



Sun Java System Web Server 7.0 Update 3 Administrator's Configuration File Reference



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

Part No: 820-4841
June 2008

Copyright 2009 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more U.S. patents or pending patent applications in the U.S. and in other countries.

U.S. Government Rights – Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

This distribution may include materials developed by third parties.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, the Solaris logo, the Java Coffee Cup logo, docs.sun.com, JavaServer Pages, JDK, Java Naming and Directory Interface, JDBC, Java, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. or its subsidiaries in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. Netscape is a trademark or registered trademark of Netscape Communications Corporation in the United States and other countries.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

Products covered by and information contained in this publication are controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical or biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2009 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. Tous droits réservés.

Sun Microsystems, Inc. détient les droits de propriété intellectuelle relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et ce sans limitation, ces droits de propriété intellectuelle peuvent inclure un ou plusieurs brevets américains ou des applications de brevet en attente aux Etats-Unis et dans d'autres pays.

Cette distribution peut comprendre des composants développés par des tierces personnes.

Certains composants de ce produit peuvent être dérivées du logiciel Berkeley BSD, licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays; elle est licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, le logo Solaris, le logo Java Coffee Cup, docs.sun.com, JavaServer Pages, JDK, Java Naming and Directory Interface, JDBC, Java et Solaris sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc., ou ses filiales, aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc. Netscape est une marque de Netscape Communications Corporation aux Etats-Unis et dans d'autres pays.

L'interface d'utilisation graphique OPEN LOOK et Sun a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui, en outre, se conforment aux licences écrites de Sun.

Les produits qui font l'objet de cette publication et les informations qu'il contient sont régis par la législation américaine en matière de contrôle des exportations et peuvent être soumis au droit d'autres pays dans le domaine des exportations et importations. Les utilisations finales, ou utilisateurs finaux, pour des armes nucléaires, des missiles, des armes chimiques ou biologiques ou pour le nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou réexportations vers des pays sous embargo des Etats-Unis, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière non exclusive, la liste de personnes qui font objet d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui sont régis par la législation américaine en matière de contrôle des exportations et la liste de ressortissants spécifiquement désignés, sont rigoureusement interdites.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFACON.

Contents

Preface	17
1 Overview of Configuration Files and Directories	25
Configuration Files	25
The server.xml File	26
The magnus.conf File	26
The obj.conf File	26
The mime.types File	26
ACL Files	26
Other Configuration Files	26
Directory Structure	27
admin-server	27
bin	28
https-server_id	28
include	29
jdk	29
lib	29
plugins	29
samples	29
setup	29
Dynamic Reconfiguration	30
2 Syntax and Use of server.xml	33
Overview of server.xml	33
sun-web-server_7_0.xsd	33
Editing server.xml	33
Understanding server.xml	35

Access Control	35
Clustering	35
HTTP Protocol	36
Java	36
Logging and Monitoring	37
Performance Tuning	37
Search	37
SSL, TLS, and PKCS #11	37
Variables	38
Virtual Servers	38
WebDAV	38
Sample server.xml File	38
3 Elements in server.xml	41
List of Elements	41
access-log	43
access-log-buffer	43
acl-cache	44
acl-db	45
acl-file	46
audit-accesses	46
auth	47
auth-db	47
auth-realm	48
cgi	49
cluster	50
connection-creation-property	50
connection-lease-property	51
convert	52
custom-resource	52
dav	53
dav-collection	54
default-auth-db-name	55
default-auth-realm-name	56
default-soap-auth-provider-name	56

display-name	56
dns	56
dns-cache	57
env-variable	58
event	58
external-jndi-resource	59
file-cache	60
http	61
http-listener	63
include	64
index	65
instance	65
jdbc-resource	66
jvm	67
keep-alive	69
lifecycle-module	69
localization	70
lock-db	71
log	71
mail-resource	73
mime-file	74
pkcs11	74
profiler	74
property	75
property-db	76
qos	77
qos-limits	77
request-policy	78
response-policy	78
search-app	79
search-collection	79
server	80
servlet-container	83
session-replication	85
single-sign-on	86
snmp	87

soap-auth-provider	87
ssl	88
ssl2-ciphers	89
ssl3-tls-ciphers	90
ssl-session-cache	93
stats	94
thread-pool	95
time	95
token	96
variable	96
virtual-server	97
web-app	99
4 Syntax and Use of magnus.conf	101
Editing magnus.conf	101
Parameters	102
Case Sensitivity	102
Separators	102
Quotation Marks	102
Spaces	102
Line Continuation	102
Path Names	102
Comments	103
ChildRestartCallback	103
Syntax	103
Init Directives	103
Syntax	103
KernelThreads	104
Syntax	104
Default	104
NativePoolMaxThreads	104
Default	104
NativePoolMinThreads	104
Default	104
NativePoolQueueSize	105

Default	105
NativePoolStackSize	105
Default	105
TerminateTimeout	105
Syntax	105
Default	105
Umask	106
Syntax	106
Default	106
5 Predefined SAFs in magnus.conf	107
Init SAFs	107
cindex-init	108
define-perf-bucket	109
init-dav	110
init-filter-order	110
init-request-limits	112
init-uhome	112
load-modules	113
pool-init	114
register-http-method	114
thread-pool-init	115
Common SAFs	116
Deprecated Init SAFs	117
6 Syntax and Use of obj.conf	119
Request-Handling Process Overview	120
Stages in the Request-Handling Process	120
Directives in obj.conf	121
Objects in obj.conf	122
Objects That Use the name Attribute	123
Objects That Use the ppath Attribute	123
Using the Client, If, Elseif, and Else Tags	124
Flow of Control in obj.conf	127
AuthTrans	127

NameTrans	127
PathCheck	129
ObjectType	129
Input	131
Output	131
Route	132
Service	132
AddLog	134
Error	134
Changes in Function Flow	134
Restarted Requests	134
Internal Requests	135
URI Translation	135
Editing obj.conf	135
Order of Directives	135
Parameters	135
Case Sensitivity	136
Separators	136
Quotation Marks	136
Spaces	136
Line Continuation	136
Path Names	136
Comments	136
7 Predefined SAFs and Filters in obj.conf	137
The bucket Parameter	138
AuthTrans	138
basic-auth	138
basic-nca	140
get-sslid	141
qos-handler	141
NameTrans	142
assign-name	143
document-root	144
home-page	145

map	146
ntrans-dav	147
ntrans-j2ee	148
pfx2dir	148
reverse-map	150
rewrite	151
strip-params	152
unix-home	152
PathCheck	153
check-acl	154
check-request-limits	155
deny-existence	157
find-compressed	158
find-index	159
find-index-j2ee	160
find-links	161
find-pathinfo	162
get-client-cert	163
nt-uri-clean	164
ntcgicheck	165
pcheck-dav	166
require-auth	166
set-virtual-index	167
ssl-check	168
ssl-logout	169
unix-uri-clean	169
ObjectType	170
block-auth-cert	171
block-cache-info	172
block-cipher	172
block-ip	173
block-issuer-dn	173
block-jroute	174
block-keysize	174
block-proxy-agent	175
block-proxy-auth	176

block-secret-keysize	176
block-ssl-id	177
block-user-dn	177
block-via	178
force-type	178
forward-auth-cert	179
forward-cache-info	180
forward-cipher	180
forward-ip	181
forward-issuer-dn	181
forward-jroute	182
forward-keysize	183
forward-proxy-agent	183
forward-proxy-auth	184
forward-secret-keysize	184
forward-ssl-id	185
forward-user-dn	185
forward-via	186
http-client-config	186
set-basic-auth	187
set-cache-control	188
set-cookie	189
set-default-type	190
shtml-hacktype	190
ssl-client-config	191
type-by-exp	192
type-by-extension	193
type-j2ee	194
Input	194
sed-request	195
Output	196
compress-file	196
http-compression	198
sed-response	200
Route	200
set-origin-server	201

set-proxy-server	202
Service	203
add-footer	205
add-header	206
append-trailer	208
delete-file	209
imagemap	210
index-common	211
index-simple	214
key-toosmall	215
list-dir	216
make-dir	218
proxy-retrieve	219
remove-dir	220
rename-file	221
send-cgi	222
send-file	225
send-range	226
send-shellcgi	227
send-wincgi	228
service-dav	229
service-dump	231
service-j2ee	232
service-trace	234
shtml-send	235
stats-xml	236
upload-file	238
AddLog	239
flex-log	239
Error	240
error-j2ee	240
qos-error	241
Common SAFs	242
insert-filter	243
match-browser	244
query-handler	245

redirect	246
remove-filter	248
restart	249
send-error	250
set-variable	252
FastCGI Plug-in SAFs	256
8 MIMETypes	257
Determining the MIME Type	257
Referencing MIME Types Files in server.xml	258
Generating the Server Response Using the MIME Type	258
Processing the Response in the Client Using the MIME Type	259
MIME Types Syntax	259
Sample MIME Types File	259
9 ACL Files	265
Referencing ACL Files in server.xml and obj.conf	265
ACL File Syntax	266
General Syntax	267
Authentication Methods	267
Authorization Statements	268
Hierarchy of Authorization Statements	269
Sample ACL File	271
10 Other Server Configuration Files	273
certmap.conf	273
Location	274
Syntax	274
See Also	275
sun-web.xml	275
Location	275
login.conf	275
Location	275
server.policy	275

Location	276
Syntax	276
See Also	276
default-web.xml	276
Location	276
See Also	276
A Using Variables, Expressions, and String Interpolation	277
Variables	277
Predefined Variables	277
Custom Variables	280
Resolving Variables	280
Expressions	281
Expression Syntax	281
Expression Results as Booleans	282
Expression Literals	282
Expression Variables	283
Expression Operators	284
Expression Functions	286
Regular Expressions	295
String Interpolation	296
Using Variables in Interpolated Strings	297
Using Expressions in Interpolated Strings	297
B Using Wildcard Patterns	299
Wildcard Patterns	299
Wildcard Examples	300
C Using the Custom Log File Format	301
Custom Log File Format	301
D Using Time Formats	305
Format Strings	305

E	Configuration Changes Between Sun ONE Web Server 6.1 and Sun Java System Web Server 7.0	307
	Element Changes in server.xml	307
	Directive and Init Function Changes in magnus.conf	309
	Directive Changes	309
	Init Function Changes	313
	Other Configuration File Changes	314
F	Web Server Interfaces	315
G	Alphabetical List of Server Configuration Elements and Predefined SAFs	321
	Index	329

Examples

EXAMPLE 2-1	server.xml File	38
-------------	-----------------------	----

Preface

The *Sun Java System Web Server Administrator's Configuration File Reference* discusses the purpose and use of the configuration files for Sun Java System Web Server (Web Server), including `server.xml`, `magnus.conf`, `obj.conf`, and `mime.types`. This document provides a comprehensive list of the elements and directives in these configuration files.

Who Should Use This Book

The intended audience for this document is the person who administers and maintains Web Server.

This document assumes you are familiar with the following topics:

- Java Platform, Enterprise Edition (Java EE)
- HTTP
- HTML
- XML
- Relational database concepts

Before You Read This Book

Sun Java System Web Server can be installed as a stand-alone product or as a component of Sun Java™ Enterprise System (Java ES), a software infrastructure that supports enterprise applications distributed across a network or Internet environment. If you are installing Sun Java System Web Server as a component of Java ES, you should be familiar with the system documentation at <http://docs.sun.com/coll/1286.2>.

Sun Java System Web Server Documentation Set

The Sun Java System Web Server documentation set describes how to install and administer the Web Server. The URL for Sun Java System Web Server documentation is <http://docs.sun.com/coll/1653.2>. For an introduction to Sun Java System Web Server, refer to the books in the order in which they are listed in the following table.

TABLE P-1 Books in the Sun Java System Web Server Documentation Set

Documentation Title	Contents
<i>Sun Java System Web Server 7.0 Update 3 Documentation Center</i>	Web Server documentation topics organized by tasks and subject
<i>Sun Java System Web Server 7.0 Update 3 Release Notes</i>	<ul style="list-style-type: none"> ■ Late-breaking information about the software and documentation ■ Supported platforms and patch requirements for installing Web Server
<i>Sun Java System Web Server 7.0 Update 3 Installation and Migration Guide</i>	Performing installation and migration tasks: <ul style="list-style-type: none"> ■ Installing Web Server and its various components ■ Migrating data from Sun ONE Web Server 6.0 or 6.1 to Sun Java System Web Server 7.0
<i>Sun Java System Web Server 7.0 Update 3 Administrator's Guide</i>	Performing the following administration tasks: <ul style="list-style-type: none"> ■ Using the Administration and command-line interfaces ■ Configuring server preferences ■ Using server instances ■ Monitoring and logging server activity ■ Using certificates and public key cryptography to secure the server ■ Configuring access control to secure the server ■ Using Java Platform Enterprise Edition (Java EE) security features ■ Deploying applications ■ Managing virtual servers ■ Defining server workload and sizing the system to meet performance needs ■ Searching the contents and attributes of server documents, and creating a text search interface ■ Configuring the server for content compression ■ Configuring the server for web publishing and content authoring using WebDAV

TABLE P-1 Books in the Sun Java System Web Server Documentation Set (Continued)

Documentation Title	Contents
<i>Sun Java System Web Server 7.0 Update 3 Developer's Guide</i>	Using programming technologies and APIs to do the following: <ul style="list-style-type: none"> ■ Extend and modify Sun Java System Web Server ■ Dynamically generate content in response to client requests and modify the content of the server
<i>Sun Java System Web Server 7.0 Update 3 NSAPI Developer's Guide</i>	Creating custom Netscape Server Application Programmer's Interface (NSAPI) plug-ins
<i>Sun Java System Web Server 7.0 Update 3 Developer's Guide to Java Web Applications</i>	Implementing Java Servlets and JavaServer Pages™ (JSP™) technology in Sun Java System Web Server
<i>Sun Java System Web Server 7.0 Update 3 Administrator's Configuration File Reference</i>	Editing configuration files
<i>Sun Java System Web Server 7.0 Update 3 Performance Tuning, Sizing, and Scaling Guide</i>	Tuning Sun Java System Web Server to optimize performance
<i>Sun Java System Web Server 7.0 Update 3 Troubleshooting Guide</i>	Troubleshooting Web Server

Related Books

The URL for all documentation about Sun Java Enterprise System (Java ES) and its components is <http://docs.sun.com/coll/1286.2>.

Default Paths and File Names

The following table describes the default paths and file names that are used in this book.

TABLE P-2 Default Paths and File Names

Placeholder	Description	Default Value
<i>install_dir</i>	Represents the base installation directory for Sun Java System Web Server.	<p>Sun Java Enterprise System (Java ES) installations on the Solaris™ platform:</p> <p><i>/opt/SUNWwbsvr7</i></p> <p>Java ES installations on the Linux and HP-UX platform:</p> <p><i>/opt/sun/webserver/</i></p> <p>Java ES installations on the Windows platform:</p> <p><i>System Drive:\Program Files\Sun\JavaES5\WebServer7</i></p> <p>Other Solaris, Linux, and HP-UX installations, non-root user:</p> <p><i>user's home directory/sun/webserver7</i></p> <p>Other Solaris, Linux, and HP-UX installations, root user:</p> <p><i>/sun/webserver7</i></p> <p>Windows, all installations:</p> <p><i>System Drive:\Program Files\Sun\WebServer7</i></p>
<i>instance_dir</i>	Directory that contains the instance-specific subdirectories.	<p>For Java ES installations, the default location for instances on Solaris:</p> <p><i>/var/opt/SUNWwbsvr7</i></p> <p>For Java ES installations, the default location for instances on Linux and HP-UX:</p> <p><i>/var/opt/sun/webserver7</i></p> <p>For Java ES installations, the default location for instance on Windows:</p> <p><i>System Drive:\Program Files\Sun\JavaES5\WebServer7</i></p> <p>For stand-alone installations, the default location for instance on Solaris, Linux, and HP-UX:</p> <p><i>install_dir</i></p> <p>For stand-alone installations, the default location for instance on Windows:</p> <p><i>System Drive:\Program Files\sun\WebServer7</i></p>

Typographic Conventions

The following table describes the typographic changes that are used in this book.

TABLE P-3 Typographic Conventions

Typeface	Meaning	Example
AaBbCc123	The names of commands, files, and directories, and onscreen computer output	Edit your <code>.login</code> file. Use <code>ls -a</code> to list all files. <code>machine_name%</code> you have mail.
AaBbCc123	What you type, contrasted with onscreen computer output	<code>machine_name%</code> su Password:
<i>AaBbCc123</i>	A placeholder to be replaced with a real name or value	The command to remove a file is <code>rm filename</code> .
<i>AaBbCc123</i>	Book titles, new terms, and terms to be emphasized (note that some emphasized items appear bold online)	Read Chapter 6 in the <i>User's Guide</i> . <i>A cache</i> is a copy that is stored locally. Do <i>not</i> save the file.

Symbol Conventions

The following table explains symbols that might be used in this book.

TABLE P-4 Symbol Conventions

Symbol	Description	Example	Meaning
[]	Contains optional arguments and command options.	<code>ls [-l]</code>	The <code>-l</code> option is not required.
{ }	Contains a set of choices for a required command option.	<code>-d {y n}</code>	The <code>-d</code> option requires that you use either the <code>y</code> argument or the <code>n</code> argument.
`\${ }`	Indicates a variable reference.	<code>\${com.sun.javaRoot}</code>	References the value of the <code>com.sun.javaRoot</code> variable.
-	Joins simultaneous multiple keystrokes.	Control-A	Press the Control key while you press the A key.
+	Joins consecutive multiple keystrokes.	Ctrl+A+N	Press the Control key, release it, and then press the subsequent keys.

TABLE P-4 Symbol Conventions (Continued)

Symbol	Description	Example	Meaning
→	Indicates menu item selection in a graphical user interface.	File → New → Templates	From the File menu, choose New. From the New submenu, choose Templates.

Accessing Sun Resources Online

The <http://docs.sun.com> (docs.sun.comSM) web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. Books are available as online files in PDF and HTML formats. Both formats are readable by assistive technologies for users with disabilities.

To access the following Sun resources, go to <http://www.sun.com>:

- Downloads of Sun products
- Services and solutions
- Support (including patches and updates)
- Training
- Research
- Communities (for example, Sun Developer Network)

Searching Sun Product Documentation

Besides searching Sun product documentation from the docs.sun.com web site, you can use a search engine by typing the following syntax in the search field:

```
search-term site:docs.sun.com
```

For example, to search for “Web Server,” type the following:

```
Web Server site:docs.sun.com
```

To include other Sun web sites in your search (for example, java.sun.com, www.sun.com, and developers.sun.com), use “sun.com” in place of “docs.sun.com” in the search field.

Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

Note – Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. To share your comments, go to <http://docs.sun.com> and click Send Comments. In the online form, provide the full document title and part number. The part number is a 7-digit or 9-digit number that can be found on the book's title page or in the document's URL. For example, the part number of this book is 820-2203.

Overview of Configuration Files and Directories

The configuration and behavior of Sun Java System Web Server 7.0 (Web Server) is determined by a set of configuration files. You can use the Admin Console and the command-line interface (CLI) to change the configuration file settings. You can also manually edit these files.

This chapter has the following sections:

- “Configuration Files” on page 25
- “Directory Structure” on page 27
- “Dynamic Reconfiguration” on page 30

Configuration Files

Each server instance has its own directory, called *instance_dir* in this document. The *instance_dir/config* directory contains configuration files for the Web Server components. The exact number and names of the configuration files depend on the components that have been enabled or loaded into the server. For the default location of the *instance_dir*, see “[Default Paths and File Names](#)” on page 19.

These files, as well as some other configuration files not included in the *config* directory, are described in the following sections:

- “The *server.xml* File” on page 26
- “The *magnus.conf* File” on page 26
- “The *obj.conf* File” on page 26
- “The *mime.types* File” on page 26
- “ACL Files” on page 26
- “Other Configuration Files” on page 26

The server.xml File

The `server.xml` file contains most of the server configuration. A schema file, `sun-web-server_7_0.xsd`, validates its format and content. For more information about `sun-web-server_7_0.xsd` and the various elements of `server.xml`, see [Chapter 2, “Syntax and Use of server.xml,”](#) and [Chapter 3, “Elements in server.xml.”](#)

The magnus.conf File

The `magnus.conf` file contains the NSAPI plug-in initialization directives and settings that control the way NSAPI plug-ins are run. For more information about `magnus.conf`, see [Chapter 4, “Syntax and Use of magnus.conf,”](#) and [Chapter 5, “Predefined SAFs in magnus.conf.”](#)

The obj.conf File

The `obj.conf` file contains directives for HTTP request processing. For more information about `obj.conf`, see [Chapter 6, “Syntax and Use of obj.conf,”](#) and [Chapter 7, “Predefined SAFs and Filters in obj.conf.”](#)

The mime.types File

The `mime.types` file maps file extensions to MIME types to enable the server to determine the content type of a requested resource. For example, requests for resources with `.html` extensions indicate that the client is requesting an HTML file, while requests for resources with `.gif` extensions indicate that the client is requesting an image file in GIF format. For more information about `mime.types`, see [Chapter 8, “MIME Types.”](#)

ACL Files

The Access Control List (ACL) files contain lists that define who can access resources stored on your Web Server. By default, Web Server uses one ACL file. You can create multiple ACL files and reference them in the `obj.conf` and `server.xml` files. For more information about ACL files, see [Chapter 9, “ACL Files.”](#)

Other Configuration Files

Other configuration files for administration and for applications include the `certmap.conf`, `sun-web.xml`, `login.conf`, `server.policy`, and `default-web.xml`. For more information on these files, see [Chapter 10, “Other Server Configuration Files.”](#)

Directory Structure

This section describes the directory structure that is created when you first install Sun Java System Web Server. In a stand-alone Web Server installation, all these directories are in the *install_dir* by default. In Web Servers installed as part of Java Enterprise System, the instance directories (which in this case includes both *admin-server* and *https-server_id*) are in a different location. For more information on the default locations for these directories on different platforms, see the information on *instance_dir* in “Default Paths and File Names” on page 19.

- “admin-server” on page 27
- “bin” on page 28
- “https-server_id” on page 28
- “include” on page 29
- “jdk” on page 29
- “lib” on page 29
- “plugins” on page 29
- “samples” on page 29
- “setup” on page 29

admin-server

The *admin-server* directory has the following subdirectories:

- *bin* – Contains the binary files that are required to start, stop, and restart Web Server. On UNIX systems, this directory also contains the file required for rotating logs.
- *config* – Contains the private configuration files for the Administration Server. These files are for internal use.
- *config-store* – Contains files used by the Administration Server to track server configuration information.

Note – The files in this directory are created by Web Server for internal use. Do not edit, run scripts on, or otherwise access any files in the *config-store* directory.

- *generated* – Contains files generated by the instance, such as Java class files corresponding to JavaServer Pages™ (JSP™).
- *logs* – Contains any error or access log files that are generated by a server instance.
- *sessions* – Contains session data files.

bin

The `bin` directory contains the binary files for administering Web Server. These files include `wadm`, the administration command-line interface (CLI).

https-server_id

An `https-server_id` directory is created for every instance you create. This directory has the following subdirectories and files:

- `bin` – Contains the binary files for starting, stopping, restarting, and reconfiguring the server. On UNIX, it also contains the binary file for rotating the log files.
- `collections` – Contains the search collection data.
- `config` – Contains the following instance-specific configuration files:
 - `cert8.db` – NSS certificate database
 - `certmap.conf` – Certificate to LDAP DN mapping configuration
 - `default.acl` – Default ACL file for the server instance named *server_id*
 - `default-web.xml` – Default values for all web applications
 - `key3.db` – NSS private key database
 - `keyfile` – The usernames and hashed passwords for flat file authentication
 - `login.conf` – Information for file authentication used by the Java Authentication and Authorization Service (JAAS)
 - `magnus.conf` – Information for NSAPI plug-in initialization and operation
 - `mime.types` – Map of file extensions to MIME types that enables the server to determine the content type of a requested resource
 - `obj.conf` – Instructions for Web Server for handling HTTP requests from clients
 - `secmod.db` – NSS PKCS #11 module database
 - `server.policy` – Policy controlling the access that applications have to resources
 - `server.xml` – Most of the server configuration settings
- `docs` – Contains the files in the default document root for an instance.
- `generated` – Contains files generated by the instance.
- `lock-db` – The default directory for WebDAV lock database files.
- `logs` – Contains log files generated by a server instance.
- `sessions` – Contains session data files.
- `web-app` – The default directory for deployed web applications.

include

The `include` directory contains the various include files, for example, NSAPI and SHTML include files.

jdk

The `jdk` directory contains the bundled Java development kit (JDK™). For stand-alone installations only.

lib

The `lib` directory contains internal binaries, scripts, libraries, and bundled plug-ins. These files are private files, for internal use only.

plugins

The `plugins` directory contains the plug-in subdirectories. .

- `digest` contains the digest authentication plug-in for Sun Java Directory Server, as well as information about the plug-in.
- `fastcgi` contains the files for the FastCGI plug-in.
- `htaccess` contains server plug-in for `.htaccess` access control and `htconvert`, which is an `.nsconfig` to `.htaccess` converter.
- `loadbal` contains the required files for the third-party load-balancer integration plug-in.

For more information on these plug-ins, see [Sun Java System Web Server 7.0 Update 3 Administrator's Guide](#).

samples

The `samples` directory 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 environment.

setup

The `setup` directory contains the various Web Server setup files, including the installation logs.

Dynamic Reconfiguration

Dynamic reconfiguration allows you to make configuration changes to a runtime Web Server. You do not have to stop or restart the Web Server for the changes to take effect.

Dynamic configuration happens in one of the following ways:

- When you deploy a configuration through the Admin Console or CLI
- When you run the `reconfig` script in the server instance's `bin` directory

You can dynamically change the configuration settings in the `obj.conf`, `mime.types`, and ACL files without restarting the server. In addition, most settings in the `server.xml` file can be changed without restarting the server. If a server restart is required, a warning message appears in the server log when you deploy the configuration or run the `reconfig` command.

You cannot dynamically reconfigure the following `server.xml` configuration parameters:

- `user`
- `temp-path`
- `log` (with the exception of `log-level`)
- `thread-pool`
- `pkcs11`
- `stats`
- `cgi`
- `dns`
- `dns-cache`
- `file-cache`
- `acl-cache`
- `ssl-session-cache`
- `access-log-buffer`
- `jvm` (with exception of `log-level`)

When you run the `reconfig` command, a new configuration object is created and all new incoming requests are processed based on this new configuration object. The current configuration object gets removed when no HTTP requests are using the object. For example, if you are using Web Server with reverse proxy in the front-end can dynamically add a new back-end server and apply the reconfiguration by using the `reconfig` command.

The following additional configuration changes within `server.xml` are supported by the `reconfig` command:

- Add a new HTTP listener
- Add a new Java web application

Note – If the web application requires changes to the JVM or require a new JDBC resource, then you must restart the server instead of using the `reconfig` command.

Note – Some configuration changes cannot be instantly propagated to the running server. For example, adding a JVM property of a resource.

