<tr>
<td>MaxThreads</td>
<td>(none)</td>
<td>Obsolete due to new thread handling system.</td>
</tr>
<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>4.x Directive</td>
<td>6.1 Directive</td>
<td>Comments</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>----------</td>
</tr>
<tr>
<td>Port</td>
<td>(none)</td>
<td>See the &quot;LS&quot; on page 33 element in server.xml.</td>
</tr>
<tr>
<td>RootObject</td>
<td>(none)</td>
<td>See the rootobject attribute of the &quot;VSCCLASS&quot; on page 38 element in server.xml.</td>
</tr>
<tr>
<td>RqThrottleMinPerSocket</td>
<td>(none)</td>
<td>See the the acceptorthreads attribute of the &quot;LS&quot; on page 33 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 &quot;LS&quot; on page 33 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 &quot;SSLPARAMS&quot; on page 35 element in server.xml.</td>
</tr>
<tr>
<td>SSL3</td>
<td>(none)</td>
<td>See the ssl3 attribute of the &quot;SSLPARAMS&quot; on page 35 element in server.xml.</td>
</tr>
<tr>
<td>SSL3Ciphers</td>
<td>(none)</td>
<td>See the the ssl3tlsciphers attribute of &quot;SSLPARAMS&quot; on page 35 element in server.xml.</td>
</tr>
<tr>
<td>SSLClientAuth</td>
<td>clientauth</td>
<td>See the cclientauth attribute of the &quot;SSLPARAMS&quot; on page 35 element in server.xml.</td>
</tr>
<tr>
<td>VirtualServerFile</td>
<td>(none)</td>
<td>Obsolete due to virtual server implementation</td>
</tr>
</tbody>
</table>

The obj.conf file has lost its Init directives to the magnus.conf file and acquired new directives and parameters. The following table 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>Init functions</td>
<td>(none)</td>
<td>All functions have moved to magnus.conf except for cache-init and load-types, which are obsolete (for load-types, see the MIME element in the server.xml file).</td>
</tr>
<tr>
<td>Service fn=parse-html</td>
<td>Service fn=sh.html_send</td>
<td></td>
</tr>
</tbody>
</table>

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.

The following table 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, see 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>To add context attributes, implement the javax.servlet.ServletContextListener interface. For more information, see to the Servlet 2.3 specification.</td>
</tr>
<tr>
<td>initArgs</td>
<td>(none)</td>
<td></td>
</tr>
<tr>
<td>respondCookieVersion</td>
<td>(none)</td>
<td>Might be supported in a future release.</td>
</tr>
<tr>
<td>tempDir</td>
<td>tempdir property</td>
<td></td>
</tr>
</tbody>
</table>
The rules.properties file is no longer supported in Sun Java System 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
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

The following table 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 “WEBAPP” on page 49 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 Java System Web Server 6.1

This chapter summarizes major configuration file changes between the 6.0 and the 6.1 version of Sun Java System Web Server. The following files are described:

- “magnus.conf” on page 225
- “obj.conf” on page 227
- “server.xml” on page 227

magnus.conf

This section lists the magnus.conf-related changes in the following areas:

- “Init Functions” on page 225
- “Directives” on page 226

Init Functions

The magnus.conf file in SUN Java System 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>
<tr>
<td>NSServletLateInit</td>
<td>(none)</td>
<td>Removed.</td>
</tr>
</tbody>
</table>
TABLE B–1  magnus.conf Init Functions  (Continued)

<table>
<thead>
<tr>
<th>6.0 Function/Parameter</th>
<th>6.1 Function/Parameter</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>nt-console-init</td>
<td>createconsole</td>
<td>Removed. On Windows, you can configure the createconsole attribute of the &quot;LOG&quot; on page 65 element to redirect stderr output to the console.</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 B–2  Changes in magnus.conf Directives

<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 &quot;Attributes&quot; on page 65 attribute of the &quot;LOG&quot; on page 65 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 &quot;Attributes&quot; on page 65 attribute of the &quot;LOG&quot; on page 65 element in server.xml.</td>
</tr>
<tr>
<td>LogVald</td>
<td>&quot;Attributes&quot; on page 65</td>
<td>See the &quot;Attributes&quot; on page 65 attribute of the &quot;LOG&quot; on page 65 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 (Continued)