If a misconfiguration occurs during dynamic reconfiguration, the server displays an error message. The server logs the error message to a log file specified by the previous known good configuration.

Certain misconfigurations result in warning messages but do not cause the server to reject the configuration. Other misconfigurations result in error messages and cause the server to reject the configuration. If the server rejects a configuration during startup, the server does not start. If the server rejects a configuration during dynamic reconfiguration, the server reverts to the previous known good configuration.

Syntax and Use of server.xml

The `server.xml` file contains most of the server configuration. This chapter describes the basic syntax of `server.xml` and gives a high-level view of the elements that configure server features. It contains the following sections:

- “Overview of `server.xml`” on page 33
- “Understanding `server.xml`” on page 35
- “Sample `server.xml` File” on page 38

Overview of server.xml

The `server.xml` file contains most of the configuration information needed to run the server. The `server.xml` file is located in the `instance_dir/config` directory. The encoding is UTF-8 to maintain compatibility with regular UNIX text editors.

sun-web-server_7_0.xsd

The `sun-web-server_7_0.xsd` schema validates the format and content of the `server.xml` file. The schema enforces type checks. For example, it ensures that the `ip` element specifies a valid IP address. The `sun-web-server_7_0.xsd` file is located in the `install_dir/lib/dtds` directory.

Editing server.xml

The structure of `server.xml` is a hierarchy, with `server` as the topmost element. The `server` element has many subelements, many of which have subelements of their own.

In general, you do not need to edit `server.xml` directly. Instead, use the Admin Console and the `wadm` command-line interface to change values in `server.xml`. Using `wadm` when creating scripts to change `server.xml` assures you of forward compatibility. If you do edit `server.xml` directly, exercise caution to make sure that the resulting `server.xml` file is valid.

Editing Element Values

To change the values in `server.xml`, change the value between the tags associated with the element you are editing. For example, to change the log level from `info` to `fine`, find the `log` child element of the `server` element. In this example, you see the following lines:

```
<log>
  <log-file>../logs/errors</log-file>
  <log-level>info</log-level>
</log>
```

To change the log level from `info` to `fine`, change the line:

```
<log-level>info</log-level>
```

to:

```
<log-level>fine</log-level>
```

After you make changes to the `server.xml` file, you must deploy your configuration for most changes to take effect. Use the command-line interface command `wadm pull-config` to pull the modified `server.xml` file, then use the Admin Console or the `wadm deploy-config` command to deploy your changes. For some changes, you must restart the server before they take effect. For information on which changes require a restart and which do not, see [“Dynamic Reconfiguration” on page 30](#).

Adding Elements

To add a new element to `server.xml`, add the element and any required subelements. Elements begin with a tag, for example `<virtual-server>`, and end with the closing tag, for example `</virtual-server>`. The tags are case-sensitive.

Validating server.xml

After editing `server.xml`, Web Server automatically validates the XML when you start or dynamically reconfigure a the server.

You can also use the `-configtest` option of the `startserv` script to validate your configuration. From the instance's `bin` directory, run:

```
startserv -configtest
```

Understanding server.xml

To edit `server.xml` for your environment, you must know which elements contain the relevant settings. The following sections contain brief descriptions of the elements that configure the functional areas:

- “Access Control” on page 35
- “Clustering” on page 35
- “HTTP Protocol” on page 36
- “Java” on page 36
- “Logging and Monitoring” on page 37
- “Performance Tuning” on page 37
- “Search” on page 37
- “SSL, TLS, and PKCS #11” on page 37
- “Variables” on page 38
- “Virtual Servers” on page 38
- “WebDAV” on page 38

In addition, [Chapter 3, “Elements in server.xml,”](#) contains an alphabetical list of all the `server.xml` elements and their subelements.

Access Control

The `acl-file` element references an ACL file. ACL files define the authorization rules. The `auth-realm` and `default-auth-realm-name` elements configure authentication realms for Java Servlet container authentication. For more information, see [“acl-file” on page 46](#), [“auth-realm” on page 48](#), and [“default-auth-realm-name” on page 56](#).

The `auth-db` and `default-auth-db-name` elements configure the authentication databases for server authentication. Authentication databases are used with ACL files. For more information, see [“auth-db” on page 47](#), and [“default-auth-db-name” on page 55](#).

For more information on ACL files, see [Chapter 9, “ACL Files.”](#)

Clustering

The `cluster` element defines a cluster of servers to which an individual server instance belongs. The `instance` element defines an individual member of a cluster. The `session-replication` element configures how Java Servlet sessions are shared between instances in a cluster. For more information, see [“cluster” on page 50](#), [“instance” on page 65](#), and [“session-replication” on page 85](#).

HTTP Protocol

The `http` element configures the general HTTP protocol options. The `keep-alive` element configures HTTP keep-alive connection management. The `http-listener` element configures the ports and IP addresses on which the server listens for new HTTP connections. The `virtual-server` element configures how the server processes the HTTP requests. For more information, see [“http” on page 61](#), [“keep-alive” on page 69](#), [“http-listener” on page 63](#), and [“virtual-server” on page 97](#).

Java

The following elements configure the Java Servlet container:

- The `servlet-container` element configures miscellaneous Servlet container options. For more information, see [“servlet-container” on page 83](#).
- The `auth-realm` element defines an authentication realm for Java Servlet container authentication. For more information, see [“auth-realm” on page 48](#).
- The `default-auth-realm-name` element specifies the default authentication realm for Java Servlet container authentication. For more information, see [“default-auth-realm-name” on page 56](#).
- The `single-sign-on` element determines how the authentication information is shared across multiple Java web applications. For more information, see [“single-sign-on” on page 86](#).
- The `web-app` element defines the location of a Java web application. For more information, see [“web-app” on page 99](#).

The following elements configure the Java Naming and Directory Interface™ (JNDI) resources:

- The `custom-resource` element defines a resource implemented by a custom Java class. For more information, see [“custom-resource” on page 52](#).
- The `external-jndi-resource` element identifies the resource provided by an external JNDI repository. For more information, see [“external-jndi-resource” on page 59](#).
- The `jdbc-resource` element configures a Java Database Connectivity (JDBC™) data source. For more information, see [“jdbc-resource” on page 66](#).
- The `mail-resource` element configures a mail store resource. For more information, see [“mail-resource” on page 73](#).

The `lifecycle-module` element loads the custom Java plug-ins that are triggered by one or more events in the server's lifecycle. For more information [“lifecycle-module” on page 69](#).

The `soap-auth-provider` element configures message-level authentication for Java web services. For more information, see [“soap-auth-provider” on page 87](#).

The `jvm` element configures the Java Virtual Machine (JVM). For more information, see [“jvm” on page 67](#).

Logging and Monitoring

The `access-log` element configures the file name and formats of access logs. The `access-log-buffer` element configures the frequency of access log updates and ordering of the access log entries. For more information, see [“access-log” on page 43](#) and [“access-log-buffer” on page 43](#).

The `log` element configures the file name and contents of the server log. The `event` element configures the access log and server log rotation. For more information, see [“log” on page 71](#) and [“event” on page 58](#).

The `snmp` element configures SNMP, and the `stats` element configures statistics collection. For more information, see [“snmp” on page 87](#) and [“stats” on page 94](#).

Performance Tuning

The `thread-pool` element configures the number of threads used to process requests and the maximum number of HTTP connections that the server queues. For more information, see [“thread-pool” on page 95](#).

The `keep-alive` element configures the HTTP keep-alive connection management. For more information, see [“keep-alive” on page 69](#).

WebDAV ACL, lock, and property caching are controlled by the `acl-db`, `lock-db`, and `property-db` elements, respectively. For more information, see [“acl-db” on page 45](#), [“lock-db” on page 71](#), and [“property-db” on page 76](#).

The `file-cache` element configures file caching. The `dns-cache` element configures the DNS caching. The `acl-cache` element configures the authentication credential caching. For more information, see [“file-cache” on page 60](#), [“dns-cache” on page 57](#), and [“acl-cache” on page 44](#).

Search

The `search-collection` element defines the set of documents that the server should index. The `search-app` element configures the server's built-in search web application. For more information, see [“search-collection” on page 79](#) and [“search-app” on page 79](#).

SSL, TLS, and PKCS #11

The `ssl` element configures SSL and TLS. SSL and TLS can be configured separately for each HTTP listener. For more information, see [“ssl” on page 88](#) and [“http-listener” on page 63](#).

The `pkcs11` element configures the PKCS #11 subsystem, including Certificate Revocation Lists (CRLs) and third-party cryptographic modules. For more information, see [“pkcs11” on page 74](#).

Variables

The `variable` element defines a variable for use in expressions, log formats, and `obj.conf` parameters. For more information on the `variable` element, see [“variable” on page 96](#). For more information on variable and expression use, see [Appendix A, “Using Variables, Expressions, and String Interpolation.”](#) For more information on the log file format, see [Appendix C, “Using the Custom Log File Format.”](#)

Virtual Servers

The `virtual-server` element configures the virtual servers. Each virtual server accepts HTTP connections from one or more HTTP listeners. The `http-listener` element configures the HTTP listeners. For more information, see [“virtual-server” on page 97](#), and [“http-listener” on page 63](#).

You can define variables within a virtual server using the `variable` element, as described in the previous section, [“Variables” on page 38](#).

WebDAV

The `dav` element configures WebDAV. The `dav-collection` element defines the set of files that are accessible through WebDAV. For more information, see [“dav” on page 53](#) and [“dav-collection” on page 54](#).

Sample server.xml File

The following example shows a `server.xml` file.

EXAMPLE 2-1 `server.xml` File

```
?xml version="1.0" encoding="UTF-8"?>

<!--
  Copyright 2006 Sun Microsystems, Inc. All rights reserved.
  Use is subject to license terms.
-->
```

EXAMPLE 2-1 server.xml File (Continued)

```

<server>
  <cluster>
    <local-host>sun1</local-host>
    <instance>
      <host>sun1</host>
    </instance>
  </cluster>

  <log>
    <log-file>../logs/errors</log-file>
    <log-level>info</log-level>
  </log>

  <temp-path>/tmp/https-sun1-5351d5c9-2</temp-path>

  <user>myuser/user>

  <jvm>
    <java-home>/opt/webserver7/jdk</java-home>
    <server-class-path>/opt/webserver7/lib/webserv-rt.jar:/opt/webserver7/lib/pw
c.jar:/opt/webserver7/lib/ant.jar:${java.home}/lib/tools.jar:/opt/webserver7/lib
/ktsearch.jar:/opt/webserver7/lib/webserv-jstl.jar:/opt/webserver7/lib/jsf-impl.
jar:/opt/webserver7/lib/jsf-api.jar:/opt/webserver7/lib/webserv-jwsdp.jar:/opt/w
ebserver7/lib/container-auth.jar:/opt/webserver7/lib/mail.jar:/opt/webserver7/li
b/activation.jar</server-class-path>
    <debug>>false</debug>
    <debug-jvm-options>-Xdebug -Xrunjdpw:transport=dt_socket,server=y,suspend=n,
address=7896</debug-jvm-options>
    <jvm-options>-Djava.security.auth.login.config=login.conf</jvm-options>
    <jvm-options>-Djava.util.logging.manager=com.sun.webserver.logging.ServerLog
Manager</jvm-options>
    <jvm-options>-Xms128m -Xmx256m</jvm-options>
  </jvm>

  <thread-pool>
    <max-threads>128</max-threads>
    <stack-size>131072</stack-size>
  </thread-pool>

  <default-auth-db-name>keyfile</default-auth-db-name>

  <auth-db>
    <name>keyfile</name>
    <url>file</url>
    <property>
      <name>syntax</name>

```

EXAMPLE 2-1 server.xml File (Continued)

```
        <value>keyfile</value>
    </property>
</property>
    <name>keyfile</name>
    <value>keyfile</value>
</property>
</auth-db>

<acl-file>default.acl</acl-file>

<mime-file>mime.types</mime-file>

<access-log>
    <file>../logs/access</file>
</access-log>

<http-listener>
    <name>http-listener-1</name>
    <port>8082</port>
    <server-name>sun1</server-name>
    <default-virtual-server-name>sun1</default-virtual-server-name>
</http-listener>

<virtual-server>
    <name>sun1</name>
    <host>sun1</host>
    <http-listener-name>http-listener-1</http-listener-name>
</virtual-server>
</server>
```


Elements in server.xml

This chapter describes the elements in the `server.xml` file.

List of Elements

This section describes the elements in the `server.xml` file in alphabetical order.

- “access-log” on page 43
- “access-log-buffer” on page 43
- “acl-cache” on page 44
- “acl-db” on page 45
- “acl-file” on page 46
- “audit-accesses” on page 46
- “auth” on page 47
- “auth-db” on page 47
- “auth-realm” on page 48
- “cgi” on page 49
- “cluster” on page 50
- “connection-creation-property” on page 50
- “connection-lease-property” on page 51
- “convert” on page 52
- “custom-resource” on page 52
- “dav” on page 53
- “dav-collection” on page 54
- “default-auth-db-name” on page 55
- “default-auth-realm-name” on page 56
- “default-soap-auth-provider-name” on page 56
- “display-name” on page 56
- “dns” on page 56
- “dns-cache” on page 57
- “env-variable” on page 58

- “event” on page 58
- “external-jndi-resource” on page 59
- “file-cache” on page 60
- “http” on page 61
- “http-listener” on page 63
- “include” on page 64
- “index” on page 65
- “instance” on page 65
- “jdbc-resource” on page 66
- “jvm” on page 67
- “keep-alive” on page 69
- “lifecycle-module” on page 69
- “localization” on page 70
- “lock-db” on page 71
- “log” on page 71
- “mail-resource” on page 73
- “mime-file” on page 74
- “pkcs11” on page 74
- “profiler” on page 74
- “property” on page 75
- “property-db” on page 76
- “qos” on page 77
- “qos-limits” on page 77
- “request-policy” on page 78
- “response-policy” on page 78
- “search-app” on page 79
- “search-collection” on page 79
- “server” on page 80
- “servlet-container” on page 83
- “session-replication” on page 85
- “single-sign-on” on page 86
- “snmp” on page 87
- “soap-auth-provider” on page 87
- “ssl” on page 88
- “ssl-session-cache” on page 93
- “stats” on page 94
- “thread-pool” on page 95
- “time” on page 95
- “token” on page 96
- “variable” on page 96
- “virtual-server” on page 97
- “web-app” on page 99

access-log

The `access-log` element configures an HTTP access log. This element may appear zero or more times within the `server` element and zero or more times within the `virtual-server` element. For more information, see [“server” on page 80](#), and [“virtual-server” on page 97](#).

Subelements

The `access-log` element can contain the following subelements:

TABLE 3-1 List of `access-log` Subelements

Element	Occurrences	Description
<code>enabled</code>	0 or 1	Determines whether the server writes to this access log. The default value is <code>true</code> .
<code>name</code>	0 or 1	The name that uniquely identifies the access log. If you specify a name, the server will not automatically log to this access log. Instead, you should explicitly configure this access log in an <code>obj.conf AddLog</code> directive.
<code>file</code>	1	The file name of the access log. If a relative path is used, it is relative to the server's <code>config</code> directory. For example, <code>../logs/access</code> .
<code>format</code>	0 or 1	The format of the access log entries. The default format is the CLF (common log file) format. For more information on the access log format, see Appendix C, “Using the Custom Log File Format.”

See Also

- [“access-log-buffer” on page 43](#)
- [“audit-accesses” on page 46](#)
- [“event” on page 58](#)
- [“log” on page 71](#)

access-log-buffer

The `access-log-buffer` element configures the access log buffering subsystem. This element may appear zero or one time within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `access-log-buffer` element can contain the following subelements:

TABLE 3-2 List of access-log-buffer Subelements

Element	Occurrences	Description
<code>direct-io</code>	0 or 1	Indicates whether the file system should cache access log writes. The default value, <code>false</code> , indicates that the file system should cache access log writes. Setting the value to <code>true</code> indicates that the file system should not cache access log writes. The setting is purely advisory; either the server or the operating system may choose to ignore it.
<code>enabled</code>	0 or 1	Determines whether the server buffers the access log entries. The default value is <code>true</code> .
<code>buffer-size</code>	0 or 1	The size (in bytes) of individual access log buffers. The value can be from 4096 to 1048576.
<code>max-buffers</code>	1	The maximum number of access log buffers per server. The value can be from 1 to 65536.
<code>max-buffers-per-file</code>	0 or 1	The maximum number of access log buffers per access log file. The value can be from 1 to 128.
<code>max-age</code>	0 or 1	The maximum time (in seconds) to buffer a given access log entry. The value can be from 0.001 to 3600.

See Also

- [“access-log” on page 43](#)
- [“event” on page 58](#)
- [“log” on page 71](#)

acl-cache

The `acl-cache` element configures the Access Control List (ACL) cache. This element may appear zero or one time within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `acl-cache` element can contain the following subelements:

TABLE 3-3 List of acl-cache Subelements

Element	Occurrences	Description
<code>enabled</code>	0 or 1	Determines whether the server caches ACLs and information about authenticated users. The default value is <code>true</code> .

TABLE 3-3 List of `acL-cache` Subelements *(Continued)*

Element	Occurrences	Description
<code>max-users</code>	0 or 1	The maximum number of users for which the server will cache the authentication information. The value can be from 1 to 1048576.
<code>max-groups-per-user</code>	0 or 1	The maximum number of groups per user for which the server will cache the membership information. The value can be from 1 to 1024.
<code>max-age</code>	0 or 1	The maximum time (in seconds) required to cache the authentication information. The value can be from 0.001 to 3600.

See Also

- [“acL-file” on page 46](#)
- [“auth-db” on page 47](#)
- [“dns-cache” on page 57](#)
- [“file-cache” on page 60](#)

acL-db

The `acL-db` element configures the WebDAV Access Control Protocol ACL database. This element may appear zero or one time within the `dav` element and zero or one time within the `dav-collection` element. For more information, see [“dav” on page 53](#) and [“dav-collection” on page 54](#).

Subelements

The `acL-db` element can contain the following subelements:

TABLE 3-4 List of `acL-db` Subelements

Element	Occurrences	Description
<code>max-entries</code>	0 or 1	The maximum number of Access Control Entries (ACE) that can be allowed on a single resource. The value can be from 0 to 2147483647, or -1 for no limit.
<code>max-size</code>	0 or 1	The maximum size (in bytes) of memory representation of the WebDAV ACL database for a collection. If the memory limit specified using this subelement is exceeded, the server will not cache the WebDAV ACLs. The value can be from 0 to 2147483647, or -1 for no limit.

TABLE 3-4 List of `ac1-db` Subelements *(Continued)*

Element	Occurrences	Description
<code>update-interval</code>	0 or 1	The interval (in seconds) at which the WebDAV ACL databases are synchronized to the disk. The value can be from 0.001 to 3600, or 0 to disable caching of WebDAV ACLs.

See Also

- [“ac1-file” on page 46](#)
- [“auth” on page 47](#)
- [“auth-db” on page 47](#)
- [Chapter 9, “ACL Files”](#)

ac1-file

The `ac1-file` element defines a file that controls access to the server. This element may appear zero or more times within the `server` element and zero or more times within the `virtual-server` element. For more information, see [“server” on page 80](#), and [“virtual-server” on page 97](#).

The value of this element is the file name of the ACL file. If a relative path is used, it is relative to the server's config directory. This element does not contain any subelements.

See Also

- [“ac1-db” on page 45](#)
- [“auth” on page 47](#)
- [“auth-db” on page 47](#)
- [Chapter 9, “ACL Files”](#)

audit-accesses

The `audit-accesses` element determines whether authentication and authorization events are logged. This element may appear zero or one time within the `server` element. For more information, see [“server” on page 80](#). The default value is `false`. This element does not contain any subelements.

See Also

- [“access-log” on page 43](#)
- [“event” on page 58](#)
- [“log” on page 71](#)

auth

The `auth` element configures WebDAV Access Control Protocol authentication. This element may appear zero or one time within the `dav` element and zero or one time within the `dav-collection` element. For more information, see [“dav” on page 53](#) and [“dav-collection” on page 54](#).

Subelements

The `auth` element can contain the following subelements:

TABLE 3-5 List of auth Subelements

Element	Occurrences	Description
<code>auth-db-name</code>	0 or 1	The ACL authentication database to use. The value is the name from an <code>auth-db</code> element. The default value is the value of the <code>default-auth-db-name</code> element. For more information, see “auth-db” on page 47 .
<code>method</code>	0 or 1	The authentication method to use. The value can be <code>basic</code> , <code>digest</code> , or <code>ssl</code> . The default value is <code>basic</code> .
<code>prompt</code>	0 or 1	The prompt that is displayed to clients when they request authentication. The default prompt is Sun Java System Web Server WebDAV.

See Also

- [“acl-file” on page 46](#)
- [“acl-db” on page 45](#)
- [“auth-db” on page 47](#)
- [“default-auth-db-name” on page 55](#)
- Chapter 9, “ACL Files”

auth-db

The `auth-db` element configures an ACL authentication database. This element may appear zero or more times within the `server` element and zero or more times within the `virtual-server` element. For more information, see [“server” on page 80](#), and [“virtual-server” on page 97](#).

Subelements

The `auth-db` element can contain the following subelements:

TABLE 3-6 List of auth-db Subelements

Element	Occurrences	Description
enabled	0 or 1	Determines whether the ACL authentication database is enabled at runtime. The default value is <code>true</code> .
name	1	The name that uniquely identifies the ACL authentication database for use in ACL files.
url	1	The URL of the ACL authentication database. The type of ACL authentication database is specified in the URL scheme. For example, <code>ldap://ds.example.com/dc=example,dc=com</code> configures a LDAP directory server as an ACL authentication database.
auth-expiring-url	0 or 1	The URL to which the server redirects the client if the supplied password is about to expire.
property	0 or more	Configures the ACL authentication database properties. For more details, see “property” on page 75 .
description	0 or 1	The description of the ACL authentication database. The value is in text format.

See Also

- [“acl-file” on page 46](#)
- [“acl-db” on page 45](#)
- [“auth” on page 47](#)
- [“default-auth-db-name” on page 55](#)
- [Chapter 9, “ACL Files”](#)

auth-realm

The `auth-realm` element configures a Servlet container authentication realm, which is used to authenticate access to web applications. This element may appear zero or more times within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `auth-realm` element can contain the following subelements:

TABLE 3-7 List of auth-realm Subelements

Element	Occurrences	Description
name	1	The name that uniquely identifies the Servlet container authentication realm.
type	0 or 1	The type of a built-in authentication realm. Only applicable when <code>class</code> is omitted. The value can be <code>file</code> , <code>ldap</code> , <code>pam</code> , <code>certificate</code> , or <code>native</code> .
class	0 or 1	The class that implements a Servlet container authentication realm. Only applicable when <code>type</code> is omitted. The value is a class name.
property	0 or more	The Servlet container authentication realm properties. For more details, see “property” on page 75 .

See Also

- [“default-auth-realm-name” on page 56](#)
- [“servlet-container” on page 83](#)

cgi

The `cgi` element configures the CGI execution subsystem. This element may appear zero or one time within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `cgi` element can contain the following subelements:

TABLE 3-8 List of cgi Subelements

Element	Occurrences	Description
timeout	0 or 1	The timeout (in seconds) after which the server will terminate a CGI program. The value can be from 0.001 to 3600, or -1 for no timeout.
idle-timeout	0 or 1	The timeout (in seconds) after which the server will terminate a nonresponsive CGI program. The value can be from 0.001 to 3600.
cgistub-path	0 or 1	The path to the <code>Cgistub</code> binary. If a relative path is used, it is relative to the server's config directory.

TABLE 3-8 List of cgi Subelements (Continued)

Element	Occurrences	Description
<code>cgistub-idle-timeout</code>	0 or 1	The timeout (in seconds) after which an unused <code>Cgistub</code> process will be terminated. The value can be from 0.001 to 3600.
<code>min-cgistubs</code>	0 or 1	The minimum number of <code>Cgistub</code> processes the server keeps on hand, waiting to run the CGI programs. The value can be from 0 to 4096.
<code>max-cgistubs</code>	0 or 1	The maximum number of <code>Cgistub</code> processes the server keeps on hand, waiting to run the CGI programs. The value can be from 1 to 4096.
<code>env-variable</code>	0 or more	Configures the CGI program environment variables. For more details, see “env-variable” on page 58 .

cluster

The `cluster` element defines the cluster to which the server belongs. This element may appear zero or one time within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `cluster` element can contain the following subelements:

TABLE 3-9 List of cluster Subelements

Element	Occurrences	Description
<code>local-host</code>	1	Defines the network address of an instance. The value is the host value from an <code>instance</code> element. For more details, see “instance” on page 65 .
<code>instance</code>	1 or more	Defines a member of the server cluster. For more details, see “instance” on page 65 .
<code>session-replication</code>	0 or 1	Configures the Servlet session replication for the server cluster. For more details, see “session-replication” on page 85 .

connection-creation-property

The `connection-creation-property` element configures the properties that are set when a JDBC connection (`java.sql.Connection`) is created. This element may appear zero or more times within the `jdbc-resource` element. For more information, see [“jdbc-resource” on page 66](#).

Subelements

The `connection-creation-property` element can contain the following subelements:

TABLE 3-10 List of `connection-creation-property` Subelements

Element	Occurrences	Description
<code>name</code>	1	The name of the property
<code>value</code>	1	The value of the property
<code>description</code>	0 or 1	The description of the property

See Also

- [“connection-lease-property” on page 51](#)
- [“jdbc-resource” on page 66](#)
- [“property” on page 75](#)

connection-lease-property

The `connection-lease-property` element configures the properties that are set each time a JDBC connection (`java.sql.Connection`) is leased to an application. This element may appear zero or more times within the `jdbc-resource` element. For more information, see [“jdbc-resource” on page 66](#).

Subelements

The `connection-lease-property` element can contain the following subelements:

TABLE 3-11 List of `connection-lease-property` Subelements

Element	Occurrences	Description
<code>name</code>	1	The name of the property
<code>value</code>	1	The value of the property
<code>description</code>	0 or 1	The description of the property

See Also

- [“connection-creation-property” on page 50](#)
- [“jdbc-resource” on page 66](#)
- [“property” on page 75](#)

convert

The `convert` element determines the type of documents that are converted prior to indexing. This element may appear zero or one time within the `search-collection` element. Documents with the `pdf` file extension are always converted to HTML prior to indexing. For more information, see [“search-collection” on page 79](#).

Subelements

The `convert` element can contain the following subelements:

TABLE 3-12 List of `convert` Subelements

Element	Occurrences	Description
<code>extension</code>	0 or more	The file extension of a document type that should be converted to HTML.

See Also

- [“include” on page 64](#)
- [“index” on page 65](#)
- [“search-app” on page 79](#)
- [“search-collection” on page 79](#)

custom-resource

The `custom-resource` element configures a resource implemented by a custom Java class. This element may appear zero or more times within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `custom-resource` element can contain the following subelements:

TABLE 3-13 List of `custom-resource` Subelements

Element	Occurrences	Description
<code>enabled</code>	0 or 1	Determines whether the custom resource is enabled at runtime. The default value is <code>true</code> .
<code>jndi-name</code>	1	The JNDI name of the custom resource.
<code>res-type</code>	1	The type of custom resource. The value is a class name.