<table>
<thead>
<tr>
<th>Directive</th>
<th>6.0 Value</th>
<th>6.1 Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServerConfigurationFile</td>
<td>(none)</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>ServerID</td>
<td>(none)</td>
<td>Deprecated.</td>
<td></td>
</tr>
<tr>
<td>ServerName</td>
<td>(none)</td>
<td>Deprecated.</td>
<td>See the servername attribute of the &quot;LS&quot; on page 33 element in the server.xml file.</td>
</tr>
<tr>
<td>#ServerRoot</td>
<td>(none)</td>
<td>Deprecated.</td>
<td></td>
</tr>
</tbody>
</table>

obj.conf

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>
<thead>
<tr>
<th>Support 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 Java System 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:

- Table B–4
- Table B–5
- Table B–6

The following table lists the correspondences between the server.xml file in iPlanet Web Server 6.0 and the server.xml file in Sun Java System 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 Java System Web Server 6.1:

### TABLE B–5  start-jvm and server.xml Correspondences

<table>
<thead>
<tr>
<th>NSES_JDK</th>
<th>“Attributes” on page 50</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSES_CLASSPATH</td>
<td>“Attributes” on page 50</td>
</tr>
<tr>
<td>NSES_JRE_RUNTIME_LIBPATH</td>
<td>“Attributes” on page 50</td>
</tr>
<tr>
<td>NSES_JRE_RUNTIME_CLASSPATH</td>
<td>Use the -Xbootclasspath 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 Java System Web Server 6.1:

### TABLE B–6  jvm12.conf and server.xml Correspondences

<table>
<thead>
<tr>
<th>jvm.minHeapSize</th>
<th>Use the -Xms&lt;value&gt; JVM option. Example: &lt;JVMOPTIONS&gt;-Xms128m -Xmx256m&lt;/JVMOPTIONS&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>jvm.maxHeapSize</td>
<td>Use the -Xmx&lt;value&gt; JVM option. Example: &lt;JVMOPTIONS&gt;-Xms128m -Xmx256m&lt;/JVMOPTIONS&gt;</td>
</tr>
</tbody>
</table>
### TABLE B–6  jvm12.conf and server.xml Correspondences (Continued)

<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 “JVMOPTIONS” on page 52 element.</td>
</tr>
<tr>
<td>jvm.profiler</td>
<td>Use the “PROFILER” on page 53 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 LOGLEVEL 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 server.policy file:</td>
</tr>
<tr>
<td></td>
<td><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 “Attributes” on page 50 attribute of the “JAVA” on page 50 element.</td>
</tr>
<tr>
<td>jvm.enableDebug</td>
<td>Use the debug and debugoptions attributes of the “JAVA” on page 50 element.</td>
</tr>
<tr>
<td>jvm.classpath</td>
<td>Use the “Attributes” on page 50 and “Attributes” on page 50 attributes of the JAVA element.</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 Java System Web Server 6.1:

### TABLE B–7  web-apps.xml and sun-web.xml Correspondences

<table>
<thead>
<tr>
<th>&lt;config-param&lt;param-name&gt; redirect-to-absolute-url&lt;/param-name&gt;</th>
<th>Use the <code>&lt;property name=&quot;relativeRedirectAllowed&quot; value=&quot;true&quot;&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;param-value&gt;false&lt;/param-value&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;/config-param&gt;</td>
<td></td>
</tr>
</tbody>
</table>
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>
</tbody>
</table>
**TABLE C-1 Format Strings (Continued)**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Allowed Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>%c</td>
<td>Date &amp; time &quot;%m/%d/%y %H:%M:%S&quot;</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>
Alphabetical List of Server Configuration Elements

A  “AUTHREALM” on page 55
C  “CONNECTIONPROPERTY” on page 63
     “CUSTOMRESOURCE” on page 57
D  “DAVCOLLECTION” on page 45
     “DESCRIPTION” on page 32
     “DISPLAYNAME” on page 48
E  “EXTERNALJNDIRESOURCE” on page 58
J  “JAVA” on page 50
     “JDBCCONNECTIONPOOL” on page 59
     “JDBCRESOURCE” on page 59
     “JVMOPTIONS” on page 52
L  “LS” on page 33
M  “MAILRESOURCE” on page 64
     “MIME” on page 36
P  “PROFILER” on page 53
     “PROPERTY” on page 31
Q  “QOSPARAMS” on page 41
R  “RESOURCES” on page 56
S  “SEARCH” on page 47
Alphabetical List of Server Configuration Elements

“SEARCHCOLLECTION” on page 47
“SECURITY” on page 54
“SERVER” on page 30
“SSLPARAMS” on page 35
U “USERDB” on page 42
V “VARS” on page 32
“VS” on page 39
“VSCLASS” on page 38
W “WEBAPP” on page 49
Alphabetical List of Predefined SAFs

This appendix provides an alphabetical list for the easy lookup of predefined SAFs.

A
- “add-footer” on page 165
- “add-header” on page 166
- “append-trailer” on page 167
- “assign-name” on page 126

B
- “basic-auth” on page 117
- “basic-ncsa” on page 118

C
- “check-acl” on page 136
- “common-log” on page 195

D
- “deny-existence” on page 139
- “document-root” on page 128

E
- “error-j2ee” on page 199

F
- “find-compressed” on page 137
- “find-index” on page 139
- “find-links” on page 140
- “find-pathinfo” on page 141
Alphabetical List of Predefined SAFs

"flex-log" on page 196
"force-type" on page 152

G
"get-client-cert" on page 142
"get-sslid" on page 119

H
"home-page" on page 129

I
"imagemap" on page 169
"insert-filter" on page 158
"insert-filter" on page 160

K
"key-too small" on page 172

L
"list-dir" on page 173
"load-config" on page 143

M
"make-dir" on page 175
"match-browser" on page 120

N
"ntcgicheck" on page 146
"ntrans-dav" on page 130
"ntrans-j2ee" on page 131
"nt-uri-clean" on page 145

P
"pcheck-dav" on page 147
"pfx2dir" on page 131

Q
"qos-error" on page 200
"qos-handler" on page 121
"query-handler" on page 176
Alphabetical List of Predefined SAFs

“query-handler” on page 200

R
“record-useragent” on page 197
“redirect” on page 133
“remove-dir” on page 176
“delete-file” on page 177
“remove-filter” on page 159
“remove-filter” on page 161
“remove-filter” on page 178
“remove-filter” on page 201
“rename-file” on page 179
“require-auth” on page 147

S
“send-cgi” on page 180
“send-error” on page 182
“send-error” on page 202
“send-file” on page 183
“send-range” on page 185
“send-shellcgi” on page 185
“send-wincgi” on page 186
“service-dav” on page 187
“service-dump” on page 188
“service-j2ee” on page 189
“service-trace” on page 190
“set-default-type” on page 153
“set-variable” on page 122
“set-virtual-index” on page 148
Alphabetical List of Predefined SAFs

“shtml_send” on page 191
“shtml-hacktype” on page 154
“ssl-check” on page 149
“ssl-logout” on page 150
“stats-xml” on page 192
“strip-params” on page 134
T
“type-by-exp” on page 155
“type-by-extension” on page 156
U
“unix-home” on page 135
“unix-uri-clean” on page 150
“upload-file” on page 194
Index

A
 AccelFileCache directive, 219
 acceptlanguage, 39
 AcceptLanguage directive, 219
 AcceptTimeout, magnus.conf directive, 80
 access log, 65
 ACL, magnus.conf directives, 90-91
 acl parameter, 137
 ACLCacheLifetime, magnus.conf directive, 90-91
 ACLCacheLifetime directive, 104
 ACLFILE, 37
 ACLFile directive, 219
 ACLGroupCacheSize, magnus.conf directive, 91
 ACLGroupCacheSize directive, 104
 ACLUserCacheSize, magnus.conf directive, 91
 ACLUserCacheSize directive, 104
 add-footer function, 165-166
 add-header function, 166-167
 addCgiInitVars parameter, 192
 AddLog, function descriptions, 195-198
 Address directive, 219
 Administration interface, more information about, 16
 AdminLanguage directive, 219,226
 alias directory, 23
 alphabetical reference, SAFs, 235-238
 append-trailer function, 167-169
 assign-name function, 126-128
 AsyncDNS, magnus.conf directive, 78
 AsyncDNS directive, 105 220 226
 auth-group parameter, 148
 auth-type parameter, 117 118 148
 auth-user parameter, 148
 authdb property, 223
 AUTHREALM, 55-56
 AuthTrans, function descriptions, 116-125