TABLE 3-13 List of custom-resource Subelements (Continued)

Element	Occurrences	Description
factory-class	1	The class that instantiates a naming context which is used to look up the external resource. The value is a name of a class that implements <code>javax.naming.spi.ObjectFactory</code> .
property	0 or more	Configures the optional resource-specific properties. For more details, see “ property ” on page 75.
description	0 or 1	The description of the custom resource. The value of this element is in text format.

dav

The `dav` element configures WebDAV. This element may appear zero or one time within the `server` element. For more information, see “[server](#)” on page 80.

Subelements

The `dav` element can contain the following subelements:

TABLE 3-14 List of dav Subelements

Element	Occurrences	Description
enabled	0 or 1	Determines whether WebDAV is enabled at runtime. The default value is <code>true</code> .
default-owner	0 or 1	Defines the name of the default owner of the resource.
min-lock-timeout	0 or 1	The minimum expiration time (in seconds) for WebDAV locks. The value can be from 0.001 to 3600, or 0 for no limit.
max-xml-request-body-size	0 or 1	The maximum size (in bytes) allowed for WebDAV XML request bodies. The value can be from 0 to 2147483647, or -1 for no limit.
max-propfind-depth	0 or 1	The maximum depth of <code>PROPFIND</code> requests sent to collections. The value can be 0, 1, or infinity.
max-expand-property-depth	0 or 1	The maximum depth allowed for WebDAV <code>expand-property REPORT</code> responses. The value can be from 0 to 100.
max-report-response-elements	0 or 1	The maximum number of response elements a <code>REPORT</code> response body can have. The value can be from 0 to 2147483647. The default value is 1000.

TABLE 3-14 List of dav Subelements (Continued)

Element	Occurrences	Description
auth	0 or 1	Configures the WebDAV Access Control Protocol authentication. For more details, see “auth” on page 47.
acl-db	0 or 1	Configures the WebDAV Access Control Protocol ACL database. For more details, see “acl-db” on page 45.
lock-db	0 or 1	Configures the WebDAV lock database. For more details, see “lock-db” on page 71.
property-db	0 or 1	Configures the WebDAV property database. For more details, see “property-db” on page 76.

See Also

- “dav-collection” on page 54
- “lock-db” on page 71
- “property-db” on page 76

dav-collection

The `dav-collection` element configures a WebDAV collection. This element may appear zero or more times within the `virtual-server` element. For more information, see “virtual-server” on page 97.

Subelements

The `dav-collection` element can contain the following subelements:

TABLE 3-15 List of dav-collection Subelements

Element	Occurrences	Description
enabled	0 or 1	Determines whether WebDAV is enabled at runtime. The default value is <code>true</code> .
default-owner	0 or 1	The name of the default owner of the resource.
uri	1	The existing root URI on which the WebDAV should be enabled.
source-uri	0 or 1	The URI which the WebDAV clients can use to access the source code of content.
min-lock-timeout	0 or 1	The minimum expiration time (in seconds) for WebDAV locks. The value can be from 0.001 to 3600, or <code>-1</code> for no limit.

TABLE 3-15 List of dav-collection Subelements (Continued)

Element	Occurrences	Description
max-xml-request-body-size	0 or 1	The maximum size (in bytes) allowed for WebDAV XML request bodies. The value can be from 0 to 2147483647, or -1 for no limit.
max-propfind-depth	0 or 1	The maximum depth of PROPFIND requests sent to collections. The value can be 0, 1, or infinity.
max-expand-property-depth	0 or 1	The maximum depth allowed for WebDAV expand-property REPORT responses. The value can be from 0 to 100.
max-report-response-elements	0 or 1	The maximum number of response elements a REPORT response body can have. The value can be from 0 to 2147483647. The default value is 1000.
auth	0 or 1	Configures the WebDAV Access Control Protocol authentication. For more details, see “auth” on page 47 .
acl-db	0 or 1	Configures the WebDAV Access Control Protocol ACL database. For more details, see “acl-db” on page 45 .
lock-db	0 or 1	Configures the WebDAV lock database. For more details, see “lock-db” on page 71 .
property-db	0 or 1	Configures the WebDAV property database. For more details, see “property-db” on page 76 .
description	0 or 1	The description of the WebDAV collection.

See Also

- [“dav” on page 53](#)
- [“lock-db” on page 71](#)
- [“property-db” on page 76](#)

default-auth-db-name

The `default-auth-db-name` element specifies the name of the default ACL authentication database. This element may appear zero or one time within the `server` element. For more information, see [“server” on page 80](#). This element does not contain any subelements.

See Also

- [“auth-db” on page 47](#)

default-auth-realm-name

The `default-auth-realm-name` element specifies the name of the default Servlet container authentication realm. This element may appear zero or one time within the `server` element. For more information, see [“server” on page 80](#). This element does not contain any subelements.

See Also

[“auth-realm” on page 48](#)

default-soap-auth-provider-name

The `default-soap-auth-provider-name` element specifies the name of the default Simple Object Access Protocol (SOAP) message-level authentication provider. This element may appear zero or one time within the `server` element. For more information, see [“server” on page 80](#). This element does not contain any subelements.

See Also

[“soap-auth-provider” on page 87](#)

display-name

The `display-name` element specifies a human-readable name for the collection to be used while displaying the collection to the end user. This element does not contain any subelements.

See Also

[“search-collection” on page 79](#)

dns

The `dns` element configures how the server uses the domain name system (DNS). This element may appear zero or one time within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `dns` element can contain the following subelements:

TABLE 3-16 List of dns Subelements

Element	Occurrences	Description
enabled	0 or 1	Determines whether the server does DNS lookups. The default value is <code>false</code> .
async	0 or 1	Determines whether the server uses its own asynchronous DNS resolver instead of the operating system's synchronous resolver. The default value is <code>true</code> .
timeout	0 or 1	The timeout (in seconds) for asynchronous DNS lookups. The value can be from 0.001 to 3600.

See Also

[“dns-cache” on page 57](#)

dns-cache

The `dns-cache` element configures the DNS cache. This element may appear zero or one time within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `dns-cache` element can contain the following subelements:

TABLE 3-17 List of dns-cache Subelements

Element	Occurrences	Description
enabled	0 or 1	Determines whether the server caches DNS lookup results. The default value is <code>true</code> .
max-age	0 or 1	The maximum amount of time (in seconds) to cache a DNS lookup result. The value can be from 1 to 31536000.
max-entries	0 or 1	The maximum number of DNS lookup results to cache. The value can be from 32 to 32768.

See Also

- [“acl-cache” on page 44](#)
- [“dns” on page 56](#)
- [“file-cache” on page 60](#)

env-variable

The `env-variable` element defines an environment variable. This element may appear zero or one time within the `cgi` element. For more information, see [“cgi” on page 49](#).

Subelements

The `env-variable` element can contain the following subelements:

TABLE 3-18 List of `env-variable` Subelements

Element	Occurrences	Description
<code>name</code>	1	The name of the environment variable
<code>value</code>	1	The value of the environment variable
<code>description</code>	0 or 1	The description of the environment variable

See Also

[“variable” on page 96](#)

event

The `event` element configures a recurring event. This element may appear zero or more times within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `event` element can contain the following subelements:

TABLE 3-19 List of `event` Subelements

Element	Occurrences	Description
<code>enabled</code>	0 or 1	Determines whether the event is enabled at runtime. The default value is <code>true</code> .
<code>time</code>	0 or more	Configures a specific time when the event occurs. For more details, see “time” on page 95 .
<code>interval</code>	0 or 1	The interval (in seconds) at which the event occurs. The value can be from 60 to 86400.
<code>rotate-log</code>	0 or 1	Rotates the log files. The default value is <code>false</code> .

TABLE 3-19 List of event Subelements *(Continued)*

Element	Occurrences	Description
rotate-access-log	0 or 1	Rotates the access log files. The default value is <code>false</code> .
command	0 or more	The command to execute when the event runs.
reconfig	0 or 1	Dynamically reconfigures the server. The default value is <code>false</code> .
restart	0 or 1	Restarts the server. The default value is <code>false</code> .
description	0 or 1	The description of the event. The value of this element is in text format.

See Also

- [“access-log” on page 43](#)
- [“log” on page 71](#)

external-jndi-resource

The `external-jndi-resource` element configures a resource provided by an external JNDI repository. This element may appear zero or more times within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `external-jndi-resource` element can contain the following subelements:

TABLE 3-20 List of external-jndi-resource Subelements

Element	Occurrences	Description
enabled	0 or 1	Determines whether the event is enabled at runtime. The default value is <code>true</code> .
jndi-name	1	The JNDI name of the resource.
jndi-lookup-name	1	The JNDI lookup name of the resource.
res-type	1	The type of the external JNDI resource. The default value is a class name.
factory-class	1	The class that instantiates resources of this type. The default value is a class name that implements <code>javax.naming.spi.InitialContextFactory</code> .
property	0 or more	Configures the optional resource-specific properties. For more details, see “property” on page 75 .

TABLE 3-20 List of external-jndi-resource Subelements (Continued)

Element	Occurrences	Description
description	0 or 1	The description of the resource. The value of this element should be in text format.

file-cache

The `file-cache` element configures the file cache. This element may appear zero or one time within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `file-cache` element can contain the following subelements:

TABLE 3-21 List of file-cache Subelements

Element	Occurrences	Description
enabled	0 or 1	Determines whether the server cache is enabled. The default value is <code>true</code> . Whether file content is cached in addition to meta information is controlled by the <code>cache-content</code> subelement.
max-age	0 or 1	The maximum amount of time (in seconds) to cache file content and meta information. The value can be from <code>-1</code> to <code>3600</code> . The value <code>-1</code> indicates that the file cache entries do not expire.
max-entries	0 or 1	The maximum number of paths to cache content and/or meta information. The value can be from <code>1</code> to <code>1048576</code> .
max-open-files	0 or 1	The maximum number of file descriptors the file cache will keep open. The value can be from <code>1</code> to <code>1048576</code> .
sendfile	0 or 1	Determines whether the server will attempt to use the operating system's <code>sendfile</code> , <code>sendfilev</code> , <code>send_file</code> , or <code>TransmitFile</code> system call. The default value is <code>true</code> on Windows and <code>false</code> on other platforms.
copy-files	0 or 1	Determines whether the server copies cached files to a temporary directory. The default value is <code>true</code> on Windows and <code>false</code> on other platforms.
copy-path	0 or 1	The temporary directory that is used when <code>copy-files</code> is <code>true</code> . If a relative path is used, it is relative to the server's <code>config</code> directory.
replacement	0 or 1	The cache entry replacement algorithm. The value can be <code>false</code> , <code>lru</code> , or <code>lfu</code> .

TABLE 3-21 List of file-cache Subelements (Continued)

Element	Occurrences	Description
cache-content	0 or 1	Determines whether the server caches file content in addition to the meta information. The default value is <code>true</code> .
max-heap-file-size	0 or 1	The maximum size (in bytes) of files to cache on the heap. The value can be from 0 to 2147483647.
max-heap-space	0 or 1	The maximum amount (in bytes) of heap to use for caching files. The value can be from 0 to 1099511627776.
max-mmap-file-size	0 or 1	The maximum size (in bytes) of files to mmap. The value can be from 0 to 2147483647.
max-mmap-space	0 or 1	The maximum amount (in bytes) of mmap address space to use for caching files. The value can be from 0 to 1099511627776.
sendfile-size	0	Web Server uses <code>sendfile-size</code> only when the file size is greater than <code>max-heap-file-size</code> . When <code>sendfile-size</code> is set to non-zero value and <code>file-cache</code> and <code>sendfile</code> are enabled, it hints the file-cache subsystem to send the file in chunks. The chunk size can be at the most the size of <code>sendfile-size</code> . When <code>sendfile-size</code> is set to default value, that is, zero and <code>sendfile</code> is enabled, the entire file is attempted to be sent out. In either case, the left over file is sent in subsequent transfers.

See Also

- “[acl-cache](#)” on page 44
- “[dns-cache](#)” on page 57

http

The `http` element configures miscellaneous HTTP protocol options. This element may appear zero or one time within the `server` element. For more information, see “[server](#)” on page 80.

Subelements

The `http` element can contain the following subelements:

TABLE 3-22 List of http Subelements

Element	Occurrences	Description
version	0 or 1	The highest HTTP protocol version the server supports. The default HTTP version string is <code>HTTP/1.1</code> .

TABLE 3-22 List of http Subelements (Continued)

Element	Occurrences	Description
server-header	0 or 1	The server header information, such as the name of the server software and version. The default server header is <code>Sun-Java-System-Web-Server/7.0</code> .
request-header-buffer-size	0 or 1	The size (in bytes) of the buffer used to read HTTP request headers. The value can be from 0 to 2147483647.
strict-request-headers	0 or 1	Determines whether the server rejects certain malformed HTTP request headers. The default value is <code>false</code> .
max-request-headers	0 or 1	The maximum number of header fields to allow in an HTTP request header. The value can be from 1 to 512.
output-buffer-size	0 or 1	The size (in bytes) of buffer used to buffer HTTP responses. The value can be from 0 to 2147483647.
max-unchunk-size	0 or 1	The maximum size (in bytes) of a chunked HTTP request body that the server will unchunk. The value can be from 0 to 2147483647.
unchunk-timeout	0 or 1	The maximum time (in seconds) that the server waits for a chunked HTTP request body to arrive. The value can be from 0 to 3600, or -1 for no timeout.
io-timeout	0 or 1	The maximum time (in seconds) that the server waits for an individual packet. The value can be from 0 to 3600, or -1 for no timeout.
request-header-timeout	0 or 1	The maximum time (in seconds) that the server waits for a complete HTTP request header. The value can be from 0 to 604800, or -1 for no timeout.
request-body-timeout	0 or 1	The maximum time (in seconds) that the server waits for a complete HTTP request body. The value can be from 0 to 604800, or -1 for no timeout.
favicon	0 or 1	Determines whether the server replies to requests for <code>favicon.ico</code> with its own built-in icon file. The default value is <code>true</code> .
etag	0 or 1	Controls whether the server includes an Etag header field in its responses. The default value is <code>true</code> .

See Also

- “[http-listener](#)” on page 63
- “[keep-alive](#)” on page 69
- “[thread-pool](#)” on page 95
- “[virtual-server](#)” on page 97

http-listener

The `http-listener` element configures an HTTP listener. This element may appear zero or more times within the `server` element. For more information, see “[server](#)” on page 80.

Subelements

The `http-listener` element can contain the following subelements:

TABLE 3–23 List of `http-listener` Subelements

Element	Occurrences	Description
<code>enabled</code>	0 or 1	Determines whether the HTTP listener is enabled at runtime. The default value is <code>true</code> .
<code>name</code>	1	The name that uniquely identifies the HTTP listener.
<code>ip</code>	0 or 1	The IP address on which to listen. The value of this element is a specific IP address, or <code>*</code> to listen on all IP addresses.
<code>port</code>	1	The port on which to listen. The value of this element is the port number.
<code>acceptor-threads</code>	0 or 1	The number of threads dedicated to accept connections received by this listener. The value can be from 1 to 128.
<code>server-name</code>	1	The default server name. The value can include a scheme prefix (for example, <code>http://</code>) and port suffix (for example, <code>:80</code>).
<code>blocking-io</code>	0 or 1	Determines whether the server uses blocking IO. The default value is <code>false</code> .
<code>family</code>	0 or 1	The name of the protocol family.
<code>handle-protocol-mismatch</code>	0 or 1	Controls the server's response to SSL or non-SSL protocol mismatches in client requests. A mismatch occurs when a client uses SSL to send a request to a non-SSL listener, or when a client sends a request to an SSL listener without using SSL. The default is <code>true</code> , which means that the server attempts to detect SSL or non-SSL protocol mismatches and sends an HTTP redirect or SSL alert when a mismatch is detected.
<code>listen-queue-size</code>	0 or 1	The maximum size (in bytes) of the operating system listen queue backlog. The value of this element can be from 1 to 1048576.
<code>receive-buffer-size</code>	0 or 1	The size (in bytes) of the operating system socket receive buffer. The value of this element can be from 1 to 1048576.
<code>send-buffer-size</code>	0 or 1	The size (in bytes) of the operating system socket send buffer. The value of this element can be from 1 to 1048576.

TABLE 3-23 List of http-listener Subelements (Continued)

Element	Occurrences	Description
default-virtual-server-name 1		The name of the virtual server that processes requests that do not match a host. The value of this element is the name value from a virtual-server element. For more details, see “virtual-server” on page 97 .
ssl	0 or 1	Configures SSL/TLS. For more details, see “ssl” on page 88 .
description	0 or 1	The description of the HTTP listener. The value of this element should be in a text format.

See Also

- [“http” on page 61](#)
- [“keep-alive” on page 69](#)
- [“virtual-server” on page 97](#)

include

The `include` element configures the document types that should be indexed. This element may appear zero or one time within the `search-collection` element. For more information, see [“search-collection” on page 79](#).

If the `include` element is not present, only documents matching the `*.html`, `*.htm`, `*.txt`, `*.pdf`, patterns are indexed. Documents with the `jar`, `sxc`, `sxg`, `sxi`, `sxm`, `sxw`, `war`, and `zip` file extensions are never indexed.

Subelements

The `include` element can contain the following subelement:

TABLE 3-24 List of include Subelements

Element	Occurrences	Description
pattern	0 or more	Specifies the wildcard pattern of files to be indexed

See Also

- [“convert” on page 52](#)
- [“index” on page 65](#)
- [“search-app” on page 79](#)
- [“search-collection” on page 79](#)

index

The `index` element configures the document fields that are indexed for searching. This element may appear zero or one time within the `search-collection` element. For more information, see [“search-collection” on page 79](#).

Subelements

The `index` element can contain the following subelement:

TABLE 3–25 List of index subelement

Element	Occurrences	Description
<code>meta-tag</code>	0 or more	The name of the HTML meta tag that should be indexed

See Also

- [“convert” on page 52](#)
- [“include” on page 64](#)
- [“search-app” on page 79](#)
- [“search-collection” on page 79](#)

instance

The `instance` element defines a member of a server cluster. This element may appear one or more times within the `cluster` element. For more information, see [“cluster” on page 50](#).

Subelements

The `instance` element can contain the following subelements:

TABLE 3–26 List of instance Subelements

Element	Occurrences	Description
<code>enabled</code>	0 or 1	Determines whether the instance is enabled at runtime. The default value is <code>true</code> .
<code>host</code>	1	The network address of the instance. The value is the host name or the IP address.
<code>session-replication</code>	0 or 1	Configures the Servlet session replication for the instance. For more details, see “session-replication” on page 85 .

See Also

- [“cluster” on page 50](#)
- [“session-replication” on page 85](#)

jdbc-resource

The `jdbc-resource` element configures a Java Database Connectivity (JDBC) resource. This element may appear zero or more times within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `jdbc-resource` element can contain the following subelements:

TABLE 3-27 List of `jdbc-resource` Subelements

Element	Occurrences	Description
<code>enabled</code>	0 or 1	Determines whether the resource is enabled at runtime. The default value is <code>true</code> .
<code>jndi-name</code>	1	The JNDI name of the resource.
<code>datasource-class</code>	1	The JDBC driver class. The value is a name of the class that implements <code>java.sql.DataSource</code> and <code>java.sql.XADataSource</code> .
<code>min-connections</code>	1	The minimum number of concurrent connections to maintain to the database server. The value can be from 1 to 4096.
<code>max-connections</code>	1	The maximum number of concurrent connections to maintain to the database server. The value can be from 1 to 4096.
<code>idle-timeout</code>	1	The timeout (in seconds) after which an idle connection to the database server will be closed. The value can be from 0 to 3600, or -1 for no timeout.
<code>wait-timeout</code>	1	The timeout (in seconds) after which a caller waiting for an available connection to the database server will receive an error. The value can be from 0.001 to 3600, or -1 for no timeout.
<code>isolation-level</code>	0 or 1	Specifies the transaction isolation level. The value can be <code>read-uncommitted</code> , <code>read-committed</code> , <code>repeatable-read</code> , or <code>serializable</code> .

TABLE 3-27 List of jdbc-resource Subelements *(Continued)*

Element	Occurrences	Description
isolation-level-guaranteed	0 or 1	Determines whether the server sets the isolation level each time a connection is leased to an application. The default value is <code>false</code> .
connection-validation	0 or 1	Specifies how the server validates a connection before leasing it to an application. The value can be <code>false</code> , <code>auto-commit</code> , <code>meta-data</code> , or <code>table</code> .
connection-validation-table-name	0 or 1	The name of the table used when <code>connection-validation</code> is <code>table</code> . The value is the database table name.
fail-all-connections	0 or 1	Determines whether all connections are immediately closed and reestablished when there is an error validating an individual connection. The default value is <code>false</code> .
property	0 or more	Configures the JDBC driver (<code>java.sql.DataSource</code> and <code>java.sql.XADataSource</code>) properties. For more details, see “property” on page 75 .
connection-creation-property	0 or more	Configures the JDBC connection (<code>java.sql.Connection</code>) properties, when a new connection is created. For more details, see “connection-creation-property” on page 50 .
connection-lease-property	0 or more	Configures the JDBC connection (<code>java.sql.Connection</code>) properties each time a connection is leased to an application. For more details, see “connection-lease-property” on page 51 .
description	0 or 1	The description of the resource.

See Also

- [“connection-creation-property” on page 50](#)
- [“connection-lease-property” on page 51](#)
- [“property” on page 75](#)

jvm

The `jvm` element configures the Java Virtual Machine (JVM). This element may appear zero or one time within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `jvm` element can contain the following subelements:

TABLE 3-28 List of JVM Subelements

Element	Occurrences	Description
<code>enabled</code>	0 or 1	Determines whether the server creates a JVM. The default value is <code>true</code> .
<code>java-home</code>	1	The location of the JDK. If relative path is used, it is relative to the server's <code>config</code> directory.
<code>class-path-prefix</code>	0 or 1	The prefix for the system classpath. Because this classpath takes precedence over the server classpath, changing it can prevent the server from working properly. To add classes to the system classpath, use the <code>class-path-suffix</code> element instead.
<code>server-class-path</code>	0 or 1	The classpath containing server classes. Changing this classpath can prevent the server from working properly. To add classes to the system class path, use the <code>class-path-suffix</code> element instead.
<code>class-path-suffix</code>	0 or 1	The suffix for the system classpath.
<code>env-class-path-ignored</code>	0 or 1	Determines whether the server ignores the <code>CLASSPATH</code> environment variable. The default value is <code>true</code> .
<code>native-library-path-prefix</code>	0 or 1	The prefix for the operating system native library path.
<code>sticky-attach</code>	0 or 1	Determines whether the server attaches each HTTP request processing thread to the JVM only once or attaches and detaches on each request. The default value is <code>true</code> .
<code>debug</code>	0 or 1	Determines whether JVM is started in debug mode, ready for attachment with a Java Platform Debugger Architecture (JPDA) debugger. The default value is <code>false</code> .
<code>debug-jvm-options</code>	0 or more	Defines the JPDA options. For more details, see http://java.sun.com/products/jpda/doc/conninv.html#Invocation
<code>jvm-options</code>	0 or more	Defines the server-wide JVM options. For more details, see http://java.sun.com/docs/hotspot/VMOptions.html
<code>bytecode-preprocessor-class</code>	0 or more	The name of the bytecode preprocessor class. The value is a name of a class that implements <code>com.sun.appserv.BytecodePreprocessor</code> .
<code>profiler</code>	0 or 1	Configures a Java profiler. For more details, see “ profiler ” on page 74 .

See Also

“[servlet-container](#)” on [page 83](#)

keep-alive

The `keep-alive` element configures the HTTP keep-alive subsystem. This element may appear zero or one time within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `keep-alive` element can contain the following subelements:

TABLE 3-29 List of keep-alive Subelements

Element	Occurrences	Description
<code>enabled</code>	0 or 1	Determines whether the keep-alive subsystem is enabled at runtime. The default value is <code>true</code> .
<code>threads</code>	0 or 1	The number of keep-alive subsystem threads. The value can be from 1 to 128. The default value is 1.
<code>max-connections</code>	0 or 1	The maximum number of concurrent keep-alive connections that the server supports. The value can be from 1 to 1048576. The default value is 200.
<code>timeout</code>	0 or 1	The timeout (in seconds) after which an inactive keep-alive connection can be closed. The value can be from 0.001 to 3600. The default value is 30 seconds.
<code>poll-interval</code>	0 or 1	The interval (in seconds) between polls. The value can be from 0.001 to 1. The default value is .001.

See Also

- [“http” on page 61](#)
- [“http-listener” on page 63](#)
- [“virtual-server” on page 97](#)
- [“thread-pool” on page 95](#)

lifecycle-module

The `lifecycle-module` element configures a Java server lifecycle module, a user-defined class that implements `com.sun.appserv.server.LifecycleListener`. This element may appear zero or more times within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `lifecycle-module` element can contain the following subelements:

TABLE 3-30 List of lifecycle-modules Subelements

Element	Occurrences	Description
enabled	0 or 1	Determines whether the lifecycle module is enabled at runtime. The default value is <code>true</code> .
name	1	The name that uniquely identifies the lifecycle module.
class	1	The class that implements the lifecycle module. The value is the name of a class that implements <code>com.sun.appserv.server.LifecycleListener</code> .
load-order	0 or 1	The order in which the lifecycle module is loaded. It is recommended that you choose a load-order value that is greater than or equal to 100 to avoid conflicts with internal lifecycle modules. The value can be from 0 to 2147483647. Values from 0 to 99 are reserved for internal use.
is-failure-fatal	0 or 1	Determines whether the server should treat exceptions thrown during lifecycle module initialization as fatal. The default value is <code>true</code> .
class-path	0 or 1	The classpath for the lifecycle module.
property	0 or more	Configures optional lifecycle-module-specific properties. For more details, see “property” on page 75 .
description	0 or 1	The description of the resource.

Localization

The `localization` element configures how the server chooses the language in which it presents information to the client. This element may appear zero or one time within the `server` element, and zero or one time within the `virtual-server` element. For more information, see [“server” on page 80](#), and [“virtual-server” on page 97](#).

Subelements

The `localization` element can contain the following subelements:

TABLE 3-31 List of localization Subelements

Element	Occurrences	Description
default-language	0 or 1	The default language in which the messages and content are displayed. The value is a language tag.

TABLE 3-31 List of localization Subelements (Continued)

Element	Occurrences	Description
<code>negotiate-client-language</code>	0 or 1	Determines whether the server attempts to use the <code>Accept-language</code> HTTP header to negotiate the content language with clients. The default value is <code>false</code> .

lock-db

The `lock-db` element configures the WebDAV lock database. This element may appear zero or one time within the `dav` element, and zero or one time within the `dav-collection` element. For more information, see [“dav” on page 53](#) and [“dav-collection” on page 54](#).