B
 basedir parameter, 145
 basic-auth function, 117-118
 basic-ncsa function, 118-119
 bin directory, 23
 binddn property, 212
 bindpw property, 212
 BlockingListenSockets directive, 220
 bong-file parameter, 139,150
 bucket parameter, 115
 buffer-size parameter, 100
 buffers-per-file parameter, 100,197
 bufferSize property, 223
 built-in SAFs, 113-203

cache-size parameter, 99
 CacheFileContent parameter, 215
 certificates, settings in magnus.conf, 91-93
 CGI, settings in magnus.conf, 87-89
 CGIEpirationTimeout, magnus.conf directive, 87
 CGIEpirationTimeout directive, 105
 cgistub-path parameter, 101
 CGIStubIdleTimeout, magnus.conf directive, 88
 CGIStubIdleTimeout directive, 105
Index

CGIWaitPid, magnus.conf directive, 88
CGIWaitPid directive, 105 220
charset parameter, 153 154 156
check-acl function, 136-137
checkFileExistence parameter, 141
ChildRestartCallback, magnus.conf directive, 95
ChildRestartCallback directive, 105
croot parameter, 181
chunked encoding, 94-95
ChunkedRequestBufferSize
  magnus.conf directive, 94-95
  obj.conf Service parameter, 163
ChunkedRequestBufferSize directive, 105
ChunkedRequestTimeout
  magnus.conf directive, 95
  obj.conf Service parameter, 163-165
ChunkedRequestTimeout directive, 105
cindex-init function, 98
Ciphers directive, 220
ClassCache directory, 24
ClassCache file, 25
classpath property, 223 224
 .clfilter files, 218
clientauth, 36
ClientLanguage directive, 220 226
CmapLdapAttr property, 211
code parameter, 200,201,203
code property, 224
common-log function, 195-196
conf_bk directory, 24
conf_bk file, 26
config directory, 24
config file, 26
configuration, new, installing dynamically, 26
configuration files, stored in server root, 24
CONNECTIONPROPERTY, 63-64
connectons, settings in magnus.conf, 79-85
ConnQueueSize, magnus.conf directive, 81
ConnQueueSize directive, 105
ccontent-type icons, 170
ccontentTypelgnoreFromSSI property, 223
ccontext property, 224
contexts.properties, changes to, 222-223
convergence tree
  auxiliary class inetSubscriber, 68
  in LDAP schema, 67
  organization of, 68
  user entries are called inetOrgPerson, 68
CopyFiles parameter, 216
core SAFs, 113-203
Core Server Elements, 30-32
createconsole, 66
CUSTOMRESOURCE, 57-58

D
DaemonStats directive, 220
DAV, 43-45
DAVCOLLECTION, 45-46
day of month, 231
dbm parameter, 118
dcsuffix property, 212
default virtual server, for a connection group, 35
DefaultCharSet directive, 220 226
DefaultLanguage, magnus.conf directive, 78
DefaultLanguage directive, 105
define-perf-bucket function, 99
delete-file function, 177-178
deny-existence function, 139
descend parameter, 144
description parameter, 99
digest directory, 25
digestauth property, 212
digestfile, 213
dir parameter, 132 141 181
directives, obj.conf, 113-203
disable parameter, 103 140
disable-types parameter, 144
DISPLAYNAME, 48
DNComps property, 210
DNS, magnus.conf directive, 79
dns-cache-init function, 99
DNS directive, 106
DNS lookup, directives in magnus.conf, 78-79
docRoot property, 223
docs directory, 23
document-root function, 128-129
domain component tree, 68
domain component tree (dc), 68-69
dorequest parameter, 142
dotdirok parameter, 146, 151
DTD
   Attributes, 29
   Data, 28-29
   Subelements, 28
dynamic reconfiguration, overview, 26
dyngroups property, 212

E
Elements in the server.xml File, 29
enc parameter, 152, 154, 155, 205
encoding, chunked, 94-95
Error directive, function descriptions, 198-203
error logging, settings in magnus.conf, 89-90
ErrorLog directive, 226
ErrorLogDateFormat, magnus.conf directive, 89
ErrorLogDateFormat directive, 106
errors
   sending customized messages, 200, 201, 203
errors log, 65
escape parameter, 133
exexec-hack parameter, 155
exp parameter, 155
expire parameter, 99
extension parameter, 146
EXTERNALJNDIRESOUCRE, 58-59
ExtraPath, magnus.conf directive, 76
ExtraPath directive, 106
extras directory, 23