Subelements

The `lock-db` element can contain the following subelements:

TABLE 3-32 List of lock-db Subelements

Element	Occurrences	Description
<code>path</code>	0 or 1	The path of the WebDAV lock database. If a relative path is used, it is relative to the server's <code>config</code> directory.
<code>update-interval</code>	0 or 1	The interval (in seconds) at which WebDAV lock databases are synchronized to disk. The value can be from 0.001 to 3600, or 0 to disable caching of WebDAV lock information.

See Also

- [“dav” on page 53](#)
- [“dav-collection” on page 54](#)
- [“property-db” on page 76](#)

log

The `log` element configures the logging subsystem. This element may appear zero or one time within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `log` element can contain the following subelements:

TABLE 3-33 List of Log Subelements

Element	Occurrences	Description
<code>log-stdout</code>	0 or 1	Determines whether the server logs data that applications write to <code>stdout</code> . The default value is <code>true</code> .
<code>log-stderr</code>	0 or 1	Determines whether the server logs data that applications write to <code>stderr</code> . The default value is <code>true</code> .
<code>log-virtual-server-name</code>	0 or 1	Determines whether the server includes the virtual server name in log messages. The default value is <code>false</code> .
<code>create-console</code>	0 or 1	Determines whether the server creates a console window (Windows only). The default value is <code>false</code> .
<code>log-to-console</code>	0 or 1	Determines whether the server writes log messages to the console. The default value is <code>true</code> .
<code>log-to-syslog</code>	0 or 1	Determines whether the server writes log messages to <code>syslog</code> (UNIX only) or <code>Event Viewer</code> (Windows only). The default value is <code>false</code> .
<code>date-format</code>	0 or 1	The date format for log message timestamps. The default value is <code>%d/%b/%Y:%H:%M:%S</code> . For more information, see Appendix D, "Using Time Formats."
<code>archive-suffix</code>	0 or 1	The suffix appended to rotated log file names. The default value is <code>%Y%m%d%H%M</code> .
<code>archive-command</code>	0 or 1	The command executed after the server rotates a log file. The program is passed the post-rotation file name of the log file as an argument. The value is a program command line. For example, <code><archive-command>gzip</archive-command></code> or <code><archive-command>"c:\Program Files\Perl\perl.exe" archive.pl</archive-command></code>
<code>log-level</code>	0 or 1	The log verbosity for the server. The value can be <code>finest</code> (most verbose), <code>finer</code> , <code>fine</code> , <code>info</code> , <code>warning</code> , <code>failure</code> , <code>config</code> , <code>security</code> , or <code>catastrophe</code> (least verbose).
<code>log-file</code>	0 or 1	Defines the log file for the server. The value is the file name of the log file, for example, <code>./logs/errors</code> . If a relative path is used, it is relative to the server's config directory.

See Also

- “`access-log`” on page 43
- “`access-log-buffer`” on page 43
- “`audit-accesses`” on page 46

- “event” on page 58

mail-resource

The `mail-resource` element configures a mail store resource. This element may appear zero or more times within the `server` element. For more information, see “[server](#)” on page 80.

Subelements

The `mail-resource` element can contain the following subelements:

TABLE 3-34 List of `mail-resource` Subelements

Element	Occurrences	Description
<code>enabled</code>	0 or 1	Determines whether the mail resource is enabled at runtime. The default value is <code>true</code> .
<code>jndi-name</code>	1	The JNDI name of the resource.
<code>description</code>	0 or 1	The description of the resource
<code>property</code>	0 or more	Configures optional mail-resource-specific properties. The properties are the standard JavaMail™ properties. For more details, see the JavaMail API Specification at http://java.sun.com/products/javamail/JavaMail-1.2.pdf , and “ property ” on page 75.
<code>store-protocol</code>	0 or 1	The protocol used to retrieve messages.
<code>store-protocol-class</code>	0 or 1	The storage service provider implementation for <code>store-protocol</code> . The value is a name of a class that implements <code>store-protocol</code> . The default value is <code>com.sun.mail.imap.IMAPStore</code> .
<code>transport-protocol</code>	0 or 1	The protocol used to send messages.
<code>transport-protocol-class</code>	0 or 1	The transport service provider implementation for <code>transport-protocol</code> . The value is a name of a class that implements <code>transport-protocol</code> . The default value is <code>com.sun.mail.smtp.SMTPTransport</code> .
<code>host</code>	1	The mail server host name.
<code>user</code>	1	The mail server username.
<code>from</code>	1	The email address from which the server sends email.
<code>description</code>	0 or 1	The description of the mail resource.

mime-file

The `mime-file` element defines a file that configures the MIME type mappings for the server. This element may appear zero or more times within the `server` element and zero or more times within the `virtual-server` element. For more information, see [“server” on page 80](#), and [“virtual-server” on page 97](#).

The value of this element is the file name of a MIME types file. If a relative path is used, it is relative to the server's `config` directory. This element does not contain any subelements.

For more information, see [Chapter 8, “MIME Types.”](#)

pkcs11

The `pkcs11` element configures the PKCS #11 subsystem. This element may appear zero or one time within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `pkcs11` element can contain the following subelements:

TABLE 3-35 List of `pkcs11` Subelements

Element	Occurrences	Description
<code>enabled</code>	0 or 1	Determines whether the server initializes PKCS #11 tokens, prompting for Personal Identification Numbers (PINs) as necessary. The default value is <code>true</code> if SSL is enabled and <code>false</code> if SSL is not enabled.
<code>crl-path</code>	0 or 1	The directory that contains dynamically updated CRL files. The value is the name of the directory. If a relative path is used, it is relative to the server's <code>config</code> directory.
<code>token</code>	0 or more	Configures a PKCS #11 token. For more details, see “token” on page 96 .

See Also

- [“ssl” on page 88](#)
- [“http-listener” on page 63](#)

profiler

The `profiler` element configures a JVM profiler. This element may appear zero or one time within the `jvm` element. For more information, see [“jvm” on page 67](#).

Subelements

The `profiler` element can contain the following subelements:

TABLE 3-36 List of profiler Subelements

Element	Occurrences	Description
<code>enabled</code>	0 or 1	Determines whether the profiler is enabled at runtime. The default value is <code>true</code> .
<code>class-path</code>	0 or 1	The classpath for the profiler.
<code>native-library-path</code>	0 or 1	The native library path for the profiler. The value is the operating system library path.
<code>jvm-options</code>	0 or more	The JVM options for the profiler. For more details, see (http://java.sun.com/docs/hotspot/VMOptions.html).

property

The `property` element defines a name-value pair. The effect of defining a property name-value pair depends on the context in which the property element appears as described below:

- Properties defined at the `auth-db` level configure ACL authentication databases. For more information, see “[auth-db](#)” on page 47.
- Properties defined at the `soap-auth-provider` level configure SOAP message-level authentication providers. For more information, see “[soap-auth-provider](#)” on page 87.
- Properties defined at the `auth-realm` level configure Servlet container authentication realms. For more information, see “[auth-realm](#)” on page 48.
- Properties defined at the `jdbc-resource` level configure JDBC drivers. For more information, see “[jdbc-resource](#)” on page 66.
- Properties defined at the `custom-resource` and `external-jndi-resource` levels configure JNDI resources. For more information, see “[custom-resource](#)” on page 52 and “[external-jndi-resource](#)” on page 59.
- Properties defined at the `mail-resource` level configure standard Java mail properties. For more information, see “[mail-resource](#)” on page 73.

Subelements

The `property` element can contain the following subelements:

TABLE 3-37 List of property Subelements

Element	Occurrences	Description
name	1	The name of the property.
value	1	The value of the property.
encoded	0 or 1	Determines whether the property value has been encoded using the unencode algorithm. The default value is <code>false</code> .
description	0 or 1	The description of the property.

See Also

- “[connection-creation-property](#)” on page 50
- “[connection-lease-property](#)” on page 51
- “[env-variable](#)” on page 58
- “[variable](#)” on page 96

property-db

The `property-db` element configures the WebDAV property database. This element may appear zero or one time within the `dav` element and zero or one time within the `dav-collection` element. For more information, see “[dav](#)” on page 53, and “[dav-collection](#)” on page 54.

Subelements

The `property-db` element can contain the following subelements:

TABLE 3-38 List of property-db Subelements

Element	Occurrences	Description
max-size	0 or 1	The maximum size (in bytes) of WebDAV property database files. The value can be from 0 to 2147483647, or -1 for no limit.
update-interval	0 or 1	The interval (in seconds) at which the WebDAV property databases are synchronized to disk. The value can be from 0.001 to 3600, or 0 to disable caching of WebDAV properties.

See Also

- “[dav](#)” on page 53
- “[dav-collection](#)” on page 54
- “[lock-db](#)” on page 71

qos

The `qos` element configures the Quality of Service (QoS) statistics collection subsystem. This element may appear zero or one time within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `qos` element can contain the following subelements:

TABLE 3–39 List of `qos` Subelements

Element	Occurrences	Description
<code>enabled</code>	0 or 1	Determines whether the system tracks the QOS information. The default value is <code>true</code> if <code>qos-limits</code> are enabled, and <code>false</code> if <code>qos-limits</code> are not enabled. For details, see “qos-limits” on page 77 .
<code>interval</code>	0 or 1	The interval (in seconds) over which the QOS information is averaged. The value can be from 0.001 to 3600.

See Also

[“qos-limits” on page 77](#)

qos-limits

The `qos-limits` element configures QOS (Quality of Service) limits. This element may appear zero or one time within the `server` element, and zero or one time within the `virtual-server` element. For more information, see [“server” on page 80](#), and [“virtual-server” on page 97](#).

Subelements

The `qos-limits` element can contain the following subelements:

TABLE 3–40 List of `qos-limits` Subelements

Element	Occurrences	Description
<code>enabled</code>	0 or 1	Determines whether the QOS limits are enforced at runtime. The default value is <code>true</code> .
<code>max-bps</code>	0 or 1	The maximum transfer rate (bytes per second). The value can be from 1 to 2147483647.

TABLE 3-40 List of qos-limits Subelements (Continued)

Element	Occurrences	Description
max-connections	0 or 1	The maximum number of concurrent connections. The value can be from 1 to 1048576.

See Also

[“qos” on page 77](#)

request-policy

The request-policy element configures the authentication policy requirements for web services requests. This element may appear zero or one time within the soap-auth-provider element. For more information, see [“soap-auth-provider” on page 87](#).

Subelements

The request-policy element can contain the following subelements:

TABLE 3-41 List of request-policy Subelements

Element	Occurrences	Description
auth-source	0 or 1	Specifies a requirement for message layer sender authentication. For example, username and password, or content authentication such as a digital signature.
auth-recipient	0 or 1	Specifies a requirement for message layer authentication of the receiver of a message to its sender, for example, by XML encryption. The value can be before-content or after-content.

response-policy

The response-policy element configures the authentication policy requirements for web services responses. This element may appear zero or one time within the soap-auth-provider element. For more information, see [“soap-auth-provider” on page 87](#).

Subelements

The response-policy element can contain the following subelements:

TABLE 3-42 List of response-policy Subelements

Element	Occurrences	Description
auth-source	0 or 1	Defines a requirement for message layer sender authentication. For example, username and password, or content authentication such as a digital signature.
auth-recipient	0 or 1	Defines a requirement for message layer authentication of the receiver of a message to its sender, for example, by XML encryption. The value can be before-content or after-content.

search-app

The search-app element configures the built-in search web application. This element may appear zero or one time within the virtual-server element. For more information, see [“virtual-server” on page 97](#).

Subelements

The search-app element can contain the following subelements:

TABLE 3-43 List of search-app Subelements

Element	Occurrences	Description
enabled	0 or 1	Determines whether the search application is enabled at runtime. The default value is true.
max-hits	0 or 1	The maximum number of search results to return in response to a single search query. The value can be from 0 to 10000.
uri	1	The root URI for the search web application.

See Also

- [“convert” on page 52](#)
- [“include” on page 64](#)
- [“index” on page 65](#)
- [“search-collection” on page 79](#)

search-collection

The search-collection element configures a collection of searchable documents. This element may appear zero or more times within the virtual-server element. For more information, see [“virtual-server” on page 97](#).

Subelements

The search-collection element can contain the following subelements:

TABLE 3-44 List of search-collection Subelements

Element	Occurrences	Description
enabled	0 or 1	Determines whether the collection can be searched. The default value is true.
name	1	The name that uniquely identifies the search collection.
display-name	0 or 1	The description of the search collection displayed to end users.
uri	1	The root URI for the searchable documents.
document-root	1	The file system root for the searchable documents. If a relative path is used, it is relative to the server's config directory.
path	1	The file system path where search collection meta data is stored. If a relative path is used, it is relative to the server's config directory.
index	0 or 1	Configures the document fields to be indexed. For more details, see “index” on page 65 .
convert	0 or 1	Configures the document type to be converted. For more details, see “convert” on page 52 .
include	0 or 1	Configures document types that should be included. For more details, see “include” on page 64 .
description	0 or 1	The description of the search collection.

See Also

- [“convert” on page 52](#)
- [“include” on page 64](#)
- [“index” on page 65](#)
- [“search-app” on page 79](#)

server

The server element defines a server. This is the root element, and there can be only one server element in the server.xml file.

Subelements

The server element has the following subelements:

TABLE 3-45 List of server Subelements

Element	Occurrences	Description
cluster	0 or 1	The server cluster to which the server belongs. For more details, see “cluster” on page 50 .
log	0 or 1	Configures the logging subsystem. For more details, see “log” on page 71 .
user	0 or 1	The account the server runs as (UNIX only). The value is the user account. If the server is started as root, any UNIX account can be specified. If the server is started by a non-root account, only that non-root account should be specified.
platform	0 or 1	Determines whether the server runs as a 32-bit or 64-bit process. The value can be 32 or 64.
temp-path	0 or 1	The directory where the server stores its temporary files. If a relative path is used, it is relative to the server's config directory. The directory must be owned by the account that the server runs as.
variable	0 or more	Defines a variable for use in expressions, log formats, and obj.conf parameters. For more details, see “variable” on page 96 .
localization	0 or 1	Configures localization. For more details, see “localization” on page 70 .
http	0 or 1	Configures the HTTP protocol options. For more details, see “http” on page 61 .
keep-alive	0 or 1	Configures the HTTP keep-alive subsystem. For more details, see “keep-alive” on page 69 .
thread-pool	0 or 1	Configures the HTTP request processing threads. For more details, see “thread-pool” on page 95 .
pkcs11	0 or 1	Configures the PKCS #11 subsystem. For more details, see “pkcs11” on page 74 .
stats	0 or 1	Configures the statistics collection subsystem. For more details, see “stats” on page 94 .
cgi	0 or 1	Configures the CGI subsystem. For more details, see “cgi” on page 49 .
qos	0 or 1	Configures the QOS subsystem. For more details, see “qos” on page 77 .
dns	0 or 1	Configures the server's use of DNS. For more details, see “dns” on page 56 .

TABLE 3-45 List of server Subelements (Continued)

Element	Occurrences	Description
<code>dns-cache</code>	0 or 1	Configures the DNS cache. For more details, see “dns-cache” on page 57 .
<code>file-cache</code>	0 or 1	Configures the file cache. For more details, see “file-cache” on page 60 .
<code>acl-cache</code>	0 or 1	Configures the ACL cache. For more details, see “acl-cache” on page 44 .
<code>ssl-session-cache</code>	0 or 1	Configures the SSL/TLS session cache. For more details, see “ssl-session-cache” on page 93 .
<code>access-log-buffer</code>	0 or 1	Configures the access log buffering subsystem. For more details, see “access-log-buffer” on page 43 .
<code>dav</code>	0 or 1	Configures WebDAV. For more details, see “dav” on page 53 .
<code>snmp</code>	0 or 1	Configures SNMP. For more details, see “snmp” on page 87 .
<code>qos-limits</code>	0 or 1	Configures the QOS limits for the server. For more details, see “qos-limits” on page 77 .
<code>audit-accesses</code>	0 or 1	Specifies whether authentication and authorization events are logged. The default value is <code>false</code> .
<code>jvm</code>	0 or 1	Configures JVM. For more details, see “jvm” on page 67 .
<code>servlet-container</code>	0 or 1	Configures the Servlet container. For more details, see “servlet-container” on page 83 .
<code>lifecycle-module</code>	0 or more	Configures a Java server lifecycle module. For more details, see “lifecycle-module” on page 69 .
<code>custom-resource</code>	0 or more	Configures a resource implemented by a custom class. For more details, see “custom-resource” on page 52 .
<code>external-jndi-resource</code>	0 or more	Configures a resource provided by an external JNDI repository. For more details, see “external-jndi-resource” on page 59 .
<code>jdbc-resource</code>	0 or more	Configures a JDBC resource. For more details, see “jdbc-resource” on page 66 .
<code>mail-resource</code>	0 or more	Configures a mail store. For more details, see “mail-resource” on page 73 .

TABLE 3-45 List of server Subelements (Continued)

Element	Occurrences	Description
<code>default-soap-auth-provider-name</code>	0 or 1	The name of the default SOAP message-level authentication provider. The value is the name value from a <code>soap-auth-provider</code> element. For more details, see “default-soap-auth-provider-name” on page 56
<code>soap-auth-provider</code>	0 or more	Configures a SOAP message-level authentication provider. For more details, see “soap-auth-provider” on page 87 .
<code>default-auth-realm-name</code>	0 or 1	The name of the default Servlet container authentication realm. The value is the name value from an <code>auth-realm</code> element. For more details, see “auth-realm” on page 48 .
<code>auth-realm</code>	0 or more	Configures a Servlet container authentication realm. For more details, see “auth-realm” on page 48 .
<code>default-auth-db-name</code>	0 or 1	The name of the default ACL authentication database. The value is the name value from an <code>auth-db</code> element, and the default value is <code>default</code> . For more details, see “auth-db” on page 47 .
<code>auth-db</code>	0 or more	Configures an ACL authentication database for the server. For more details, see “auth-db” on page 47 .
<code>acl-file</code>	0 or more	The ACL file that controls access to the server. The value is the name of an ACL file. For more details, see “acl-file” on page 46 .
<code>mime-file</code>	0 or more	The <code>mime.types</code> file that configures MIME mappings for the server as a whole. The value is the name of a <code>mime.types</code> file. For more details, see “mime-file” on page 74 .
<code>access-log</code>	0 or more	Configures an HTTP access log for the server. For more details, see “access-log” on page 43 .
<code>http-listener</code>	0 or more	Configures an HTTP listener. For more details, see “http-listener” on page 63 .
<code>virtual-server</code>	0 or more	Configures a virtual server. For more details, see “virtual-server” on page 97 .
<code>event</code>	0 or more	Configures a recurring event. For more details, see “event” on page 58 .

servlet-container

The `servlet-container` element configures the Servlet container. This element may appear zero or one time within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `servlet-container` element can contain the following subelements:

TABLE 3-46 List of `servlet-container` Subelements

Element	Occurrences	Description
<code>dynamic-reload-interval</code>	0 or 1	Specifies how often the server checks the deployed web applications for modifications. The value can be from 1 to 60, or 0 to disable dynamic reloading.
<code>log-level</code>	0 or 1	The log verbosity for the Servlet container. The value can be <code>finest</code> (most verbose), <code>finer</code> , <code>fine</code> , <code>info</code> , <code>warning</code> , <code>failure</code> , <code>config</code> , <code>security</code> , or <code>catastrophe</code> (least verbose).
<code>anonymous-role</code>	0 or 1	The name of the default, or anonymous role assigned to all principals. The default role is <code>ANYONE</code> .
<code>single-threaded-servlet-pool-size</code>	0 or 1	The number of Servlet instances to instantiate per <code>SingleThreadedServlet</code> . The value can be from 1 to 4096. The default value is 5.
<code>cross-context-allowed</code>	0 or 1	Determines whether request dispatchers are allowed to dispatch to another context. The default is <code>true</code> .
<code>reuse-session-id</code>	0 or 1	Determines whether any existing session ID number is reused when creating a new session for that client. The default value is <code>false</code> .
<code>encode-cookies</code>	0 or 1	Determines whether the Servlet container encodes cookie values. The default value is <code>true</code> .
<code>dispatcher-max-depth</code>	0 or 1	The maximum depth for the Servlet container allowing nested request dispatches. The value can be from 0 to 2147483647. The default value is 20.
<code>secure-session-cookie</code>	0 or 1	Controls the conditions under which the <code>JSESSIONID</code> cookie is marked secure. The value can be as follows: <ul style="list-style-type: none"> ▪ <code>dynamic</code> – Marks the cookie secure only when the request is received on a secure connection ▪ <code>true</code> – Always marks the cookie secure ▪ <code>false</code> – Never marks the cookie secure The default value is <code>dynamic</code> .

See Also

- [“auth-realm” on page 48](#)
- [“default-auth-realm-name” on page 56](#)

- “jvm” on page 67
- “single-sign-on” on page 86
- “web-app” on page 99

session-replication

The `session-replication` element configures Servlet session replication within a server cluster. This element may appear zero or one time within the `cluster` element, and zero or one time within the `instance` element. For more information, see “[cluster](#)” on page 50, and “[instance](#)” on page 65.

Subelements

The `session-replication` element can contain the following subelements:

TABLE 3-47 List of `session-replication` Subelements

Element	Occurrences	Description
<code>enabled</code>	0 or 1	Determines whether the session replication is enabled at runtime. The default value is <code>true</code> .
<code>port</code>	0 or 1	Specifies the port on which the server will listen. The default port number is 1099.
<code>instance-id</code>	0 or 1	(Only applicable at the instance level.) The value that uniquely identifies the instance for use in cookies.
<code>key</code>	0 or 1	(Only applicable at the cluster level.) The shared secret which members of the cluster use to authenticate to each other. The value of this subelement should be in text format.
<code>encrypted</code>	0 or 1	(Only applicable at the cluster level.) Determines whether the session data is encrypted prior to replication. The default value is <code>false</code> .
<code>protocol</code>	0 or 1	(Only applicable at the cluster level.) The protocol used for session replication. The value can be <code>http</code> or <code>jrmf</code> .
<code>async</code>	0 or 1	(Only applicable at the cluster level.) Determines whether session replication is asynchronous from HTTP request/response processing. The default value is <code>false</code> .
<code>getAttribute-triggers-replication</code>	0 or 1	(Only applicable at the cluster level.) Determines whether a call to the <code>HttpSession.getAttribute</code> method should cause a session to be backed up. The default value is <code>true</code> .

TABLE 3-47 List of session-replication Subelements (Continued)

Element	Occurrences	Description
replica-discovery-max-hops	0 or 1	(Only applicable at the cluster level.) The maximum number of instances that should be contacted while attempting to find the backup of a session. The value can be from 1 to 2147483647, or -1 for no limit.
startup-discovery-timeout	0 or 1	(Only applicable at the cluster level.) The maximum time (in seconds) that an instance spends trying to contact its designated backup instance. The value can be from 0.001 to 3600.
cookie-name	0 or 1	(Only applicable at the cluster level.) The name of the cookie that tracks which instance owns a session.
cipher	0 or 1	(Only applicable at the cluster level.) The value of a JCE cipher. JCE ciphers are specified using the form algorithm/mode/padding. The value should be in text format. The default value is AES/CBC/PKCS5Padding.

single-sign-on

The `single-sign-on` element configures a single authentication mapping across multiple Java web applications sharing the same realm. This element may appear zero or one time within the `virtual-server` element. For more information, see [“virtual-server” on page 97](#).

Subelements

The `single-sign-on` element can contain the following subelements:

TABLE 3-48 List of single-sign-on Subelements

Element	Occurrences	Description
enabled	0 or 1	Determines whether the single-sign-on feature is enabled at runtime. The default value is <code>false</code> .
idle-timeout	0 or 1	The timeout (in seconds) after which a user's single sign-on records becomes eligible for purging if no activity is seen. The value can be from 0.001 to 3600, or -1 for no timeout. The default value is <code>300</code> seconds.

See Also

- [“servlet-container” on page 83](#)
- [“web-app” on page 99](#)

snmp

The `snmp` element configures the server's SNMP subagent. This element may appear zero or more times within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `snmp` element can contain the following subelements:

TABLE 3-49 List of `snmp` Subelements

Element	Occurrences	Description
<code>enabled</code>	0 or 1	Determines whether SNMP is enabled at runtime. The default value is <code>true</code> .
<code>master-host</code>	0 or 1	The network address of the SNMP master agent. The value is a host name or IP address.
<code>description</code>	1	The description of the server. The value should be in text format.
<code>organization</code>	1	The name of the organization responsible for the server. The value should be in text format.
<code>location</code>	1	The location of the server. The value should be in text format.
<code>contact</code>	1	The contact information of the person responsible for the server. The value should be in text format.

See Also

[“stats” on page 94](#)

soap-auth-provider

The `soap-auth-provider` element configures a SOAP message-level authentication provider for web services. This element may appear zero or more times within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `soap-auth-provider` element can contain the following subelements:

TABLE 3-50 List of soap-auth-provider Subelements

Element	Occurrences	Description
name	1	The name that uniquely identifies the SOAP message-level authentication provider for use in default-soap-auth-provider-name and sun-web.xml.
class	1	The class that implements the provider realm. The value is a name of a class that implements javax.security.auth.XXX.
request-policy	0 or 1	Configures the authentication policy requirements for requests. For more details, see “request-policy” on page 78 .
response-policy	0 or 1	Configures the authentication policy requirements for responses. For more details, see “response-policy” on page 78 .
property	0 or more	Configures the optional provider-specific properties. For more details, see “property” on page 75 .

ssl

The `ssl` element configures the SSL/TLS settings. This element may appear zero or one time within the `http-listener` element. For more information, see [“http-listener” on page 63](#).

Subelements

The `ssl` element can contain the following subelements:

TABLE 3-51 List of ssl Subelements

Element	Occurrences	Description
enabled	0 or 1	Determines whether SSL/TLS is enabled at runtime. The default value is <code>true</code> .
server-cert-nickname	0 or more	The nickname of the certificate that server presents to the clients. You can specify zero or one RSA certificates, plus zero or one ECC certificates.
ssl2	0 or 1	Determines whether SSL2 connections are accepted. The default value is <code>false</code> .
ssl3	0 or 1	Determines whether SSL3 connections are accepted. The default value is <code>true</code> .
tls	0 or 1	Determines whether TLS connections are accepted. The default value is <code>true</code> .

TABLE 3-51 List of `ssl` Subelements (Continued)

Element	Occurrences	Description
<code>tls-rollback-detection</code>	0 or 1	Determines whether the server detects and blocks TLS version rollback attacks. The default value is <code>true</code> .
<code>ssl2-ciphers</code>	0 or 1	Configures the SSL2 cipher suites. For more details, see “ssl2-ciphers” on page 89 .
<code>ssl3-tls-ciphers</code>	0 or 1	Configures the SSL3 and TLS cipher suites. For more details, see “ssl3-tls-ciphers” on page 90 .
<code>client-auth</code>	0 or 1	The method of client certificate authentication. The value can be <code>required</code> , <code>optional</code> , or <code>false</code> .
<code>client-auth-timeout</code>	0 or 1	The timeout (in seconds) after which client authentication handshake fails. The value can be from 0.001 to 3600.
<code>max-client-auth-data</code>	0 or 1	The maximum amount of application-level data to buffer during a client authentication handshake. The value can be from 0 to 2147483647.

See Also

- [“http-listener” on page 63](#)
- [“pkcs11” on page 74](#)
- [“ssl2-ciphers” on page 89](#)
- [“ssl3-tls-ciphers” on page 90](#)
- [“ssl-session-cache” on page 93](#)

ssl2-ciphers

The `ssl2-ciphers` element configures SSL2 cipher suites. This element may appear zero or one time within the `ssl` element. For more information, see [“ssl” on page 88](#).

Subelements

The `ssl2-ciphers` element can contain the following subelements:

TABLE 3-52 List of `ssl2-ciphers` Subelements

Element	Occurrences	Description
<code>SSL_RC4_128_WITH_MD5</code>	0 or 1	Determines whether the <code>SSL_RC4_128_WITH_MD5</code> cipher suite is enabled at runtime. The default value is <code>true</code> .

TABLE 3-52 List of `ssl2-ciphers` Subelements (Continued)

Element	Occurrences	Description
<code>SSL_RC4_128_EXPORT40_WITH_MD5</code>	0 or 1	Determines whether the <code>SSL_RC4_128_EXPORT40_WITH_MD5</code> cipher suite is enabled at runtime. The default value is <code>true</code> .
<code>SSL_RC2_128_CBC_WITH_MD5</code>	0 to 1	Determines whether the <code>SSL_RC2_128_CBC_WITH_MD5</code> cipher suite is enabled at runtime. The default value is <code>true</code> .
<code>SSL_RC2_128_CBC_EXPORT40_WITH_MD5</code>	0 or 1	Determines whether the <code>SSL_RC2_128_CBC_EXPORT40_WITH_MD5</code> cipher suite is enabled at runtime. The default value is <code>true</code> .
<code>SSL_DES_64_CBC_WITH_MD5</code>	0 to 1	Determines whether the <code>SSL_DES_64_CBC_WITH_MD5</code> cipher suite is enabled at runtime. The default value is <code>true</code> .
<code>SSL_DES_192_EDE3_CBC_WITH_MD5</code>	0 to 1	Determines whether the <code>SSL_DES_192_EDE3_CBC_WITH_MD5</code> cipher suite is enabled at runtime. The default value is <code>true</code> .

See Also

- “`http-listener`” on page 63
- “`pkcs11`” on page 74
- “`ssl`” on page 88
- “`ssl3-tls-ciphers`” on page 90
- “`ssl-session-cache`” on page 93

ssl3-tls-ciphers

The `ssl3-tls-ciphers` element configures SSL3 and TLS cipher suites. This element may appear zero or one time within the `ssl` element. For more information, see “`ssl`” on page 88.

Subelements

The `ssl3-tls-ciphers` element can contain the following subelements:

TABLE 3-53 List of `ssl3-tls-ciphers` Subelements

Element	Occurrences	Description
<code>SSL_RSA_WITH_RC4_128_MD5</code>	0 or 1	Determines whether the <code>SSL_RSA_WITH_RC4_128_MD5</code> cipher suite is enabled at runtime. The default value is <code>true</code> .

TABLE 3-53 List of `ssl3-tls-ciphers` Subelements (Continued)

Element	Occurrences	Description
<code>SSL_RSA_WITH_RC4_128_SHA</code>	0 or 1	Determines whether the <code>SSL_RSA_WITH_RC4_128_SHA</code> cipher suite is enabled at runtime. The default value is <code>true</code> .
<code>SSL_RSA_WITH_3DES_EDE_CBC_SHA</code>	0 or 1	Determines whether the <code>SSL_RSA_WITH_3DES_EDE_CBC_SHA</code> cipher suite is enabled at runtime. The default value is <code>true</code> .
<code>SSL_RSA_WITH_DES_CBC_SHA</code>	0 or 1	Determines whether the <code>SSL_RSA_WITH_DES_CBC_SHA</code> cipher suite is enabled at runtime. The default value is <code>true</code> .
<code>SSL_RSA_EXPORT_WITH_RC4_40_MD5</code>	0 or 1	Determines whether the <code>SSL_RSA_EXPORT_WITH_RC4_40_MD5</code> cipher suite is enabled at runtime. The default value is <code>true</code> .
<code>SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5</code>	0 or 1	Determines whether the <code>SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5</code> cipher suite is enabled at runtime. The default value is <code>true</code> .
<code>SSL_RSA_WITH_NULL_MD5</code>	0 or 1	Determines whether the <code>SSL_RSA_WITH_NULL_MD5</code> cipher suite is enabled at runtime. The default value is <code>false</code> .
<code>SSL_RSA_WITH_NULL_SHA</code>	0 or 1	Determines whether the <code>SSL_RSA_WITH_NULL_SHA</code> cipher suite is enabled at runtime. The default value is <code>false</code> .
<code>SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA</code>	0 or 1	Determines whether the <code>SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA</code> cipher suite is enabled at runtime. The default value is <code>true</code> .
<code>SSL_RSA_FIPS_WITH_DES_CBC_SHA</code>	0 or 1	Determines whether the <code>SSL_RSA_FIPS_WITH_DES_CBC_SHA</code> cipher suite is enabled at runtime. The default value is <code>true</code> .
<code>TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA</code>	0 or 1	Determines whether the <code>TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA</code> cipher suite is enabled at runtime. The default value is <code>true</code> .
<code>TLS_ECDH_RSA_WITH_AES_128_CBC_SHA</code>	0 or 1	Determines whether the <code>TLS_ECDH_RSA_WITH_AES_128_CBC_SHA</code> cipher suite is enabled at runtime. The default value is <code>false</code> .
<code>TLS_ECDH_RSA_WITH_RC4_128_SHA</code>	0 or 1	Determines whether the <code>TLS_ECDH_RSA_WITH_RC4_128_SHA</code> cipher suite is enabled at runtime. The default value is <code>false</code> .

TABLE 3-53 List of ssl3-tls-ciphers Subelements *(Continued)*

Element	Occurrences	Description
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA	0 or 1	Determines whether the TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA cipher suite is enabled at runtime. The default value is false.
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA	0 or 1	Determines whether the TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA cipher suite is enabled at runtime. The default value is false.
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	0 or 1	Determines whether the TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA cipher suite is enabled at runtime. The default value is false.
TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA	0 or 1	Determines whether the TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA cipher suite is enabled at runtime. The default value is true.
TLS_RSA_EXPORT1024_WITH_RC4_56_SHA	0 or 1	Determines whether the TLS_RSA_EXPORT1024_WITH_RC4_56_SHA cipher suite is enabled at runtime. The default value is true.
TLS_RSA_WITH_AES_128_CBC_SHA	0 or 1	Determines whether the TLS_RSA_WITH_AES_128_CBC_SHA cipher suite is enabled at runtime. The default value is true.
TLS_RSA_WITH_AES_256_CBC_SHA	0 or 1	Determines whether the TLS_RSA_WITH_AES_256_CBC_SHA cipher suite is enabled at runtime. The default value is true.
TLS_ECDHE_ECDSA_WITH_NULL_SHA	0 or 1	Determines whether the TLS_ECDHE_ECDSA_WITH_NULL_SHA cipher suite is enabled at runtime. The default value is false.
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	0 or 1	Determines whether the TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA cipher suite is enabled at runtime. The default value is false.
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	0 or 1	Determines whether the TLS_ECDHE_ECDSA_WITH_RC4_128_SHA cipher suite is enabled at runtime. The default value is false.

TABLE 3-53 List of `ssl3-tls-ciphers` Subelements *(Continued)*

Element	Occurrences	Description
<code>TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA</code>	0 or 1	Determines whether the <code>TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA</code> cipher suite is enabled at runtime. The default value is <code>false</code> .
<code>TLS_ECDHE_RSA_WITH_NULL_SHA</code>	0 or 1	Determines whether the <code>TLS_ECDHE_RSA_WITH_NULL_SHA</code> cipher suite is enabled at runtime. The default value is <code>false</code> .
<code>TLS_ECDHE_RSA_WITH_RC4_128_SHA</code>	0 or 1	Determines whether the <code>TLS_ECDHE_RSA_WITH_RC4_128_SHA</code> cipher suite is enabled at runtime. The default value is <code>false</code> .
<code>TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA</code>	0 or 1	Determines whether the <code>TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA</code> cipher suite is enabled at runtime. The default value is <code>false</code> .
<code>TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA</code>	0 or 1	Determines whether the <code>TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA</code> cipher suite is enabled at runtime. The default value is <code>false</code> .

See Also

- [“http-listener” on page 63](#)
- [“pkcs11” on page 74](#)
- [“ssl” on page 88](#)
- [“ssl3-tls-ciphers” on page 90](#)
- [“ssl-session-cache” on page 93](#)

ssl-session-cache

The `ssl-session-cache` element configures the SSL/TLS session cache. This element may appear zero or one time within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `ssl-session-cache` element can contain the following subelements:

TABLE 3-54 List of `ssl-session-cache` Subelements

Element	Occurrences	Description
<code>enabled</code>	0 or 1	Determines whether the server caches SSL/TLS sessions. The default value is <code>true</code> .
<code>max-entries</code>	0 or 1	The maximum number of SSL/TLS sessions the server will cache. The value can be from 1 to 524288.
<code>max-ssl2-session-age</code>	0 or 1	The maximum amount of time to cache an SSL2 session. The value can be from 5 to 100.
<code>max-ssl3-tls-session-age</code>	0 or 1	The maximum amount of time to cache an SSL3/TLS session. The value can be from 5 to 86400.

See Also

- “`http-listener`” on page 63
- “`pkcs11`” on page 74
- “`ssl`” on page 88
- “`ssl2-ciphers`” on page 89
- “`ssl3-tls-ciphers`” on page 90

stats

The `stats` element configures the statistics collection subsystem. This element may appear zero or one time within the `server` element. For more information, see “`server`” on page 80.

Subelements

The `stats` element can contain the following subelements:

TABLE 3-55 List of `stats` Subelements

Element	Occurrences	Description
<code>enabled</code>	0 or 1	Determines whether the server collects statistics. The default value is <code>true</code> .
<code>interval</code>	0 or 1	Interval (in seconds) at which statistics are updated. The value can be from 0.001 to 3600.
<code>profiling</code>	0 or 1	Determines whether the performance buckets, used to track NSAPI function execution time, are enabled at runtime. The default value is <code>true</code> .

See Also

[“snmp” on page 87](#)

thread-pool

The `thread-pool` element configures the threads used to process HTTP requests. This element may appear zero or one time within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `thread-pool` element can contain the following subelements:

TABLE 3-56 List of thread-pool Subelements

Element	Occurrences	Description
<code>min-threads</code>	0 or 1	The minimum number of HTTP request processing threads. The value can be from 1 to 4096.
<code>max-threads</code>	0 or 1	The maximum number of HTTP request processing threads. The value can be from 1 to 4096.
<code>stack-size</code>	0 or 1	The stack size (in bytes) for HTTP request processing threads. The value can be from 8192 to 67108864.
<code>queue-size</code>	0 or 1	The maximum number of concurrent HTTP connections that can be queued waiting for processing. The value can be from 1 to 1048576.

See Also

- [“http” on page 61](#)
- [“keep-alive” on page 69](#)

time

The `time` element configures the time when an event will occur. This element may appear zero or more times within the `event` element. For more information, see [“event” on page 58](#).

Subelements

The `time` element can contain the following subelements:

TABLE 3-57 List of time Subelements

Element	Occurrences	Description
time-of-day	1	The time when the event will occur. The value should be in the hh:mm format.
day-of-week	0 or 1	The day of the week. The value can be Sun, Mon, Tue, Wed, Thu, Fri, or Sat.
day-of-month	0 or 1	The day of month. The value can be from 1 to 31.
month	0 or 1	The name of the month. The value can be Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, or Dec.

token

The `token` element configures a PKCS #11 token. This element may appear zero or more times within the `pkcs11` element. For more information, see [“pkcs11” on page 74](#).

Subelements

The `token` element can contain the following subelements:

TABLE 3-58 List of token Subelements

Element	Occurrences	Description
enabled	0 or 1	Determines whether the server initializes this PKCS #11 token, prompting for a PIN if necessary. The default value is <code>true</code> .
name	1	The name of the token. The server's built-in token is named <code>internal</code> .
pin	0 or 1	The PIN required to initialize the token.

variable

The `variable` element defines a variable for use in expressions, log formats, and `obj.conf` parameters. This element may appear zero or more times within the `server` element, and zero or more times within the `virtual-server` element. For more information, see [“server” on page 80](#), and [“virtual-server” on page 97](#).

Subelements

The `variable` element can contain the following subelements:

TABLE 3-59 List of variable Subelements

Element	Occurrences	Description
name	1	The name of the variable. The value should be in text format.
value	1	The value of the variable. The value should be in text format.
description	0 or 1	The description of the variable. The value should be in text format.

See Also

- [“env-variable” on page 58](#)
- [“property” on page 75](#)
- [Appendix A, “Using Variables, Expressions, and String Interpolation”](#)

virtual-server

The `virtual-server` element configures an HTTP virtual server. Each server would typically have at least one virtual server. This element may appear zero or more times within the `server` element. For more information, see [“server” on page 80](#).

Subelements

The `virtual-server` element can contain the following subelements:

TABLE 3-60 List of virtual-server Subelements

Element	Occurrences	Description
enabled	0 or 1	Determines whether the virtual server is enabled at runtime. The default value is <code>true</code> .
name	1	A name that uniquely identifies the virtual server.
http-listener-name	0 or more	The name of an HTTP listener associated with one or more of the virtual server's host names. The value is the name from an <code>http-listener</code> element. For more details, see “http-listener” on page 63 .
host	0 or more	The host name that the virtual server services. Host comparisons are not case sensitive. The value can be a host name or a wildcard pattern. For more information on wildcards, see Appendix B, “Using Wildcard Patterns”
canonical-server-name	0 or 1	The canonical name of the virtual server. Requests using a different name will be redirected to the canonical name. The value is a host name or URL prefix.

TABLE 3-60 List of virtual-server Subelements (Continued)

Element	Occurrences	Description
<code>acl-file</code>	0 or more	The name of the ACL file that controls access to the virtual server.
<code>mime-file</code>	0 or more	The <code>mime.types</code> file that configures MIME mappings for the virtual server.
<code>object-file</code>	1	The <code>obj.conf</code> file that controls request processing for the virtual server.
<code>default-object-name</code>	0 or 1	The name of the root <code>obj.conf</code> object. The default value is <code>default</code> .
<code>document-root</code>	1	The document root for the virtual server.
<code>localization</code>	0 or 1	Configures localization. For more details, see “localization” on page 70 .
<code>qos-limits</code>	0 or 1	Configures QOS limits for the virtual server. For more details, see “qos-limits” on page 77 .
<code>search-app</code>	0 or 1	Configures the built-in search web application for the virtual server. For more details, see “search-app” on page 79 .
<code>access-log</code>	0 or more	Configures an HTTP access log for the virtual server. For more details, see “access-log” on page 43 .
<code>auth-db</code>	0 or more	Configures an ACL authentication database for the virtual server. For more details, see “auth-db” on page 47 .
<code>search-collection</code>	0 or more	Configures a collection of searchable documents for the virtual server. For more details, see “search-collection” on page 79 .
<code>dav-collection</code>	0 or more	Configures a WebDAV collection for the virtual server. For more details, see “dav-collection” on page 54 .
<code>web-app</code>	0 or more	Configures the Java web application mappings for the virtual server. For more details, see “web-app” on page 99 .
<code>log-file</code>	0 or 1	The log file for the virtual server. The value is the log file name, for example, <code>../logs/errors</code> .
<code>variable</code>	0 or more	Defines an <code>obj.conf</code> variable for the virtual server. For more details, see “variable” on page 96 .
<code>description</code>	0 or 1	The description of the virtual server.
<code>single-sign-on</code>	0 or 1	Configures single sign-on for Java web applications within the virtual server. For more details, see “single-sign-on” on page 86 .

See Also

- “http” on page 61
- “http-listener” on page 63
- “keep-alive” on page 69
- Chapter 6, “Syntax and Use of obj.conf”

web-app

The `web-app` element configures a Java web application mapping. This element may appear zero or more times within the `virtual-server` element. For more information, see “`virtual-server`” on page 97.

Subelements

The `web-app` element can contain the following subelements:

TABLE 3-61 List of `web-app` Subelements

Element	Occurrences	Description
<code>enabled</code>	0 or 1	Determines whether the web application is enabled at runtime. The default value is <code>true</code> .
<code>uri</code>	1	The root URI for the web application.
<code>path</code>	1	The path where the web application is stored. If a relative path is used, it is relative to the server's <code>config</code> directory.
<code>description</code>	0 or 1	The description of the web application.

See Also

- “`servlet-container`” on page 83
- “`single-sign-on`” on page 86

Syntax and Use of `magnus.conf`

The `magnus.conf` file contains NSAPI plug-in initialization directives and settings that control the way NSAPI plug-ins are run. The `magnus.conf` file is located in the `instance_dir/config` directory.

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

This chapter lists the settings that can be specified in `magnus.conf`.

- “ChildRestartCallback” on page 103
- “Init Directives” on page 103
- “KernelThreads” on page 104
- “NativePoolMaxThreads” on page 104
- “NativePoolMinThreads” on page 104
- “NativePoolQueueSize” on page 105
- “NativePoolStackSize” on page 105
- “TerminateTimeout” on page 105
- “Umask” on page 106

Editing `magnus.conf`

You can add directives or edit existing directives in `magnus.conf`. Be very careful when editing this file. Simple mistakes can make the server fail to operate correctly. When editing `magnus.conf`, use the `wadm` command `get-config-file` to pull a copy of the file, edit the file locally, then use `set-config-file` to put the edited file back. You must restart the server after editing `magnus.conf`.

Directives in `magnus.conf` either set a value or invoke a Server Application Function (SAF).

Parameters

For pre-defined SAFs, the number and names of parameters depend on the function. The order of parameters on the line is not important.

Case Sensitivity

Items in the `magnus.conf` file are case-sensitive including function names, parameter names, parameter values, and path names.

Separators

The C language allows function names to be composed only of letters, digits, and underscores. You may use the hyphen (-) character in the configuration file in place of underscore (_) for your C code function names. This is only true for function names.

Quotation Marks

Quotation marks (") are only required around the value strings when there is a space in the string. Otherwise, they are optional. Each open quotation mark must be matched by a closed quotation mark.

Spaces

- Spaces are not allowed at the beginning of a line except when continuing the previous line.
- Spaces are not allowed before or after the equal (=) sign that separates the name and value.
- Spaces are not allowed at the end of a line or on a blank line.

Line Continuation

A long line may be continued on the next line by beginning the next line with a space or tab.

Path Names

Always use forward slashes (/) rather than backslashes (\) in path names on the Windows platform. A backslash escapes the next character.

Comments

Comments begin with a pound (#) sign. If you manually add comments to `obj.conf`, then use the Admin Console or CLI to make changes to your server, your comments are overwritten when `obj.conf` is updated.

ChildRestartCallback

The `ChildRestartCallback` directive determines whether the Web Server calls the NSAPI functions that were registered using the `daemon_atrestart` function.

If you set `ChildRestartCallback` to `on`, the server calls the registered NSAPI functions when it shuts down or restarts. If you set `ChildRestartCallback` to `off`, the server never calls the registered NSAPI functions. If you do not explicitly set a value for `ChildRestartCallback`, the server calls the registered NSAPI functions when it shuts down or restarts only if all NSAPI Server Application Functions (SAFs) complete request processing before the `TerminateTimeout` timeout elapses.

Syntax

```
ChildRestartCallback value
```

where *value* is `on` or `off`.

Init Directives

The `Init` directives load and initialize server modules and NSAPI plug-ins.

Syntax

```
Init fn="function" param1="value1" ...paramN="valueN"
```

In this syntax:

- *function* is the name of a predefined `Init` SAF or the name of an `Init` SAF implemented by a NSAPI plug-in. For a list of predefined `Init` SAFs, see [Chapter 5, “Predefined SAFs in `magnus.conf`”](#)
- *param1="value1" ...paramN="valueN"* name-value pairs define SAF-specific configuration parameters.

KernelThreads

(Windows only) On Windows, the Web Server supports both kernel-level and user-level threads. User threads are scheduled by Netscape Portable Runtime (NSPR) within the process, whereas kernel threads are scheduled by the host's operating system. Usually, the standard debugger and compiler are intended for use with kernel-level threads.

If you set `KernelThreads` to 1 (on), the server uses the kernel-level threads. If you set `KernelThreads` to 0 (off), the server uses the user-level threads, which might improve performance.

Syntax

`KernelThreads` *value*

where *value* is 0 or 1.

Default

0 (off)

NativePoolMaxThreads

(Windows only) The `NativePoolMaxThreads` directive determines the maximum number of threads in the native (kernel) thread pool.

Default

0

NativePoolMinThreads

(Windows only) The `NativePoolMinThreads` directive determines the minimum number of threads in the native (kernel) thread pool.

Default

1

NativePoolQueueSize

(Windows only) The `NativePoolQueueSize` directive determines the number of threads that can wait in a queue for the thread pool. If all threads in the pool are busy, the next request-handling thread that needs to use a thread in the native pool must wait in a queue.

If a queue is full, the next request-handling thread that tries to get in the queue is rejected and the server returns a busy response to the client. The server is then free to handle another incoming request.

Default

0

NativePoolStackSize

(Windows only) The `NativePoolStackSize` directive determines the stack size of each thread in the native (kernel) thread pool.

Default

0

TerminateTimeout

The `TerminateTimeout` directive specifies the time (in seconds) that the server waits for NSAPI SAFs to complete the processing of any active requests before it shuts down. Increase the `TerminateTimeout` value to allow in-progress HTTP transactions to complete gracefully, or shorten the value to allow the server to shut down more quickly.

Syntax

`TerminateTimeout` *value*

where *value* is an interval in seconds.

Default

30

Umask

(UNIX only) The `Umask` directive specifies the maximum file permissions granted by NSAPI functions that create files and directories.

Syntax

`Umask permissions`

where *permissions* is a UNIX file permissions value in octal notation.

Default

`0777`

Predefined SAFs in `magnus.conf`

This chapter lists the `Init` Server Application Functions (SAF) that you can specify in `magnus.conf`. `Init` SAFs load and initialize server modules and NSAPI plug-ins.

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

The following topics are described in detail in this chapter:

- [“Init SAFs” on page 107](#)
- [“Common SAFs” on page 116](#)
- [“Deprecated Init SAFs” on page 117](#)

Init SAFs

The `Init` directives are executed only once at server startup. Each `Init` directive has an `fn` parameter that specifies which `Init` SAF to invoke.

Each `Init` directive has an optional `LateInit` parameter. For the UNIX platform, if `LateInit` is set to `Yes`, the function is executed by the child process after it is forked from the parent. If `LateInit` is set to `No` or is not provided, the function is executed by the parent process before the fork. For the Windows platform, `LateInit` functions are executed later than functions that do not have the `LateInit` parameter.

When the server is started by a root user but runs as another user, perform all activities that must be performed as the user root (such as writing to a root-owned file) before the fork. Functions that create threads, with the exception of `thread-pool-init`, should be executed after the fork, that is, the relevant `Init` directive should have `LateInit=yes` set.

This section describes the following SAFs:

- “cindex-init” on page 108
- “define-perf-bucket” on page 109
- “init-dav” on page 110
- “init-filter-order” on page 110
- “init-request-limits” on page 112
- “init-uhome” on page 112
- “load-modules” on page 113
- “pool-init” on page 114
- “register-http-method” on page 114
- “thread-pool-init” on page 115

cindex-init

The `cindex-init` function sets the default settings for common indexing. Common indexing (also known as fancy indexing) is performed by the Service function `index-common`. Indexing occurs:

- When the requested URL translates to a directory that does not contain an index file or home page.
- If no index file or home page has been specified.

This function is applicable in `Init-class` directives. In common (fancy) indexing, the directory list shows the name, last modified date, size, and description of each indexed file or directory.

Parameters

The following table describes the `cindex-init` parameters.

TABLE 5-1 cindex-init Parameters

Parameter	Description
<code>opts</code>	(Optional) String of letters specifying the options to activate. Currently there is only one possible option: <code>s</code> instructs the server to scan each HTML file in the directory that is being indexed for the contents of the HTML <code>TITLE</code> tag. The <code>TITLE</code> tag must be within the first 255 characters of the file. This option is <code>off</code> by default. The search for <code>TITLE</code> is not case-sensitive.

TABLE 5-1 `cindex-init` Parameters (Continued)

Parameter	Description
<code>widths</code>	<p>(Optional) Specifies the width of each column in the indexing display. The string is a comma-separated list of numbers that specify the column widths in characters for name, last-modified date, size, and description respectively.</p> <p>The default value for the <code>widths</code> parameter is 22, 18, 8, and 33.</p> <p>The final three values (corresponding to last-modified date, size, and description) can each be set to 0 to turn off the display for that column. The name column cannot be turned off.</p> <p>The minimum size of a column (if the value is non-zero) is specified by the length of its title. For example, the minimum size of the date column is 5 (the length of the date plus one space). If you set a non-zero value for a column that is less than the length of its title, the width defaults to the minimum required to display the title.</p>
<code>timezone</code>	<p>(Optional) Determines whether the last-modified time is shown in local time or in Greenwich Mean Time. The values are <code>GMT</code> or <code>local</code>. The default is <code>local</code>.</p>
<code>format</code>	<p>(Optional) Determines the format of the last modified date. It uses the format specification for the UNIX function <code>strftime()</code>.</p> <p>The default is <code>%d-%b-%Y %H:%M</code>.</p>
<code>ignore</code>	<p>(Optional) Specifies a wildcard pattern for file names that the server should ignore while indexing. By default, file names starting with a period (.) are always ignored. For more information, see Appendix B, "Using Wildcard Patterns."</p>
<code>icon-uri</code>	<p>(Optional) Specifies the URI prefix the <code>index-common</code> function uses when generating URLs for file icons (.gif files). By default, it is <code>/mc-icons/</code>.</p> <p>If <code>icon-uri</code> is different from the default, the <code>px2dir</code> function in the <code>NameTrans</code> directive must be changed so that the server can find these icons.</p>

Example

```
Init fn="cindex-init" widths="50,1,1,0"
Init fn="cindex-init" ignore="*private*"
Init fn="cindex-init" widths="22,0,0,50"
```

define-perf-bucket

The `define-perf-bucket` function creates a performance bucket, which you can use to measure the performance of SAFs in `obj.conf`.

This function is applicable in `Init`-class directives. For more information about performance buckets, see *Sun Java System Web Server 7.0 Update 3 Performance Tuning, Sizing, and Scaling Guide*.

Parameters

The following table describes the `define-perf-bucket` parameters.