F
Favicon, magnus.conf directive, 96
file name extensions, MIME types, 205
file parameter, 144, 165, 167
FileCacheEnable parameter, 215
files, mapping types of, 205
filter parameter, 158, 159, 161, 162
FilterComps property, 210
find-index function, 139-140
find-links function, 140-141
find-pathinfo-forward parameter, 127, 132
find-pathinfo function, 141
flex-init function, 99
flex-log function, 196-197
flex-rotate-init function, 100
flexanlg directory, 23
flushTimer parameter, 163
force-type function, 152
format parameter, 98, 100
free-size parameter, 103
from parameter, 127, 132, 133, 149
funcs parameter, 102

G
gel-client-cert function, 142-143
get-sslid function, 119-120
group parameter, 181
groupdb parameter, 117
groupfile, 213
groupfn parameter, 117
grpfile parameter, 119

H
hard links, finding, 140
HashInitSize parameter, 216
header parameter, 171
HeaderBufferSize, magnus.conf directive, 81
HeaderBufferSize directive, 106
home-page function, 129-130
http-compression filter, 138
httpacl directory, 24
https-admserv directory, 24
https-server_id.domain, 24
HTTPVersion, magnus.conf directive, 96
HTTPVersion directive, 106
HUP signal, PidLog and, 90
Index

I
icon-uri parameter, 99
ignore parameter, 98
imagemap function, 169
include directory, 25
index-names parameter, 140
index-simple function, 171-172
inetOrgPerson, in convergence tree, 68
Init, function descriptions, 76
init-cgi function, 101
init-clf function, 101
Init functions, 97-104 222, 225-226
init-uhome function, 101
initArgs property, 222, 224
InitFn property, 211
Input, function descriptions, 157-159
inputStreamLengthCheck property, 223
insert-filter SAF, 158 160-161
iponly function, 196, 197
isModifiedCheckAggressive property, 223

J
J2SESecurityManager, 217
JAVA, 50-52
Java Configuration Elements, 50-56
JDBCConnectionPool, 59-63
JDBCResource, 59
JVMOptions, 52-53

K
KeepAliveQueryMaxSleepTime, magnus.conf directive, 81
KeepAliveQueryMeanTime, magnus.conf directive, 81
KeepAliveThreads directive, 107
KeepAliveTimeout
  magnus.conf directive, 82
KeepAliveTimeout directive, 107
KernelThreads, magnus.conf directive, 82
KernelThreads directive, 107
key-too-small function, 172-173
keyfile, 213

L
lang parameter, 152 154 155 205
language issues, directives in magnus.conf, 78
LDAP, iPlanet schema, 67-69
lib directory, 25
library property, 211
LICENSE.txt, 25
links, finding hard links, 140-141
list-dir function, 173-174
Listener Elements, 33-43
ListenQ, magnus.conf directive, 83
ListenerQ directive, 107
load-config function, 143-145
load-modules function, 102
loadbal directory, 25
LoadObjects directive, 220
LOG, 65-66
log analyzer, 195 196
log_anly directory, 23
log file
  analyzer for, 195 196
LogFlushInterval directive, 108
logging, settings in magnus.conf, 89-90
login.conf, 214
logs directory, 24
logs file, 26
logstderr, 66
logstdout, 66
logtoconsole, 66
LogVerbosedirective, 220 226
LS
  id, 34
  ip attribute, 34

M
magnus.conf
  changes to, 219-221 225-227
  miscellaneous directives, 95-96
MAILResource, 64-65
make-dir function, 175
manual directory, 25
match-browser function, 120-121
MaxAge parameter, 215
MaxCGIStubs, magnus.conf directive, 88
MaxCGIStubs directive, 108
MaxFiles parameter, 216
MaxKeepAliveConnections, magnus.conf directive, 83
MaxKeepAliveConnections directive, 108
MaxProcs, magnus.conf directive, 83
MaxProcs directive, 108
MaxRqHeaders, magnus.conf directive, 96
MaxRqHeaders directive, 108
MaxThreads directive, 220
maxthreads parameter, 104
MediumFileSizeLimit parameter, 216
MediumFileSpace parameter, 216
method parameter, 143 162
methods function, 103
mime.types file, 205
sample of, 207-208
syntax, 207
MinCGIStubs, magnus.conf directive, 88
MinCGIStubs directive, 108
MinProcs directive, 220
MinThreads directive, 220
minthreads parameter, 104
MMapSessionManager, 24
month name, 231
MtaHost directive, 220

N
name parameter, 127 132 135 196 197
of define-perf-bucket function, 99
of thread-pool-init function, 103
NameTrans, function descriptions, 126-135
native thread pools, settings in magnus.conf, 86-87
NativePoolMaxThreads, magnus.conf directive, 86
NativePoolMaxThreads directive, 108
NativePoolMinThreads, magnus.conf directive, 86
NativePoolMinThreads directive, 108
NativePoolQueueSize, magnus.conf directive, 87
NativePoolQueueSize directive, 108
NativePoolStackSize, magnus.conf directive, 86
NativePoolStackSize directive, 108
NativeThread parameter, 102
NetSiteRoot directive, 220 226

O
obj.conf
changes to, 221-222, 227
directives, 113-203
objectfile, 39
ObjectType, function descriptions, 151-157
opts parameter, 98
Output, function descriptions, 160-162
outputStreamFlushTimer property, 223

P
parameterEncoding property, 223
parse-html function, 222
path parameter, 130, 137, 139, 148, 176, 182, 201, 202
PathCheck, function descriptions, 136-151
pcheck-dav function, 147
perf-check function, 102
pfx2dir function, 131-133
PidLog, magnus.conf directive, 90
PidLog directive, 108
plug-ins directory, 25
pool-init function, 103
pool parameter, 102
Port directive, 221
Index

PostThreadsEarly, magnus.conf directive, 83
PostThreadsEarly directive, 109
predefined SAFs, 113-203
processes, settings in magnus.conf, 79-85
PROFILER, 53-54
profiling parameter, 103
pwfile parameter, 102, 135

Q
qos-error function, 200
qos-handler function, 121
qosactive, 31
qosmetricsinterval, 31
QOSPARAMS, 41-42
qosrecomputeinterval, 31
query-handler function, 176, 200-201
query parameter, 162
queueSize parameter, 104

R
RcvBufSize, magnus.conf directive, 84
RcvBufSize directive, 109
readme parameter, 171
realm parameter, 148
reason parameter, 201, 203
record-useragent function, 197-198
redirect function, 133-134
reloadInterval property, 223
remove-dir function, 176-177
remove-filter SAF, 159, 161-162
rename-file function, 179-180
require-auth function, 147-148
require parameter, 142
Resource Elements, 56-65
RESOURCES, 56-57
respondCookieVersion property, 222
restart file, 26
rlimit_as parameter, 181
rlimit_core parameter, 181
rlimit_nofile parameter, 181
root parameter, 129
rootobject, 39
RootObject directive, 221
rotate-access parameter, 100
rotate-callback parameter, 101
rotate-error parameter, 100
rotate-interval parameter, 100
rotate-start parameter, 100
RqThrottle, magnus.conf directive, 84
RqThrottle directive, 109
RqThrottleMin, magnus.conf directive, 84
RqThrottleMinPerSocket directive, 109
rules.properties, changes to, 223