TABLE 5-2 `define-perf-bucket` Parameters

Parameter	Description
<code>name</code>	The name of the bucket, for example, <code>cgi-bucket</code>
<code>description</code>	The description of what the bucket measures, for example, <code>CGI Stats</code>

Example

```
Init fn="define-perf-bucket" name="cgi-bucket" description="CGI Stats"
```

init-dav

The `init-dav` function performs initialization tasks to load the WebDAV plug-in. This function is applicable in `Init`-class directives.

Example

```
Init fn="load-modules"
      shlib="libdavplugin.so"
      funcs="init-dav,ntrans-dav,service-dav"
Init fn="init-dav"
```

init-filter-order

The `init-filter-order` function controls the position of specific filters within the filter stacks. For example, you can use `init-filter-order` to ensure that a filter that converts outgoing XML to XHTML is inserted above a filter that converts outgoing XHTML to HTML.

This function is applicable in `Init`-class directives.

Filters that appear higher in the filter stack are given the first opportunity to process outgoing data, and filters that appear lower in the filter stack are given the first opportunity to process incoming data.

The appropriate position of a specific filter within the filter stack is defined by the filter developer. For example, filters that translate content from XML to HTML are placed higher in the filter stack than filters that compress data for transmission. Filter developers use the `filter-create` function to define the filter's position in the filter stack. You can use `init-filter-order` to override the position defined by the filter developer.

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

For example, consider two filters, `xhtml-to-html` and `xml-to-xhtml`, which convert XHTML to HTML and XML to XHTML, respectively. As both these filters transform data from one format to another, they may be defined to occupy the same position in the filter stack. To transform XML documents to XHTML and then to HTML before sending the data to the client, `Output fn="insert-filter"` directives in `obj.conf` should appear in the following order:

```
Output fn="insert-filter" filter="xhtml-to-html"
Output fn="insert-filter" filter="xml-to-xhtml"
```

In general, you should use the order of `Input fn="insert-filter"` and `Output fn="insert-filter"` directives in `obj.conf` to control the position of filters in the filter stack. `init-filter-order` should only be used to address specific filter interoperability problems.

Note – The `load-module` function that creates the filters should be called before `init-filter-order` attempts to order them.

Parameters

The following table describes the `init-filter-order` parameter.

TABLE 5-3 `init-filter-order` Parameter

Parameter	Description
<code>filters</code>	A comma-separated list of filters in the order they should appear within a filter stack, listed from highest to lowest

Example

```
Init fn="init-filter-order" filters="xml-to-xhtml,xhtml-to-html,http-compression"
```

init-request-limits

The `init-request-limits` function works with the `obj.conf` function `check-request-limits` to monitor incoming requests with a given attribute. `check-request-limits` maintains a table of monitored values. `init-request-limits` purges existing entries in that table according to the timeout. This function is not required unless you want to override the default value for the purge timeout in `check-request-limits`. For more information, see “[check-request-limits](#)” on page 155. The default is 300 seconds (five minutes). This function is applicable in `Init-class` directives.

Parameters

The following table describes the `init-request-limits` parameter.

TABLE 5-4 `init-request-limits` Parameter

Parameter	Description
<code>timeout</code>	<p>(Optional) Sets the time in seconds after which to purge entries tracked by <code>check-request-limits</code>. The default is 300 seconds (five minutes).</p> <p>An optimal value for <code>timeout</code> depends not only on your performance and memory requirements but also on the <code>check-request-limits</code> rules you are using. When using rules containing, for example, <code>monitor="\$ip"</code> on a busy public web site, new buckets are created and kept for every client IP accessing the server. Because this setting potentially creates a very large number of buckets, the expiration should be short enough that unused entries are purged in a reasonable time.</p> <p>However, to avoid removing and re-creating buckets for the same client, do not set a <code>timeout</code> that is shorter than the typical or expected client session.</p> <p>If you do not use any dynamic bucket names (that is, if all monitored values and bucket are fixed strings instead of variables, or you never specify <code>monitor</code> or bucket parameters at all) there are only a fixed number of buckets. In that case, you can disable expiration entirely by setting the <code>timeout</code> to zero.</p>

Example

```
Init fn="init-request-limits" timeout="120"
```

init-uhome

(UNIX only) The `init-uhome` function loads information about the system’s user home directories into internal hash tables. This function slightly increases memory usage, but improves performance for servers that have a lot of traffic to home directories.

This function is applicable in `Init-class` directives.

Parameters

The following table describes the `init-uhome` parameter.

TABLE 5-5 `init-uhome` Parameter

Parameter	Description
<code>pwfile</code>	(Optional) Specifies the full file system path to a file other than <code>/etc/passwd</code> . If you do not specify this parameter, the default UNIX path (<code>/etc/passwd</code>) is used.

Example

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

load-modules

The `load-modules` function loads a shared library or dynamic-link library (DLL) into the server. Specified functions from the library can then be executed from any subsequent directives. Use this function to load new plug-ins or SAFs.

This function is applicable in `Init-class` directives.

If you define your own SAFs, load them by using the `load-modules` function and specify the shared library or DLL to load.

Parameters

The following table describes the `load-modules` parameters.

TABLE 5-6 `load-modules` Parameters

Parameter	Description
<code>shlib</code>	Specifies either the full path to the shared library or DLL, the name of a file that can be found in the operating system's library path, the name of a file that can be found in the server's <code>plugins</code> directory, or a file name relative to the server's <code>config</code> directory.
<code>funcs</code>	A comma-separated list of the names of the functions in the shared library or DLL to be made available for use by other <code>Init</code> directives or by <code>Service</code> directives in <code>obj.conf</code> . The list should not contain any spaces. The dash (-) character may be used in place of the underscore (_) character in function names.

TABLE 5-6 load-modules Parameters (Continued)

Parameter	Description
NativeThread	(Optional) Specifies the threading model to use: <ul style="list-style-type: none"> ■ no causes the routines in the library to use user-level threading. ■ yes enables kernel-level threading. The default is yes.
pool	The name of a custom thread pool as specified in thread-pool-init. For more information, see “thread-pool-init” on page 115 .

Examples

```
Init fn="load-modules" shlib="C:/mysrvfns/corpfns.dll" funcs="moveit"
Init fn="load-modules" shlib="/mysrvfns/corpfns.so" funcs="myinit,myservice"
Init fn="myinit"
```

pool-init

The pool-init function changes the default values of pooled memory settings. You can change the size of the free block list, or disable pooled memory entirely.

This function is applicable in Init-class directives.

Parameters

The following table describes the pool-init function parameters.

TABLE 5-7 pool-init Parameters

Parameter	Description
disable	(Optional) The flag to disable the internal pooled memory allocator. Disabling the internal pooled memory allocator is useful when debugging plug-ins. The default value is false.
block-size	(Optional) The size (in bytes) of the memory blocks allocated by the internal pooled memory allocator. The default value is 32768.

Example

```
Init fn="pool-init" disable="true"
```

register-http-method

The register-http-method function enables you to extend the HTTP protocol by registering new HTTP methods. This function is applicable in Init-class directives.

While accepting a connection, the server checks if the method it received is known to it. If the server does not recognize the method, it returns a 501 Method Not Implemented error message.

Parameters

The following table describes the `register-http-method` parameters.

TABLE 5-8 `register-http-method` Parameters

Parameter	Description
<code>methods</code>	A comma-separated list of the methods you are registering

Example

The following example shows the use of `register-http-method`:

```
Init fn="register-http-method" methods="MY_METHOD1,MY_METHOD2"
```

The methods can be called from a `Service` function in `obj.conf`, for example:

```
Service fn="MyHandler" method="MY_METHOD1"
```

thread-pool-init

The `thread-pool-init` function creates a new pool of user threads. A pool must be declared before it is used. For a plug-in to use the new pool, specify the `pool` parameter when loading the plug-in with the `Init-class` function `load-modules`. For more information, see [“load-modules” on page 113](#).

This function is applicable in `Init-class` directives.

One reason to create a custom thread pool would be if a plug-in is not thread-aware, in which case you can set the maximum number of threads in the pool to 1. The older parameter `NativeThread=yes` always engages one default native pool, called `NativePool`.

The native pool on UNIX is normally not engaged, as all threads are kernel-level threads. In addition, native thread pool parameters can be added to the `magnus.conf` file for convenience. For more information, see [Chapter 4, “Syntax and Use of `magnus.conf`”](#).

Parameters

The following table describes the `thread-pool-init` parameters.

TABLE 5-9 thread-pool-init Parameters

Parameter	Description
name	The name of the thread pool.
maxthreads	The maximum number of threads in the pool.
minthreads	The minimum number of threads in the pool.
queueSize	The size of the pool queue. If all threads in the pool are busy, further request-handling threads that need to get a thread from the pool wait in the pool queue. The number of request-handling threads that can wait in the queue is limited by the queue size. If the queue is full, the next request-handling thread that comes to the queue is turned away, with the result that the request is turned down. But the request-handling thread remains free to handle another request instead of becoming locked up in the queue.
stackSize	Stack size of each thread in the native (kernel) thread pool.

Example

```
Init fn="thread-pool-init" name="my-custom-pool"
    minthreads="1" maxthreads="5" queuesize="200"
Init fn="load-modules" shlib="myplugin.dll" funcs="tracker"
    pool="my-custom-pool"
```

Common SAFs

You can call some SAFs from `Init` in `magnus.conf` as well as from `ObjectType` directives in `obj.conf`. These SAFs are documented in [Chapter 7, “Predefined SAFs and Filters in obj.conf,”](#) as referenced below:

- “block-auth-cert” on page 171
- “block-cache-info” on page 172
- “block-cipher” on page 172
- “block-ip” on page 173
- “block-issuer-dn” on page 173
- “block-jroute” on page 174
- “block-keysize” on page 174
- “block-proxy-agent” on page 175
- “block-proxy-auth” on page 176
- “block-secret-keysize” on page 176
- “block-ssl-id” on page 177
- “block-user-dn” on page 177
- “block-via” on page 178

- “forward-auth-cert” on page 179
- “forward-cache-info” on page 180
- “forward-cipher” on page 180
- “forward-ip” on page 181
- “forward-issuer-dn” on page 181
- “forward-jroute” on page 182
- “forward-keysize” on page 183
- “forward-proxy-agent” on page 183
- “forward-proxy-auth” on page 184
- “forward-secret-keysize” on page 184
- “forward-ssl-id” on page 185
- “forward-user-dn” on page 185
- “forward-via” on page 186
- “http-client-config” on page 186
- “ssl-client-config” on page 191

Deprecated Init SAFs

The following `magnus.conf` Init SAFs are deprecated for Sun Java System Web Server 7.0.

TABLE 5-10 List of Deprecated Init SAFs

Directive	Description
<code>dns-cache-init</code>	Superseded by the <code>server.xml</code> <code>dns-cache</code> element. For more information, see “ dns-cache ” on page 57.
<code>flex-init</code>	Superseded by the <code>server.xml</code> <code>access-log</code> element. For more information, see “ access-log ” on page 43.
<code>flex-rotate-init</code>	Superseded by the <code>server.xml</code> <code>event</code> and <code>log</code> elements. For more information, see “ event ” on page 58 and “ log ” on page 71.
<code>init-cgi</code>	Superseded by the <code>server.xml</code> <code>cgi</code> element. For more information, see “ cgi ” on page 49.
<code>init-clf</code>	Superseded by the <code>server.xml</code> <code>access-log</code> element. For more information, see “ access-log ” on page 43.
<code>nt-console-init</code>	Superseded by the <code>server.xml</code> <code>log</code> element. For more information, see “ log ” on page 71.
<code>perf-init</code>	Superseded by the <code>server.xml</code> <code>stats</code> element. For more information, see “ stats ” on page 94.
<code>stats-init</code>	Superseded by the <code>server.xml</code> <code>stats</code> element. For more information, see “ stats ” on page 94.

Syntax and Use of obj.conf

The `obj.conf` file contains directives for HTTP request processing. The `obj.conf` file is located in the `instance_dir/config` directory.

During Web Server installation an `obj.conf` file is created. If you configure multiple virtual servers using the Admin Console or CLI, separate `obj.conf` files may be created for each virtual server. These files are named `virtual-server-name-obj.conf`, where `virtual-server-name` is the name of the virtual server. The newly created `virtual-server-name-obj.conf` becomes functional. This can be verified from the `server.xml`, where the name of the functional `obj.conf` is updated.

When changes made through Admin Console or CLI does not impact the `obj.conf`, for example changing log-level, no new virtual server specific `obj.conf` files are created and the default `obj.conf` will be used for all the virtual servers of the configuration.

However, when changes made through Admin interface impacts the `obj.conf`, for example Java disabling, a new `obj.conf` file is created for each virtual server. The `server.xml` file is immediately updated to reflect the appropriate `obj.conf` file used for each virtual server. During Java disabling if there are two virtual servers, `vs1` and `vs2`, two new virtual server specific `obj.conf` files are created, `vs1-obj.conf` and `vs2-obj.conf`. These new files are updated in the `server.xml` with the exact `object-file` used by these two virtual servers.

```
<object-file>vs1-obj.conf</object-file>
```

```
<object-file>vs2-obj.conf</object-file>
```

From this point onwards, the `obj.conf` file is neither updated, used, nor deleted. However, if the user wants to modify the `obj.conf` file for either of the two virtual servers, they should edit the respective file only and not the original `obj.conf` file.

Note – When another new virtual server `vs3` is added, a new `vs3-obj.conf` should be created, updated in the `server.xml` and become functional. However, this fails to happen and the existing `obj.conf` is shared by the `vs3`. This behavior is a known issue and is recorded as a bug CR 6754145.

When this document refers to `obj.conf`, it refers either to all `obj.conf` files or to the `obj.conf` file for the virtual server being discussed.

This chapter discusses the `obj.conf` directives; the use of `Object`, `Client`, `If`, `ElseIf`, and `Else` tags; the flow of control in `obj.conf`; and the syntax rules for editing `obj.conf`.

This chapter has the following sections:

- “Request-Handling Process Overview” on page 120
- “Directives in `obj.conf`” on page 121
- “Objects in `obj.conf`” on page 122
- “Flow of Control in `obj.conf`” on page 127
- “Changes in Function Flow” on page 134
- “Editing `obj.conf`” on page 135

Request-Handling Process Overview

When Web Server first starts up, it performs some initialization tasks and then waits for an HTTP request from a client (such as a browser). When the server receives a request, it first selects a virtual server. The `obj.conf` file of the selected virtual server determines how the server handles a request.

The `obj.conf` file contains a series of instructions known as directives that tell the server what to do at each stage in the request-handling process. These directives are grouped inside `Object` tags. Each directive invokes a function with one or more arguments.

Each directive applies to a specific stage in the request-handling process. For example, a directive that applies during the authorization stage in the request-handling process is an `AuthTrans` directive.

Stages in the Request-Handling Process

1. `AuthTrans` (authorization translation)
 - Verify the authorization information (such as name and password) sent in the request.
2. `NameTrans` (name translation)
 - Translate the logical URI into a local file system path.

3. PathCheck (path checking)
Check the local file system path for validity and check if the requestor has access privileges to the requested resource on the file system.
4. ObjectType (object typing)
Determine the Multipurpose Internet Mail Encoding (MIME) type of the requested resource (for example, text/html, image/gif, and so on), and establish other resource-specific settings.
5. Input (prepare to read input)
Select filters that will process incoming request data read by the Service step.
6. Output (prepare to send output)
Select filters that will process outgoing response data generated by the Service step.
7. Route (request routing)
Select the server to service the request.
8. Service (generate the response)
Generate and return the response to the client.
9. AddLog (adding log entries)
Add entries to log files.
10. Error (error handling)
Send an error message to the client and exit processing. This step is executed only if an error occurs in the previous steps.

Directives in obj.conf

The directives in `obj.conf` invoke functions known as Server Application Functions (SAFs). Each directive calls a function, indicating when to call it and specifying parameters for it.

The syntax of each directive is:

```
Directive fn="function" name1="value1" . . . nameN="valueN"
```

The value of the function (`fn`) parameter is the name of the SAF to execute. All directives must supply a value for the `fn` parameter; if there is no function, the instruction will do nothing. The remaining parameters are the arguments needed by the function, and they vary from function to function.

For example:

```
NameTrans fn="document-root" root="/Sun/webserver7/https-server/docs"
```

In this example, the directive is executed during the NameTrans stage of request processing, and invokes the document - root SAF to specify the document root directory for the server. The document - root SAF parameter root specifies the path to the document root directory.

Parameters can contain references to variables and expressions. The variables can be predefined variables, variables defined at request time using the set - variable SAF, or variables defined in server.xml. For more information on the set - variable SAF, see [“set-variable” on page 252](#). For more information on defining variables in server.xml, see [“variable” on page 96](#). For more information on expressions and variables, see [Appendix A, “Using Variables, Expressions, and String Interpolation.”](#)

The server is shipped with a set of built-in SAFs that you can use to create and modify directives in obj.conf. [Chapter 7, “Predefined SAFs and Filters in obj.conf,”](#) discusses these SAFs in detail. You can also define new SAFs, as discussed in [Chapter 2, “Creating Custom Server Application Functions,”](#) in *Sun Java System Web Server 7.0 Update 3 NSAPI Developer’s Guide*.

The magnus.conf file contains Init directive SAFs that initialize NASPI plug-ins. For more information, see [Chapter 5, “Predefined SAFs in magnus.conf.”](#)

Objects in obj.conf

Directives in the obj.conf file are grouped into Object tags. The default object contains instructions to the server on how to process requests by default. Each new object modifies the default object’s behavior.

An Object tag may contain a name or ppath attribute. Either parameter can be a wildcard pattern. For example:

```
<Object name="cgi">
```

```
<Object ppath="/usr/sun/webserver7/https-server/docs/private/*">
```

The server always starts handling a request by processing the directives in the default object. However, the server switches to processing directives in another object after the NameTrans stage of the default object if either of the following conditions is true:

- The successful NameTrans directive specifies a name argument.
- The physical path name that results from the NameTrans stage matches the ppath attribute of another object.

When the server is alerted to use an object other than the default object, it processes the directives in the other object before processing the directives in the default object. For some steps in the process, the server stops processing directives in that particular stage (such as the Service stage) as soon as one is successfully executed, whereas for other stages the server processes all directives in that stage, including the ones in the default object as well as those in the additional object. For more details, see [“Flow of Control in obj.conf” on page 127](#).

Objects That Use the name Attribute

If a `NameTrans` directive in the default object specifies a `name` argument, the server switches to processing the directives in the object of that name before processing the remaining directives in the default object.

For example, the following `NameTrans` directive in the default object assigns the name `cgi` to any request whose URL starts with `http://server_name/cgi`:

```
<Object name="default">
NameTrans fn="pfx2dir"
          from="/cgi"
          dir="D:/sun/webserver7/https-server/docs/mycgi"
          name="cgi"
...
</Object>
```

When the `NameTrans` directive is executed, the server starts processing directives in the object named `cgi`:

```
<Object name="cgi">
...
</Object>
```

Objects That Use the ppath Attribute

When the server completes processing the `NameTrans` directives in the default object, the logical URL of the request has been converted to a physical path name. If this physical path name matches the `ppath` attribute of another object in `obj.conf`, the server switches to processing the directives in that object before processing the remaining ones in the default object.

For example, the following `NameTrans` directive translates the `http://server_name/` part of the requested URL to `D:/sun/webserver7/https-server/docs/`, the document root directory:

```
<Object name="default">
NameTrans fn="document-root"
          root="D:/sun/webserver7/https-server/docs"
...
</Object>
```

In this example, the URL `http://server_name/internalplan1.html` is translated to `D:/sun/webserver7/https-server/docs/internalplan1.html`.

However, if `obj.conf` contains the following additional object:

```
<Object ppath="*internal*">
```

```
</Object>
```

In this example, the partial path `*internal*` matches the path `D:/sun/webserver7/https-server/docs/internalplan1.html`. The server starts processing the directives in this object before processing the remaining directives in the default object.

Using the Client, If, Elseif, and Else Tags

Additional tags are available to use within the `Object` tag. These tags give you greater flexibility when invoking directives within an object. This section contains the following sections:

- [“Client” on page 124](#)
- [“If, Elseif, and Else” on page 126](#)

Client

The `Client` tag enables you to limit the execution of a set of directives to requests received from specific clients. Directives listed within the `Client` tag are executed only when information in the client request matches the parameter values specified.

Client Tag Parameters

The following table lists the `Client` tag parameters.

TABLE 6-1 Client Tag Parameters

Parameter	Description
<code>browser</code>	The User-Agent string sent by a browser to the Web Server.
<code>chunked</code>	A Boolean value set by a client requesting chunked encoding.
<code>code</code>	The HTTP response code.
<code>dns</code>	The DNS name of the client.
<code>internal</code>	The Boolean value indicating internally generated request.
<code>ip</code>	The IP address of the client.
<code>keep-alive</code>	The Boolean value indicating whether the client has requested a keep-alive connection.
<code>keysize</code>	The key size used in an SSL transaction.
<code>match</code>	The match mode for the <code>Client</code> tag. The valid values are <code>all</code> , <code>any</code> , and <code>none</code> .

TABLE 6-1 Client Tag Parameters (Continued)

Parameter	Description
method	The HTTP method used by the browser.
name	The name of an object as specified in a previous NameTrans statement.
odds	A random value for evaluating the enclosed directive. The value can be a percentage or a ratio (for example, 20% or 1/5).
path	The physical path to the requested resource.
ppath	The physical path of the requested resource.
query	The query string sent in the request.
reason	The text version of the HTTP response code.
restarted	A Boolean value indicating that a request has been restarted.
secret-keysize	The secret key size used in an SSL transaction.
security	Indicates an encrypted request.
type	The type of document requested (such as text/html or image/gif).
uri	The URI section of the request from the browser.
urlhost	The DNS name of the virtual server requested by the client (the value is provided in the Host header of the client request).
variable-headers	Prevents access to a specific site, based on the request by the client. For example, <pre>Client variable-headers="Weferer:SKVFWVRKJVZCMHVIBGDA Service type="image/*" fn="deny-existence" </Client></pre>

The Client tag parameter provides greater control when the If directive is executed. In the following example, use of the odds parameter gives the request a 25% chance of being redirected:

```
<Client odds="25%">
NameTrans fn="redirect"
          from="/Pogues"
          url-prefix="http://pogues.example.com"
</Client>
```

One or more wildcard patterns can be used to specify the Client tag parameter values. Wildcards can also be used to exclude clients that match the parameter value specified in the

Client tag. In the following example, the Client tag and the AddLog directive are combined to direct the Web Server to log access requests from all clients except those from the specified subnet:

```
<Client ip="*~192.85.250.*">
AddLog fn="flex-log" name="access"
</Client>
```

You can also create a negative match by setting the match parameter of the Client tag to none. In the following example, access requests from the specified subnet are excluded as are all requests to the virtual server sun.com:

```
<Client match="none" ip="192.85.250.*" urlhost="www.sun.com">
AddLog fn="flex-log" name="access"
</Client>
```

For more information about wildcard patterns, see [Appendix B, “Using Wildcard Patterns.”](#)

If, ElseIf, and Else

The If, ElseIf, and Else tags enable you to define the conditions under which to execute a set of directives. Like the Client tag, these tags can only appear inside an Object tag. In addition, these tags can evaluate an expression, then conditionally execute one or more contained directives. However, there are some key differences between these tags and the Client tag, as summarized below:

- If and ElseIf tags offer a richer expression syntax, including support for regular expressions. This expression syntax is different from the Client syntax. For more information on the If and ElseIf expression syntax, see [“Expressions” on page 281](#).
- If, ElseIf, and Else tags can contain other tags.
- If and ElseIf expressions are evaluated once per request, not once per contained directive.
- If, ElseIf, and Else tags cannot contain multiple types of directives.
- Directives within the If and ElseIf tags can contain regular expression backreferences.

When used, an ElseIf or Else tag must immediately follow an If or ElseIf tag. ElseIf and Else tags are skipped if the preceding If or ElseIf expression evaluates to logical true.

The following example shows If, ElseIf, and Else tag syntax:

```
<If $path eq "/">
<If $browser =~ "MSIE">
NameTrans fn="rewrite" path="/msie.html"
</If>
<ElseIf $browser =~ "Mozilla">
NameTrans fn="rewrite" path="/mozilla.html"
</ElseIf>
```

```
<Else>  
NameTrans fn="rewrite" path="/unknown.html"  
</Else>  
</If>
```

This example presents a different page based on whether the browser is Microsoft Internet Explorer, Mozilla, or another browser.

Flow of Control in obj.conf

Before the server can process a request, it must direct the request to the correct virtual server. After the virtual server is determined, the server executes the `obj.conf` file of the specified virtual server. This section discusses how the server decides which directives to execute in `obj.conf`.

AuthTrans

When the server receives a request, it executes the `AuthTrans` directives in the default object to check if the client is authorized to access the server. If there is more than one `AuthTrans` directive, the server executes them in sequence until one succeeds in authorizing the user, unless one of them results in an error. If an error occurs, the server skips all other directives except for the `Error` directive.

`AuthTrans` directives work in conjunction with the `PathCheck` directives. The `AuthTrans` directive checks if the user name and password associated with the request are acceptable, but it does not allow or deny access to the request; that is done by the `PathCheck` directive.

The authorization process is split into two steps to incorporate multiple authorization schemes easily and provide the flexibility to have resources that record authorization information.

When a client initially makes a request, the user name and password are unknown. The `AuthTrans` directive gets the user name and password from the headers associated with the request. The `AuthTrans` and `PathCheck` directives work together to reject the request if they cannot validate the user name and password. When a request is rejected, the server displays a dialog box. The client includes the user name and password in the headers and resubmits the request.

NameTrans

The server executes a `NameTrans` directive in the default object to map the logical URL of the requested resource to a physical path name on the server's file system. For example, the URL `http://www.test.com/some/file.html` could be translated to the full file system path:

```
/usr/sun/webserver7/https-server/docs/some/file.html
```

The server looks at each `NameTrans` directive in the default object in turn, until it finds one that can be applied.

Because the server might not execute all `NameTrans` directives, the order in which the directives appear is important. For example:

```
NameTrans fn="document-root"  
          root="D:/sun/webserver7/https-server/docs"  
NameTrans fn="pfx2dir"  
          from="/cgi"  
          dir="D:/sun/webserver7/https-server/docs/mycgi"  
          name="cgi"
```

In this example, the directive that calls `pfx2dir` will never be executed because the previous directive always establishes the physical path name for the resource. For the `/cgi` prefix to work, the directive that calls `pfx2dir` must be moved before the directive that calls `document-root`.

If no directive sets the physical path name, the server translates the logical URL to a file system path relative to the document root. The document root is specified by the `document-root` element in `server.xml`. For more information on the `document-root` element, see [“virtual-server” on page 97](#).

How and When the Server Processes Other Objects

As a result of executing a `NameTrans` directive, the server might start processing directives in another object. This happens if the `NameTrans` directive that was successfully executed specifies a name or generates a partial path that matches the name or `ppath` attribute of another object.