S
SAFs
   alphabetical reference, 235-238
   Init, 76
   predefined, 113-203
samples directory, 25
SEARCH, 47
search directory, 24, 25
Search Elements, 46-48
SEARCHCOLLECTION, 47-48
secret-keysize parameter, 150
Security, magnus.conf directive, 91-92
SECURITY, 54-55
security, settings in magnus.conf, 91-93
Security directive, 109
send-cgi function, 180-182
send-error function, 182-183, 202-203
send-file function, 183-184
send-range function, 185
send-shellcgi function, 185-186
send-wincgi function, 186-187
server
   handling of authorization of client users, 116
   HUP signal, 90
   killing process of, 90
   TERM signal, 90
server information, magnus.conf directives, 76-78
server.policy, 217-218
server.xml, 27
server.xml (Continued)
more information, 156
variables defined in, 182
server.xml elements
ACLFILE, 37-38
AUTHREALM, 55-56
CONNECTIONPROPERTY, 63-64
CUSTOMRESOURCE, 57-58
DAV, 43-45
DAVCOLLECTION, 45-46
DESCRIPTION, 32
DISPLAYNAME, 48
EXTERNALJNDIRESOURCE, 58-59
JAVA, 50-52
JDBCCONNECTIONPOOL, 59-63
JDBCRESOURCE, 59
JVMOPTIONS, 52-53
LOG, 65-66
LS, 33-35
MAILRESOURCE, 64-65
MIME, 36-37
PROFILER, 53-54
PROPERTY, 31-32
QOSPARAMS, 41-42
RESOURCES, 56-57
SEARCH, 47
SEARCHCOLLECTION, 47-48
SECURITY, 54-55
SERVER, 30-31
SSLPARAMS, 35-36
USERDB, 42-43
VARS, 32
VS, 39-41
VSCLASS, 38-39
WEBAPP, 49-50
servercertnickname, 35
ServerName directive, 221
serverName property, 223
ServerRoot directive, 221
Service, function descriptions, 162-195
service-dav function, 187-188
service-dump function, 188-189
servlets directory, 25
servlets.properties, changes to, 224
SessionData, 24
SessionData directory, 24
SessionData file, 26
sessionmgr.initArgs property, 222
sessionmgr property, 222
set-default-type function, 153
set-variable function, 122-125
set-virtual-index function, 148-149
setup directory, 25
shlib parameter, 102
shtml-hacktype function, 154
shtml_send function, 191-192, 222
ShtmlMaxDepth parameter, 192
singleClassLoader property, 223
SmallFileSizeLimit parameter, 216
SmallFileSpace parameter, 216
SndBufSize, magnus.conf directive, 84
SndBufSize directive, 109
snmp directory, 25
SSL, settings in magnus.conf, 91-93
ssl-check function, 149-150
ssl-logout function, 150
ssl2, 36
SSL2 directive, 221
ssl2ciphers, 36
ssl3, 36
SSL3 directive, 221
SSL3Ciphers directive, 221
SSL3SessionTimeout, magnus.conf directive, 93
SSL3SessionTimeout directive, 109
ssl3tlsciphers, 36
SSLCacheEntries
magnus.conf directive, 92
SSLCacheEntries directive, 109
SSLClientAuth directive, 221
SSLClientAuthDataLimit, magnus.conf directive, 93
SSLClientAuthDataLimit directive, 109
SSLClientAuthTimeout, magnus.conf directive, 93
SSLClientAuthTimeout directive, 110
SSLSessionTimeout, magnus.conf directive, 93
SSLSessionTimeout directive, 110
StackSize, magnus.conf directive, 84
StackSize directive, 110
stackSize parameter, 104
Index

start file, 26
startconsole file, 25
startsvr.bat, 24
startup property, 224
statistic collection, settings in magnus.conf, 89-90
stats-init function, 103
stderr parameter, 102
stdout parameter, 102
stop file, 26
stopsvr.bat, 24
StrictHttpHeaders, magnus.conf directive, 85
StrictHttpHeaders directive, 110
strip-params function, 134
subdir parameter, 135
Sun Java System LDAP Schema, 67-69
sun-web-server_6_1.dtd, 27
symbolic links, finding, 140
syntax, 213
   mime.types file, 207
type-by-exp function, 155
type-by-extension, 206
type-by-extension function, 156-157
type parameter, 152, 155, 162, 205

U
Umask, magnus.conf directive, 96
Umask directive, 110
unix-home function, 135
unix-uri-clean function, 150
Unix user account, specifying, 77-78
update-interval parameter, 103
upload-file function, 194-195
uri parameter, 165, 167
uri property, 223
URL, mapping to other servers, 131-133
url parameter, 133
url-prefix parameter, 133
UseNativePoll, magnus.conf directive, 85
UseNativePoll directive, 111
UseOutputStreamSize
   magnus.conf directive, 94
   obj.conf Service parameter, 163
UseOutputStreamSize directive, 111
User, magnus.conf directive, 77-78
user account, specifying, 77-78
User Database Selection, 66-67
User directive, 111
user home directories, symbolic links and, 140
user parameter, 181
USERDB, 42-43, 66
userdb directory, 25
userdb parameter, 117
userfile, 213
userfile parameter, 118
userfn parameter, 117
usesyslog, 66
util_strftime, 231

V
Variable Evaluation, 70-71
vARIABLES
  General Variables, 70
Variables, send-cgi Variables, 70
vARIABLES
  send-cgi Variables, 70
verifycert property, 210
virtual-index parameter, 149
virtual server log, 65
virtual-servers parameter, 103
VirtualServerFile directive, 221
VS, 39-41
VSCLASS, 38-39
  id, 39

W
Web Application Elements, 49-50
WEBAPP, 49-50
WebDAV Elements, 43-46
weekday, 231
widths parameter, 98
WincgiTimeout, magnus.conf directive, 89
WincgiTimeout directive, 111