If the successful `NameTrans` directive assigns a name by specifying a name argument, the server starts processing directives in the named object (defined with the `object` tag) before processing directives in the default object for the rest of the request-handling process.

For example, the following `NameTrans` directive in the default object assigns the name `cgi` to any request whose URL starts with `http://server_name/cgi/`.

```
<Object name="default">  
  ...  
  NameTrans fn="pfx2dir"  
            from="/cgi"  
            dir="D:/sun/webserver7/https-server/docs/mycgi" name="cgi"  
  ...  
</Object>
```

When the `NameTrans` directive is executed, the server starts processing directives in the object named `cgi`:


```
<Object name="cgi">
```

```
</Object>
```

When a NameTrans directive is successfully executed, there is a physical path name associated with the requested resource. If the resultant path name matches the ppath (partial path) attribute of another object, the server starts processing directives in the other object before processing directives in the default object for the rest of the request-handling process.

For example, assume obj.conf contains an object as follows:

```
<Object ppath ="*internal*">
```

```
</Object>
```

Consider that a successful NameTrans directive translates the requested URL to the path name `D:/sun/webserver7/https-server/docs/internalplan1.html`. In this case, the partial path `*internal*` matches the path

`D:/sun/webserver7/https-server/docs/internalplan1.html`. Hence, the server will process the directives in this object before processing the remaining directives in the default object.

PathCheck

After converting the logical URL of the requested resource to a physical path name in the NameTrans step, the server executes PathCheck directives to verify that the client is allowed to access the requested resource.

If there is more than one PathCheck directive, the server executes all directives in the order in which they appear, unless one of the directives denies access. If access is denied, the server switches to executing directives in the Error section.

If the NameTrans directive assigned a name or generated a physical path name that matches the name or ppath attribute of another object, the server first applies the PathCheck directives in the matching object before applying the directives in the default object.

ObjectType

Assuming that the PathCheck directives approve access, the server next executes the ObjectType directives to determine the MIME type of the request. The MIME type has three attributes: type, encoding, and language. When the server sends the response to the client, the type, language, and encoding values are transmitted in the headers of the response. The type also frequently helps the server to determine which Service directive to execute to generate the response to the client.

If there is more than one `ObjectType` directive, the server applies all directives in the order in which they appear. However, once a directive sets an attribute of the MIME type, further attempts to set the same attribute are ignored. The reason why all `ObjectType` directives are applied is that one directive may set one attribute, for example `type`, while another directive sets a different attribute, such as `language`.

As with the `PathCheck` directives, if another object has been matched to the request as a result of the `NameTrans` step, the server executes the `ObjectType` directives in the matching object before executing the `ObjectType` directives in the default object.

Setting the Type by File Extension

By default, the server determines the MIME type by calling the `type-by-extension` function. This function instructs the server to look up the MIME type according to the requested resource's file extension in the MIME types table. This table is created during virtual server initialization by the MIME types file (which is usually called `mime.types`). For more information, see [Chapter 8, "MIME Types."](#)

For example, the entry in the MIME types table for the extensions `.html` and `.htm` is usually:

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

which indicates that all files with the extension `.htm` or `.html` are text files formatted as HTML, and the type is `text/html`.

Note – If you make changes to the MIME types file, you must reconfigure the server for the changes to take effect.

Forcing the Type

If no `ObjectType` directive has set the type and the server does not find a matching file extension in the MIME types table, the type still has no value even after `type-by-expression` has been executed. Usually if the server does not recognize the file extension, it is a good idea to force the type to be `text/plain`, so that the content of the resource is treated as plain text. There are also other situations where you might want to set the type regardless of the file extension, such as forcing all resources in the designated CGI directory to have the MIME type `magnus-internal/cgi`.

The function that forces the type is `force-type`.

For example, the following directives first instruct the server to look in the MIME types table for the MIME type, then if the `type` attribute has not been set (that is, the file extension was not found in the MIME types table), set the `type` attribute to `text/plain`.

```
ObjectType fn="type-by-extension"  
ObjectType fn="force-type" type="text/plain"
```

If the server receives a request for a file `abc.date`, it looks in the MIME types table, does not find a mapping for the extension `.date`, and consequently does not set the `type` attribute. As the `type` attribute has not already been set, the second directive is successful in forcing the `type` attribute to `text/plain`.

The following example illustrates another use of `force-type`. In this example, the `type` is forced to `magnus-internal/cgi` before the server gets a chance to look in the MIME types table. In this case, all requests for resources in `http://server_name/cgi/` are translated into requests for resources in the directory `D:/sun/webServer7/https-server/docs/mycgi/`. As a name is assigned to the request, the server processes the `ObjectType` directives in the object named `cgi` before processing the ones in the default object. This object has one `ObjectType` directive, which forces the `type` to be `magnus-internal/cgi`.

```
NameTrans fn="pfx2dir"
          from="/cgi"
          dir="D:/sun/webserver7/https-server/docs/mycgi" name="cgi"
<Object name="cgi">
ObjectType fn="force-type" type="magnus-internal/cgi"
Service fn="send-cgi"
</Object>
```

The server continues processing all `ObjectType` directives including those in the default object, but as the `type` attribute has already been set, no other directive can set it to another value.

Input

The `Input` directive selects filters that will process incoming request data read by the `Service` step. `Input` directives are invoked when the server or plug-in first attempts to read entity body data from the client. You can add the NSAPI filters that process incoming data by invoking the `insert-filter` SAF in the `Input` stage of the request-handling process. NSAPI filters enable a function to intercept and potentially modify the content presented to or generated by another function. The `Input` directives are executed once per request.

The order of `Input fn="insert-filter"` and `Output fn="insert-filter"` directives in `obj.conf` is important if two or more filters are defined to occupy the same location in the filter stack. Filters that were inserted later will appear higher than filters that were inserted earlier.

Output

The `Output` directive selects filters that will process outgoing response data generated by the `Service` step. The `Output` directive allows you to invoke the `insert-filter` SAF to install NSAPI filters that process outgoing data. NSAPI filters enable a function to intercept and potentially modify the content presented to or generated by another function. `Output` directives

are executed when the server or a plug-in first attempts to write entity body data from the client. The Output directives are executed once per request.

The order of Input `fn="insert-filter"` and Output `fn="insert-filter"` directives in `obj.conf` is important if two or more filters are defined to occupy the same location in the filter stack. Filters that were inserted later will appear higher than filters that were inserted earlier.

Route

If a `Service` directive requires that the HTTP request be sent to another server, the server executes `Route` directives to determine how the request should be routed. Routing a request can involve selecting the server that will ultimately service the request and selecting a proxy through which the request is sent.

Service

The server executes a `Service` directive to generate the response to send to the client. The server looks at each `Service` directive to find the first one that matches the type, method, and query string. If a `Service` directive does not specify type, method, or query string, then the unspecified attribute matches anything.

If there is more than one `Service` directive, the server applies the first one that matches the conditions of the request and ignores all remaining `Service` directives.

For the `PathCheck` and `ObjectType` directives, if another object has been matched to the request as a result of the `NameTrans` step, the server considers the `Service` directives in the matching object before considering the ones in the default object. If the server successfully executes a `Service` directive in the matching object, it will not execute the `Service` directives in the default object, because it only executes one `Service` directive.

Service Examples

Consider an example where the server receives a request for the URL `D:/server_name/jos.html`. In this case, all directives executed by the server are in the default object.

1. The following `NameTrans` directive translates the requested URL to `D:/sun/webserver7/https-server/docs/jos.html`:

```
NameTrans fn="document-root"  
root="D:/sun/webserver7/https-server/docs"
```

2. Assume that the `PathCheck` directives succeed.
3. The following `ObjectType` directive tells the server to look up the resource's MIME type in the MIME types table:

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

4. The server finds the following entry in the MIME types table, which sets the type attribute to text/html:

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

5. The server invokes the following Service directive. The value of the type parameter matches anything that does *not* begin with magnus-internal/.

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

For a list of all wildcard patterns, see [Appendix B, “Using Wildcard Patterns.”](#)

Here is an example that involves using another object:

1. The following NameTrans directive assigns the name personnel to the request.

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

2. As a result of the name assignment, the server switches to processing the directives in the object named personnel. This object is defined as:

```
<Object name="personnel">
  Service fn="index-simple"
</Object>
```

3. The personnel object has no PathCheck or ObjectType directives, so the server processes the PathCheck and ObjectType directives in the default object. Assume that all PathCheck and ObjectType directives succeed.
4. When processing Service directives, the server starts by considering the Service directive in the personnel object, which is:

```
Service fn="index-simple"
```

5. The server executes this Service directive, which calls the index-simple function.

As a Service directive has now been executed, the server does not process any other Service directives. However, if the matching object did not have a Service directive that was executed, the server would continue looking at Service directives in the default object.

Default Service Directive

There is usually a Service directive that does the default task (sends a file) if no other Service directive matches a request sent by a browser. This default directive should come last in the list of Service directives in the default object to ensure that it only gets called if no other Service directives have succeeded. The default Service directive is usually:

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

This directive matches requests whose method is GET, HEAD, or POST, which covers nearly all requests sent by browsers. The value of the `type` argument uses special pattern-matching characters.

The characters `*~` mean anything that does not match the following characters, so the expression `*~magnus - internal/` means anything that does not match `magnus - internal/`. An asterisk by itself matches anything, so the whole expression `*~magnus - internal/*` matches anything that does not begin with `magnus - internal/`.

So if the server has not already executed a `Service` directive when it reaches this directive, it executes the directive as long as the request method is GET, HEAD, or POST, and the value of the `type` attribute does not begin with `magnus - internal/`. The invoked function is `send - file`, which simply sends the contents of the requested file to the client.

AddLog

After the server generates the response and sends it to the client, it executes `AddLog` directives to add entries to the log files. All `AddLog` directives are executed. The server can add entries to multiple log files.

Error

If an error occurs during the request-handling process, for example, if a `PathCheck` or `AuthTrans` directive denies access to the requested resource or the requested resource does not exist, the SAF 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.

Changes in Function Flow

There are times when the function flow changes from the normal request-handling process. This happens during internal redirects, restarts, and URI translation functions.

Restarted Requests

Requests may be restarted. For example, a `PathCheck` directive might restart a request for `http://server_name/` as a request for `http://server_name/index.html`.

Internal Requests

The server can generate internal requests. For example, an SHTML file or Servlet might include a file. While processing the original request, the server makes an internal request to retrieve this file.

URI Translation

The server can execute AuthTrans and NameTrans directives to translate a URI to a physical path name without starting a new request. For example, the server might execute AuthTrans and NameTrans directives in order to set the PATH_INFO_TRANSLATED CGI environment variable.

Editing obj.conf

Be very careful when editing this file. Simple mistakes can make the server fail to start or operate correctly.

Order of Directives

The order of directives is important, because the server executes them in the order in which they appear in obj.conf. The outcome of some directives affects the execution of other directives.

For PathCheck directives, the order within the PathCheck section is not so important because the server executes all PathCheck directives. However, the order within the ObjectType section is very important, because if an ObjectType directive sets an attribute value, no other ObjectType directive can change that value. For example, if the default ObjectType directives are listed in the following order (which is the incorrect way), every request will have its type value set to text/plain, and the server will not have a chance to set the type according to the extension of the requested resource.

```
ObjectType fn="force-type" type="text/plain"  
ObjectType fn="type-by-extension"
```

Similarly, the order of directives in the Service section is very important. The server executes the first Service directive that matches the current request and does not execute the others.

Parameters

The number and names of parameters depend on the function. The order of parameters on the line is not important.

Case Sensitivity

Items in the `obj.conf` file are case-sensitive including function names, parameter names, parameter values, and path names.

Separators

The C language allows function names to be composed only of letters, digits, and underscores. You may use the hyphen (-) character in the configuration file in place of underscore (_) for your C code function names. This is only true for function names.

Quotation Marks

Quotation marks (") are only required around the value strings when there is a space in the string. Otherwise, they are optional. Each open quotation mark must be matched by a closed quotation mark.

Spaces

- Spaces are not allowed at the beginning of a line except when continuing the previous line.
- Spaces are not allowed before or after the equal (=) sign that separates the name and value.
- Spaces are not allowed at the end of a line or on a blank line.

Line Continuation

A long line may be continued on the next line by beginning the next line with a space or tab.

Path Names

Always use forward slashes (/) rather than backslashes (\) in path names on the Windows platform. A backslash escapes the next character.

Comments

Comments begin with a pound (#) sign. If you manually add comments to `obj.conf`, then use the Admin Console or CLI to make changes to your server, your comments are overwritten when `obj.conf` is updated.

Predefined SAFs and Filters in obj.conf

This chapter describes the predefined Server Application Functions (SAFs) and filters that are used in the `obj.conf` file. For details about the syntax and use of the `obj.conf` file, see [Chapter 6, “Syntax and Use of obj.conf”](#)

Each SAF has its own parameters which are passed to it by an `obj.conf` directive. SAFs may examine, modify, or create server variables. Each SAF returns a result code that indicates whether it succeeded, did nothing, or failed.

The SAFs in this chapter are grouped by the type of directive that calls them. For an alphabetical list of predefined SAFs and server configuration elements, see [Appendix G, “Alphabetical List of Server Configuration Elements and Predefined SAFs.”](#)

This chapter contains the following sections:

- “The bucket Parameter” on page 138
- “AuthTrans” on page 138
- “NameTrans” on page 142
- “PathCheck” on page 153
- “ObjectType” on page 170
- “Input” on page 194
- “Output” on page 196
- “Route” on page 200
- “Service” on page 203
- “AddLog” on page 239
- “Error” on page 240
- “Common SAFs” on page 242

The bucket Parameter

The bucket parameter is common to all SAFs. 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"`. The bucket statistics are displayed by the `perfdump` utility, which can be set up through the Admin Console, CLI, or through the `service-dump` SAF. For more information, see [“service-dump” on page 231](#).

The following performance buckets are predefined:

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

For more information on performance buckets, see [“Using Performance Buckets” in *Sun Java System Web Server 7.0 Update 3 Performance Tuning, Sizing, and Scaling Guide*](#).

AuthTrans

The `AuthTrans` directive instructs the server to check for authorization before allowing a client to access resources. For more information, see [“AuthTrans” on page 127](#).

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

- [“basic-auth” on page 138](#)
- [“basic-ncsa” on page 140](#)
- [“get-sslid” on page 141](#)
- [“qos-handler” on page 141](#)

In addition, the following common SAFs are valid for the `AuthTrans` directive:

- [“match-browser” on page 244](#)
- [“set-variable” on page 252](#)

basic-auth

The `basic-auth` function verifies the authorization information sent by the client. The `Authorization` header is sent as part of the basic server authorization scheme. This function is usually used with the `PathCheck`-class function `require-auth`.

Parameters

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

TABLE 7-1 basic-auth Parameters

Parameter	Description
auth-type	Specifies the type of authorization to be used. The values can be <code>basic</code> , <code>digest</code> , or <code>ssl</code> . The default value is <code>basic</code> .
userdb	(Optional) Specifies the full path and file name of the database to be used for user verification. This parameter will be passed to the user function.
userfn	<p>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 (<code>user</code>), password (<code>pw</code>), user database (<code>userdb</code>), and group database (<code>groupdb</code>), if supplied, in the <code>pb</code> parameter.</p> <p>This function checks the name and password using the database and returns <code>REQ_NOACTION</code> if they are not valid. It returns <code>REQ_PROCEED</code> if the name and password are valid. The <code>basic-auth</code> function will then add <code>auth-type</code>, <code>auth-user</code> (<code>user</code>), <code>auth-db</code> (<code>userdb</code>), and <code>auth-password</code> (<code>pw</code>, Windows only) to the <code>rq->vars</code> <code>pb</code> block. For more information on custom functions, see Chapter 2, “Creating Custom Server Application Functions,” in <i>Sun Java System Web Server 7.0 Update 3 NSAPI Developer’s Guide</i>.</p>
groupdb	(Optional) Specifies the full path and file name of the user database. This parameter will be passed to the group function.
groupfn	<p>(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 (<code>user</code>), password (<code>pw</code>), user database (<code>userdb</code>), and group database (<code>groupdb</code>) in the <code>pb</code> parameter.</p> <p>This parameter also has access to the <code>auth-type</code>, <code>auth-user</code> (<code>user</code>), <code>auth-db</code> (<code>userdb</code>), and <code>auth-password</code> (<code>pw</code>, Windows only) parameters in the <code>rq->vars</code> <code>pb</code> block. The group function determines the group to which the user belongs using the group database, add it to <code>rq->vars</code> as <code>auth-group</code>, and return <code>REQ_PROCEED</code> if found. It returns <code>REQ_NOACTION</code> if the user’s group is not found.</p>
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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_auth"  
PathCheck fn="require-auth" auth-type="basic" realm="Marketing Plans"
```

See Also

[“require-auth” on page 166](#)

basic-nrsa

The `basic-nrsa` 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 with the PathCheck-class function `require-auth`.

Parameters

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

TABLE 7-2 `basic-nrsa` Parameters

Parameter	Description
<code>auth-type</code>	Specifies the type of authorization to be used. The values can be <code>basic</code> , <code>digest</code> , or <code>ssl</code> . The default value is <code>basic</code> .
<code>dbm</code>	(Optional) Specifies the full path and base file name of the user database in the native format of the server. The native format is a system DBM file, which is a hashed file format allowing instantaneous access to billions of users. If you use this parameter, do not use the <code>userfile</code> parameter.
<code>userfile</code>	(Optional) Specifies the full path name of the user database in the NCSA-style HTTPD user file format. This format consists of lines using the format <code>name:password</code> , where <code>password</code> is encrypted. If you use this parameter, do not use <code>dbm</code> .
<code>grpfile</code>	(Optional) Specifies the NCSA-style HTTPD group file to be used. Each line of a group file consists of <code>group:user1 user2 ... userN</code> where each user name is separated by spaces.
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
AuthTrans fn="basic-nrsa" auth-type="basic" dbm="/sun/server7/userdb/rs"  
PathCheck fn="require-auth" auth-type="basic" realm="Marketing Plans"  
AuthTrans fn="basic-nrsa" auth-type="basic" userfile="/sun/server7/.htpasswd"  
  grpfile="/sun/server7/.grpfile"  
PathCheck fn="require-auth" auth-type="basic" realm="Marketing Plans"
```

See Also

[“require-auth” on page 166](#)

get-sslid

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.

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

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 parameter for the `get-sslid` function.

TABLE 7-3 `get-sslid` Parameter

Parameter	Description
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

qos-handler

The `qos-handler` function examines the current quality of service (QOS) statistics for a virtual server, logs the statistics, and enforces the QOS parameters by returning an error. This function must be the first AuthTrans function configured in the `default` object.

Parameters

The following table describes parameter for the `qos-handler` function.

TABLE 7-4 qos-handler Parameter

Parameter	Description
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

See Also

[“qos-error” on page 241](#)

NameTrans

The NameTrans directive translates virtual URLs to physical directories on your server. The NameTrans directive must appear in the default object. For more information, see [“NameTrans” on page 127](#).

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

- [“assign-name” on page 143](#)
- [“document-root” on page 144](#)
- [“home-page” on page 145](#)
- [“map” on page 146](#)
- [“ntrans-dav” on page 147](#)
- [“ntrans-j2ee” on page 148](#)
- [“pfx2dir” on page 148](#)
- [“reverse-map” on page 150](#)
- [“rewrite” on page 151](#)
- [“strip-params” on page 152](#)
- [“unix-home” on page 152](#)

In addition, the following common SAFs are also valid for the NameTrans directive:

- [“match-browser” on page 244](#)
- [“redirect” on page 246](#)
- [“restart” on page 249](#)
- [“set-variable” on page 252](#)

assign-name

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, if you have the following directive in the default object:

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

Assume that 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 7-5 `assign-name` Parameters

Parameter	Description
<code>from</code>	(Optional) Wildcard pattern that specifies the path to be affected. If you do not specify the <code>from</code> parameter, all paths are affected.
<code>name</code>	Specifies an additional named object in <code>obj.conf</code> whose directives will be applied to this request.
<code>find-pathinfo-forward</code>	(Optional) Instructs the server to look for the <code>PATHINFO</code> forward in the path right after the <code>ntans-base</code> , instead of backward from the end of path as the server function <code>assign-name</code> does by default. The <code>find-pathinfo-forward</code> parameter is ignored if the <code>ntans-base</code> parameter is not set in <code>rq->vars</code> . By default, <code>ntans-base</code> is set. This feature can improve performance for certain URLs by reducing the number of statistics performed.

TABLE 7-5 assign-name Parameters (Continued)

Parameter	Description
nostat	<p>(Optional) Prevents the server from performing a stat on a specified URL.</p> <p>The effect of <code>nostat="virtual-path"</code> in the NameTrans function <code>assign-name</code> is that the server assumes that a stat on the specified <i>virtual-path</i> will fail. Therefore, use <code>nostat</code> only when the path of the <i>virtual-path</i> does not exist on the system. For example, use <code>nostat</code> for NSAPI plug-in URLs to improve performance by avoiding unnecessary stats on those URLs.</p> <p>When the default PathCheck server functions are used, the server does not stat for the paths <code>/ntrans-base/virtual-path</code> and <code>/ntrans-base/virtual-path/*</code> if <code>ntrans-base</code> is set (the default condition). It does not stat for the URLs <code>/virtual-path</code> and <code>/virtual-path/*</code> if <code>ntrans-base</code> is not set.</p>
bucket	<p>(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138.</p>

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

The `document-root` function specifies the root document directory for the server. If the physical path is not 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/webserver7/https-server/docs`, 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/webserver7/https-server/docs/a/b/file.html`.

You can also specify a document root in the `virtual-server` element of `server.xml`. For more information, see [“virtual-server” on page 97](#).

This function always returns REQ_PROCEED.

NameTrans directives listed after this directive will never be called. Ensure that the directive that invokes document-root is the last NameTrans directive.

There can be only one root document directory. To specify additional document directories, use the pfx2dir function.

Parameters

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

TABLE 7-6 document-root Parameters

Parameter	Description
root	File system path to the server's root document directory.
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
NameTrans fn="document-root" root="/usr/sun/webserver7/https-server/docs"
```

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

See Also

[“pfx2dir” on page 148](#)

home-page

The home-page function specifies the home page for your server.

Parameters

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

TABLE 7-7 home-page Parameters

Parameter	Description
path	<p>Path and name of the home page file. If path starts with a slash (/), it is assumed to be the full path to a file.</p> <p>If path is a relative path, this function sets the server's path variable and returns REQ_PROCEED.</p> <p>If path is a relative path, it is appended to the URI, and the function returns REQ_NOACTION. It then continues on to the other NameTrans directives.</p>
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

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

map

The map function maps a request URI to a URL on another server, allowing you to specify that a request should be serviced by another server. To load balance a given URI across multiple servers, use the map function in conjunction with the set-origin-server function. The map function looks for a certain prefix in the URI that the client is requesting. If map finds the prefix, it replaces the prefix with the mirror site prefix.

Parameters

The following table describes parameters for the map function.

TABLE 7-8 map Parameters

Parameter	Description
from	The URI prefix to map. The prefix should not contain trailing slashes.
to	The URL prefix to which the request should be mapped. The prefix should not contain trailing slashes.
name	(Optional) Specifies an additional named object in obj.conf. The directives of the named object will be applied to this request.

TABLE 7-8 map Parameters (Continued)

Parameter	Description
rewrite-host	(Optional) Indicates whether the Host HTTP request header is rewritten to match the host specified by the to parameter. In a reverse proxy configuration where the proxy server and origin server service the same set of virtual servers, you can specify <code>rewrite-host="false"</code> . The default is <code>true</code> , indicating that the Host HTTP request header is rewritten.
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
# Map everything under /docs to http://docs.sun.com/app/docs
NameTrans fn="map" from="/docs" to="http://docs.sun.com/app/docs"
```

See Also

[“set-origin-server” on page 201](#)

ntrans-dav

The `ntrans-dav` function determines whether a request should be handled by the WebDAV subsystem. If the request should be handled by the WebDAV subsystem, the function adds a `dav` object to the pipeline.

Parameters

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

TABLE 7-9 ntrans-dav Parameters

Parameter	Description
name	Specifies an additional named object in <code>obj.conf</code> whose directives will be applied to this request.
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

See Also

- “pcheck-dav” on page 166
- “service-dav” on page 229

ntrans-j2ee

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 7-10 ntrans-j2ee Parameters

Parameter	Description
name	Named object in <code>obj.conf</code> whose directives are applied to requests made to Java web applications.
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

See Also

- “error-j2ee” on page 240
- “find-index-j2ee” on page 160
- “service-j2ee” on page 232
- “type-j2ee” on page 194

pfx2dir

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 [“assign-name” on page 143](#) for details on using named objects.

Parameters

The following table describes parameters for the `pfx2dir` function.

TABLE 7-11 pfx2dir Parameters

Parameter	Description
from	URI prefix to convert. It should not have a trailing slash (/).
dir	Local file system directory path to which the prefix is converted. It should not have a trailing slash (/).
name	(Optional) Specifies an additional named object in <code>obj.conf</code> whose directives will be applied to this request.
find-pathinfo-forward	<p>(Optional) Instructs the to server look for the <code>PATHINFO</code> forward in the path after <code>ntans-base</code>, instead of backward from the end of path as the server function <code>find-pathinfo</code> does by default.</p> <p>The <code>find-pathinfo-forward</code> parameter is ignored if the <code>ntans-base</code> parameter is not set in <code>rq->vars</code> when the server function <code>find-pathinfo</code> is called. By default, <code>ntans-base</code> is set.</p> <p>This feature can improve performance for certain URLs by reducing the number of stats performed in the server function <code>find-pathinfo</code>.</p> <p>On Windows, you can use this feature to exclude the <code>PATHINFO</code> from the server URL normalization process (by changing <code>'\'</code> to <code>'/'</code>) when the <code>PathCheck</code> server function <code>find-pathinfo</code> is used. Some double-byte characters have hexadecimal values that might be parsed as URL separator characters such as <code>'\'</code> or <code>'~'</code>. Using the <code>find-pathinfo-forward</code> parameter can sometimes prevent incorrect parsing of URLs containing double-byte characters.</p>
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

```
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"
```

See Also

- [“assign-name” on page 143](#)
- [“rewrite” on page 151](#)

reverse-map

The `reverse-map` function rewrites the HTTP response headers when the server is functioning as a reverse proxy. `reverse-map` looks for the URL prefix specified by the `from` parameter in certain response headers. If the `from` prefix matches the beginning of the response header value, `reverse-map` replaces the matching portion with the `to` prefix.

Parameters

The following table describes parameters for the `reverse-map` function.

TABLE 7-12 reverse-map Parameters

Parameter	Description
<code>from</code>	URL prefix to be rewritten.
<code>to</code>	URL prefix that will be substituted in place of the <code>from</code> prefix.
<code>rewrite-location</code>	(Optional) Indicates whether the <code>location</code> HTTP response header should be rewritten. The default is <code>true</code> , indicating that the <code>location</code> header is rewritten.
<code>rewrite-content-location</code>	(Optional) Indicates whether the <code>Content-Location</code> HTTP response header should be rewritten. The default is <code>true</code> , indicating that the <code>Content-Location</code> header is rewritten.
<code>rewrite-headername</code>	(Optional) Indicates whether the <code>headername</code> HTTP response header should be rewritten, where <code>headername</code> is a user-defined header name. With the exception of the <code>Location</code> and <code>Content-Location</code> headers, the default is <code>false</code> , indicating that the <code>headername</code> header is not rewritten.
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
NameTrans fn="reverse-map" from="http://docs.sun.com/app/docs" to="/docs"
```

See Also

[“map” on page 146](#)

rewrite

The `rewrite` function allows flexible mappings between URIs and file system paths.

Parameters

The following table describes parameters for the `rewrite` function.

TABLE 7-13 `rewrite` Parameters

Parameter	Description
<code>from</code>	(Optional) Wildcard pattern that specifies the path of requests that should be rewritten. The default is to match all paths.
<code>root</code>	(Optional) File system path to the effective root document directory.
<code>name</code>	(Optional) Name of an object in <code>obj.conf</code> whose directives will be applied to this request.
<code>path</code>	(Optional) Rewritten partial path. If non-empty, the path must begin with a slash (/).
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

The following `obj.conf` code maps requests for the URI `/~user/index.html` to the file system path `/home/user/public_html/index.html`:

```
<If $path =~ "^/~([^/]+)(|/.*)$">
NameTrans fn="rewrite"
          root="/home/$1/public_html"
          path="$2"
</If>
```

See Also

[“restart” on page 249](#)

strip-params

The `strip-params` function removes the 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 7-14 `strip-params` Parameters

Parameter	Description
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
NameTrans fn="strip-params"
```

unix-home

(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 required 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 7-15 unix-home Parameters

Parameter	Description
subdir	Subdirectory within the user's home directory that contains the web documents of users.
pwfile	(Optional) Full path and file name of the password file if it is different from /etc/passwd.
name	(Optional) Specifies an additional named object whose directives will be applied to this request.
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
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 161](#)

PathCheck

The PathCheck directive checks the local file system path that is returned after the NameTrans step to verify that the client is allowed to access the specified resource. For more information, see [“PathCheck” on page 129](#).

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

- [“check-acl” on page 154](#)
- [“deny-existence” on page 157](#)
- [“find-compressed” on page 158](#)
- [“find-index” on page 159](#)
- [“find-index-j2ee” on page 160](#)
- [“find-links” on page 161](#)
- [“find-pathinfo” on page 162](#)
- [“get-client-cert” on page 163](#)
- [“nt-uri-clean” on page 164](#)
- [“ntcgicheck” on page 165](#)
- [“pcheck-dav” on page 166](#)
- [“require-auth” on page 166](#)

- “set-virtual-index” on page 167
- “ssl-check” on page 168
- “ssl-logout” on page 169
- “unix-uri-clean” on page 169

In addition, the following common SAFs are valid for the PathCheck directive:

- “match-browser” on page 244
- “restart” on page 249
- “set-variable” on page 252

check-acl

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 ACL 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 perform user authentication if required by the specified ACL, and also update the access control state. Because the server caches the ACLs returned by the check-acl function, do not use check-acl inside a Client, If, ElseIf, or Else container.

Parameters

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

TABLE 7-16 check-acl Parameters

Parameter	Description
acl	Name of an access control list.
path	(Optional) Wildcard pattern that specifies the path for which the ACL should be applied.
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

check-request-limits

The `check-request-limits` function monitors incoming requests matching a given attribute (for example, client IP address) and computes an average requests per second on a configurable time interval. When requests that match the monitored attribute exceed a threshold that you configure, subsequent matching requests are not serviced until the request rate drops. Use this function to detect possible denial-of-service attacks.

You must specify either `max-rps` or `max-connections`, otherwise `check-request-limits` does nothing. If you do not enter an attribute or attributes to monitor, the function monitors all requests.

By default, the function keeps entries on requests for 300 seconds (five minutes) before purging them. To adjust this time, use the `init-request-limits` SAF in `magnus.conf`. For more information, see [“init-request-limits” on page 112](#).

Parameters

The following table describes parameters for the `check-request-limits` function.

TABLE 7-17 `check-request-limits` Parameters

Parameter	Description
<code>max-rps</code>	(Optional) Threshold for matching requests per second. If this threshold is exceeded subsequent connections matching the criteria are not serviced. Because an acceptable threshold value can vary widely between sites, there is no default value for this parameter.
<code>max-connections</code>	(Optional) Maximum number of concurrent matching connections. If the server receives a request that matches the criteria while the number of matching requests currently being processed meets or exceeds this number, the request is denied. Note that this number is the current requests at any time, and is independent of the <code>interval</code> parameter. As soon as the number of concurrent requests falls below this limit, new matching requests are processed. Because an acceptable value can vary widely between sites, there is no default value for this parameter.
<code>interval</code>	(Optional) In seconds, the time interval during which average requests per second is computed. The <code>max-rps</code> limit is not applied until the next request rate computation. Because potential attackers can have unlimited requests serviced during this interval, balance the length of this interval against the performance cost of recomputing the maximum requests per second. The default is 30 seconds.

TABLE 7-17 check-request-limits Parameters (Continued)

Parameter	Description
continue	<p>(Optional) Determines what condition must be met in order for a blocked request type to become available again for servicing. Valid values are:</p> <ul style="list-style-type: none"> ■ <code>silence</code> – Refused requests must fall to zero in a subsequent interval for service to resume. ■ <code>threshold</code> – Refused requests must fall below the <code>max-rps</code> value for service to resume. <p>The default value is <code>threshold</code>.</p>
error	(Optional) The HTTP status code to use for blocked requests. The default value is 503 (the <code>Service Unavailable</code> error).
monitor	<p>(Optional) A request attribute to monitor. Request rates are tracked in a bucket named by the value of this parameter. If the <code>monitor</code> parameter is not specified, the matching requests are tracked in an unnamed (anonymous) bucket. Note that these buckets are different from the buckets you specify with the standard <code>obj.conf</code> bucket parameter.</p> <p>Although the value of the <code>monitor</code> parameter can be a fixed string, it is most useful when you use predefined variables, for example, <code>monitor="\$ip"</code>. You can also specify multiple variables, separated by a colon. For example, <code>monitor="\$ip:\$uri"</code>. For a list of predefined variables, see “Predefined Variables” on page 277.</p>
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

The following example limits a client IP to a maximum request rate of 10 requests per second in the default interval of 30 seconds:

```
PathCheck fn="check-request-limit" monitor="$ip" max-rps="10"
```

The following example limits a client IP to a maximum request rate of 10 requests per second when accessing any Perl CGIs. Other types of requests are unlimited:

```
<If path = "*.pl">
PathCheck fn="check-request-limits" monitor="$ip" max-rps="10"
</If>
```

For more information on using the `If` tag, see [“If, ElseIf, and Else” on page 126](#).

The following example limits requests globally for Perl CGIs to 10 requests per second. No specific monitor parameter is specified:

```
<If path = "*.pl">
PathCheck fn="check-request-limits" max-rps="10"
</If>
```

The following example limits a client IP from generating more than 10 Perl CGI requests per second, or 5 JSP requests per second. To track the Perl and JSP totals separately, the specified monitor parameters contain both a fixed string identifier and the client IP variable:

```
<If path = "*.pl">
PathCheck fn="check-request-limits" max-rps="10" monitor="perl:$ip"
</If>
<If path = "*.jsp">
PathCheck fn="check-request-limits" max-rps="5" monitor="jsp:$ip"
</If>
```

The following example limits any one client IP to no more than 5 connections at a given time:

```
PathCheck fn="check-request-limits" max-connections="2" monitor="$ip"
```

deny-existence

The deny-existence function sends a 404 Not Found message when a client tries to access a specified path.

Parameters

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

TABLE 7-18 deny-existence Parameters

Parameter	Description
path	(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.
bong-file	(Optional) Specifies a file to send rather than responding with the 404 Not Found message. The value is a full file system path.
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
PathCheck fn="deny-existence" path="/usr/sun/server7/docs/private"
```

```
PathCheck fn="deny-existence" bong-file="/svr/msg/go-away.html"
```

find-compressed

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 as recent as the non-compressed 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 non-compressed versions of a file. The version of the file that is selected is based on the individual client's capabilities.

A compressed version of a file must have the same file name as the non-compressed version but with a `.gz` suffix. For example, the compressed version of a file named `/httpd/docs/index.html` would be named `/httpd/docs/index.html.gz`. To compress files, you can use the freely available `gzip` program.

Because compressed files are sent as is to the client, you should not compress files such as SHTML pages, CGI programs, or pages created with JavaServer Pages™ (JSP™) technology that need to be interpreted by the server. To compress the dynamic content generated by these types of files, use the `http-compression` filter.

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

Parameters

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

TABLE 7-19 find-compressed Parameters

Parameter	Description
check-age	<p>(Optional) Specifies whether to check if the compressed version is older than the non-compressed version. The values can be yes or no.</p> <ul style="list-style-type: none"> ■ If set to yes, the compressed version will not be selected if it is older than the non-compressed version. ■ If set to no, the compressed version is always selected, even if it is older than the non-compressed version. <p>By default, the value is set to yes.</p>
vary	<p>(Optional) Specifies whether to insert a Vary: Accept-Encoding header. The values can be yes or no.</p> <ul style="list-style-type: none"> ■ If set to yes, a Vary: Accept-Encoding header is always inserted when a compressed version of a file is selected. ■ If set to no, a Vary: Accept-Encoding header is never inserted. <p>By default, the value is set to yes.</p>
bucket	<p>(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138.</p>

Example

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

See Also

[“http-compression” on page 198](#)

find-index

The `find-index` function investigates whether the requested path is a directory. If yes, the function searches for an index file in the directory, and then changes the path to point to the index file. If an index file is not found, the server generates a directory listing. If the `obj.conf` file has a `NameTrans` directive that calls `home-page` and the requested directory is the root directory, the server returns the home page to the client instead of the index page.

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 7-20 `find-index` Parameters

Parameter	Description
<code>index-names</code>	Comma-separated list of index file names to look for. Use spaces only if they are part of a file name. Do not include spaces before or after the commas. This list is case-sensitive if the file system is case-sensitive.
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

See Also

- [“find-index-j2ee” on page 160](#)
- [“home-page” on page 145](#)
- [“index-common” on page 211](#)
- [“index-simple” on page 214](#)

find-index-j2ee

The `find-index-j2ee` function implements `welcome-file-list` processing for requests that map to directories in a Java web application. When configuring the server to host Servlet or JSP-technology-based web applications, position the `find-index-j2ee` SAF above the `find-index` SAF in `obj.conf`. This position ensures that `web.xml` `welcome-file-list` ordering takes precedence over the default index file order configured for the `find-index` SAF.

Parameters

The following table describes parameter for the `find-index-j2ee` function.

TABLE 7-21 find-index-j2ee Parameter

Parameter	Description
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
PathCheck fn="find-index-j2ee"
```

See Also

- [“find-index” on page 159](#)
- [“ntrans-j2ee” on page 148](#)
- [“service-j2ee” on page 232](#)
- [“error-j2ee” on page 240](#)
- [“type-j2ee” on page 194](#)

find-links

(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 7-22 find-links Parameters

Parameter	Description
disable	Character string of links to disable: <ul style="list-style-type: none"> ▪ <code>h</code> indicates hard link ▪ <code>s</code> indicates soft link ▪ <code>o</code> allows symbolic links only if the target of the link is owned by the user that the server runs as

TABLE 7-22 find-links Parameters (Continued)

Parameter	Description
dir	(Optional) 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. If you do not specify a dir, all directories are checked.
checkFileExistence	(Optional) Checks linked file for existence and aborts the request with the 403 Forbidden error if the check fails. Controls whether the server checks if the target of the link exists. If set to Y, the server aborts the request with a 403 Forbidden error if the target of a link does not exist. The default is N, meaning the server does not check whether the target exists.
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

```
PathCheck fn="find-links" disable="so" dir="public_html"
```

See Also

[“unix-home” on page 152](#)

find-pathinfo

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 7-23 find-pathinfo Parameter

Parameter	Description
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

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

get-client-cert

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` and allows the request to proceed. Otherwise, it returns `REQ_ABORTED` and sets the protocol status to `403 forbidden`, causing the request to fail.

Parameters

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

TABLE 7-24 `get-client-cert` Parameters

Parameter	Description
<code>dorequest</code>	<p>(Optional) Controls whether to actually get the certificate, or just test for its presence.</p> <ul style="list-style-type: none"> ■ <code>1</code> 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 might already have the client certificate if it was requested on the initial handshake, or if a cached SSL session has been resumed. ■ <code>0</code> tells the function not to redo the SSL3 handshake if the server does not already have the client certificate. <p>If a certificate is obtained from the client and verified successfully by the server, the ASCII base 64 encoding of the DER-encoded X.509 certificate is placed in the parameter <code>auth-cert</code> in the <code>Request->vars</code> pblock, and the function returns <code>REQ_PROCEED</code>, allowing the request to proceed.</p> <p>The default value is <code>0</code>.</p>

TABLE 7-24 get-client-cert Parameters (Continued)

Parameter	Description
require	<p>(Optional) Controls whether failure to get a client certificate will abort the HTTP request.</p> <ul style="list-style-type: none"> ■ 1 tells the function to abort the HTTP request if the client certificate is not present after <code>dorequest</code> is handled. In this case, the HTTP status is set to <code>PROTOCOL_FORBIDDEN</code>, and the function returns <code>REQ_ABORTED</code>. ■ 0 tells the function to return <code>REQ_NOACTION</code> if the client certificate is not present after <code>dorequest</code> is handled. <p>The default value is 1.</p>
method	(Optional) Specifies a wildcard pattern for the HTTP methods for which the function will be applied. If <code>method</code> is absent, the function is applied to all requests.
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

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"
```

nt-uri-clean

(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 7-25 nt-uri-clean Parameters

Parameter	Description
tildeok	(Optional) 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.

TABLE 7-25 nt-uri-clean Parameters (Continued)

Parameter	Description
dotdirrok	(Optional) If present, /./ sequences are allowed.
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

See Also

[“unix-uri-clean” on page 169](#)

ntcgicheck

(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 7-26 ntcgicheck Parameters

Parameter	Description
extension	The replacement file extension.
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

See Also

- [“send-cgi” on page 222](#)
- [“send-wincgi” on page 228](#)
- [“send-shellcgi” on page 227](#)

pcheck-dav

The `pcheck-dav` function inserts a DAV-specific service function as the first service function, if the following are true:

- The `Translate:f` header is present
- DAV is enabled for the request URI
- 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 7-27 pcheck-dav Parameter

Parameter	Description
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

See Also

- [“ntrans-dav” on page 147](#)
- [“service-dav” on page 229](#)

require-auth

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 the `AuthTrans` directive.

If a user is authorized in the `AuthTrans` directive and the `auth-user` parameter is provided, the name of the user must match with 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 7-28 require-auth Parameters

Parameter	Description
path	(Optional) Wildcard local file system path on which this function should operate. If no path is provided, the function applies to all paths.
auth-type	Type of HTTP authorization used. Currently, <code>basic</code> is the only authorization type defined.
realm	String sent to the browser indicating the secure area (or realm) for which user name and password are requested.
auth-user	(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 given access.
auth-group	(Optional) Specifies a wildcard list of groups that are allowed access.
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

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 138](#)
- [“basic-ncsa” on page 140](#)

set-virtual-index

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, 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 is not found. `REQ_RESTART` is returned if the current URI matches any one of the URIs mentioned in the `from` parameter, or if the `from` parameter is not specified.

Parameters

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

TABLE 7-29 set-virtual-index Parameters

Parameter	Description
virtual-index	URI of the content generator that acts as an index for the URI that the user enters.
from	(Optional) Comma-separated list of URIs for which this virtual-index is applicable. If from is not specified, the virtual-index always applies.
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

ssl-check

The `ssl-check` function is used along with a `Client` tag to limit access of certain directories to non-exportable browsers. If a restriction is selected that is not consistent with the current cipher settings, this function displays a warning that ciphers with larger secret key sizes must be enabled.

The function returns `REQ_NOACTION` if SSL is not enabled, or if the `secret-keysize` parameter is not specified. If the secret key size 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 key size 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 7-30 ssl-check Parameters

Parameter	Description
secret-keysize	(Optional) Minimum number of bits required in the secret key.

TABLE 7-30 `ssl-check` Parameters (Continued)

Parameter	Description
<code>bong-file</code>	(Optional) Name of a file (not a URI) to be served if the restriction is not met.
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

ssl-logout

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 that the client connects, a new SSL session is 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 7-31 `ssl-logout` Parameter

Parameter	Description
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

unix-uri-clean

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

Parameters

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

TABLE 7-32 `unix-uri-clean` Parameters

Parameter	Description
<code>dotdirrok</code>	If present, <code>./</code> sequences are allowed.

TABLE 7-32 `unix-uri-clean` Parameters (Continued)

Parameter	Description
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

See Also

[“nt-uri-clean” on page 164](#)

ObjectType

The `ObjectType` directives determine the MIME type of the file that has to be sent to the client in response to a request. For more information, see [“ObjectType” on page 129](#).

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

- [“block-auth-cert” on page 171](#)
- [“block-cache-info” on page 172](#)
- [“block-cipher” on page 172](#)
- [“block-ip” on page 173](#)
- [“block-issuer-dn” on page 173](#)
- [“block-jroute” on page 174](#)
- [“block-keysize” on page 174](#)
- [“block-proxy-agent” on page 175](#)
- [“block-proxy-auth” on page 176](#)
- [“block-secret-keysize” on page 176](#)
- [“block-ssl-id” on page 177](#)
- [“block-user-dn” on page 177](#)
- [“block-via” on page 178](#)
- [“force-type” on page 178](#)
- [“forward-auth-cert” on page 179](#)
- [“forward-cache-info” on page 180](#)
- [“forward-cipher” on page 180](#)
- [“forward-ip” on page 181](#)
- [“forward-issuer-dn” on page 181](#)
- [“forward-jroute” on page 182](#)
- [“forward-keysize” on page 183](#)
- [“forward-proxy-agent” on page 183](#)
- [“forward-proxy-auth” on page 184](#)

- “forward-secret-keysize” on page 184
- “forward-ssl-id” on page 185
- “forward-user-dn” on page 185
- “forward-via” on page 186
- “http-client-config” on page 186
- “set-basic-auth” on page 187
- “set-cache-control” on page 188
- “set-cookie” on page 189
- “set-default-type” on page 190
- “shtml-hacktype” on page 190
- “ssl-client-config” on page 191
- “type-by-exp” on page 192
- “type-by-extension” on page 193
- “type-j2ee” on page 194

In addition, the following common SAFs are valid for the `ObjectType` directive:

- “match-browser” on page 244
- “set-variable” on page 252

block-auth-cert

The `block-auth-cert` function instructs the server not to forward the client’s SSL/TLS certificate to remote servers.

Parameters

The following table describes parameter for the `block-auth-cert` function.

TABLE 7-33 `block-auth-cert` Parameter

Parameter	Description
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
ObjectType fn="block-auth-cert"
```

See Also

[“forward-auth-cert” on page 179](#)

block-cache-info

The `block-cache-info` function instructs the server not to forward information about local cache hits to remote servers.

Parameters

The following table describes parameter for the `block-cache-info` function.

TABLE 7-34 `block-cache-info` Parameter

Parameter	Description
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
ObjectType fn="block-cache-info"
```

See Also

[“forward-cache-info” on page 180](#)

block-cipher

The `block-cipher` function instructs the server to forward the name of the client’s SSL/TLS cipher suite to remote servers.

Parameters

The following table describes parameter for the `block-cipher` function.

TABLE 7-35 `block-cipher` Parameter

Parameter	Description
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
ObjectType fn="block-cipher"
```

See Also

[“forward-cipher” on page 180](#)

block-ip

The `block-ip` function instructs the server not to forward the client’s IP address to remote servers.

Parameters

The following table describes parameter for the `block-ip` function.

TABLE 7-36 `block-ip` Parameter

Parameter	Description
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
ObjectType fn="block-ip"
```

See Also

[“forward-ip” on page 181](#)

block-issuer-dn

The `block-issuer-dn` function instructs the server not to forward the distinguished name of the issuer of the client’s SSL/TLS certificate to remote servers.

Parameters

The following table describes parameter for the `block-issuer-dn` function.

TABLE 7-37 block-issuer-dn Parameter

Parameter	Description
bucket	(Optional) Common to all obj . conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
ObjectType fn="block-auth-cert"
```

See Also

[“forward-issuer-dn” on page 181](#)

block-jroute

The block-jroute function instructs the server not to forward information about request routing to remote servers using the proprietary Proxy-jroute format.

Parameters

The following table describes parameter for the block-jroute function.

TABLE 7-38 block-jroute Parameter

Parameter	Description
bucket	(Optional) Common to all obj . conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
ObjectType fn="block-jroute"
```

See Also

[“forward-jroute” on page 182](#)

block-keysize

The block-keysize function instructs the server not to forward the size of the client’s SSL/TLS key to remote servers.

Parameters

The following table describes parameter for the `block-keysize` function.

TABLE 7-39 `block-keysize` Parameter

Parameter	Description
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
ObjectType fn="block-keysize"
```

See Also

[“forward-keysize” on page 183](#)

block-proxy-agent

The `block-proxy-agent` function instructs the server not to forward its version information to remote servers.

Parameters

The following table describes parameter for the `block-proxy-agent` function.

TABLE 7-40 `block-proxy-agent` Parameter

Parameter	Description
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
ObjectType fn="block-proxy-agent"
```

See Also

[“forward-proxy-agent” on page 183](#)

block-proxy-auth

The `block-proxy-auth` function instructs the server not to forward the client's proxy authentication credentials, that is, the client's Proxy-authorization HTTP request header, to remote servers.

Parameter

The following table describes parameter for the `block-proxy-auth` function.

TABLE 7-41 `block-proxy-auth` Parameter

Parameter	Description
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
ObjectType fn="block-proxy-auth"
```

See Also

[“forward-proxy-auth” on page 184](#)

block-secret-keysize

The `block-secret-keysize` function instructs the server not to forward the size of the client's SSL/TLS secret key to remote servers.

Parameters

The following table describes parameter for the `block-secret-keysize` function.

TABLE 7-42 `block-secret-keysize` Parameter

Parameter	Description
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
ObjectType fn="block-secret-keysize"
```


See Also

[“forward-secret-keysize” on page 184](#)

block-ssl-id

The `block-ssl-id` function instructs the server not to forward the client’s SSL/TLS session ID to remote servers.

Parameters

The following table describes parameter for the `block-ssl-id` function.

TABLE 7-43 `block-ssl-id` Parameter

Parameter	Description
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
ObjectType fn="block-ssl-id"
```

See Also

[“forward-ssl-id” on page 185](#)

block-user-dn

The `block-user-dn` function instructs the server not to forward the distinguished name of the subject of the client’s SSL/TLS certificate to remote servers.

Parameters

The following table describes parameter for the `block-user-dn` function.

TABLE 7-44 block-user-dn Parameter

Parameter	Description
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
ObjectType fn="block-user-dn"
```

See Also

[“forward-user-dn” on page 185](#)

block-via

The `block-via` function instructs the server not to forward information about request routing to remote servers using the HTTP/1.1 Via format.

Parameters

The following table describes parameter for the `block-via` function.

TABLE 7-45 block-via Parameter

Parameter	Description
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
ObjectType fn="block-via"
```

See Also

[“forward-via” on page 186](#)

force-type

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

Ensure 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. 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 settings are ignored.

Parameters

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

TABLE 7-46 `force-type` Parameters

Parameter	Description
<code>type</code>	(Optional) Type assigned to a matching request (the <code>Content-Type</code> header).
<code>enc</code>	(Optional) Encoding assigned to a matching request (the <code>Content-Encoding</code> header).
<code>lang</code>	(Optional) Language assigned to a matching request (the <code>Content-Language</code> header).
<code>charset</code>	(Optional) Character set for the <code>magnus-charset</code> parameter in <code>rq->srvhdrs</code> . If a browser sends the <code>Accept-Charset</code> header or its <code>User-Agent</code> is Mozilla/1.1 or newer, then append “; charset= <i>charset</i> ” to <code>Content-Type</code> , where <i>charset</i> is the value of the <code>magnus-charset</code> parameter in <code>rq->srvhdrs</code> .
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
ObjectType fn="force-type" type="text/plain"
```

```
ObjectType fn="force-type" lang="en_US"
```

See Also

- [“type-by-exp” on page 192](#)
- [“type-by-extension” on page 193](#)

forward-auth-cert

The `forward-auth-cert` function instructs the server to forward the client’s SSL/TLS certificate to remote servers.

Parameters

The following table describes parameters for the `forward-auth-cert` function.

TABLE 7-47 `forward-auth-cert` Parameters

Parameter	Description
<code>hdr</code>	(Optional) Name of the HTTP request header used to communicate the client's DER-encoded SSL/TLS certificate in Base 64 encoding. The default value is <code>Proxy-auth-cert</code> .
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

See Also

[“block-auth-cert” on page 171](#)

forward-cache-info

The `forward-cache-info` function instructs the server to forward information about local cache hits to remote servers.

Parameter

The following table describes parameters for the `forward-cache-info` function.

TABLE 7-48 `forward-cache-info` Parameters

Parameter	Description
<code>hdr</code>	(Optional) Name of the HTTP request header used to communicate information about local cache hits. The default value is <code>Cache-info</code> .
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

See Also

[“block-cache-info” on page 172](#)

forward-cipher

The `forward-cipher` function instructs the server to forward the name of the client's SSL/TLS cipher suite to remote servers.

Parameters

The following table describes parameters for the `forward-cipher` function.

TABLE 7-49 `forward-cipher` Parameters

Parameter	Description
<code>hdr</code>	(Optional) Name of the HTTP request header used to communicate the name of the client's SSL/TLS cipher suite. The default value is <code>Proxy-cipher</code> .
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

See Also

[“block-cipher” on page 172](#)

forward-ip

The `forward-ip` function instructs the server to forward the client's IP address to remote servers.

Parameters

The following table describes parameters for the `forward-ip` function.

TABLE 7-50 `forward-ip` Parameters

Parameter	Description
<code>hdr</code>	(Optional) Name of the HTTP request header used to communicate the client's IP address. The default value is <code>Client-ip</code> .
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

See Also

[“block-ip” on page 173](#)

forward-issuer-dn

The `forward-issuer-dn` function instructs the server to forward the distinguished name of the issuer of the client's SSL/TLS certificate to remote servers.

Parameters

The following table describes parameters for the `forward-issuer-dn` function.

TABLE 7-51 `forward-issuer-dn` Parameters

Parameter	Description
<code>hdr</code>	(Optional) Name of the HTTP request header used to communicate the distinguished name of the issuer of the client's SSL/TLS certificate. The default value is <code>Proxy-issuer-dn</code> .
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

See Also

[“block-issuer-dn” on page 173](#)

forward-jroute

The `forward-jroute` function instructs the server to forward information about request routing using the proprietary `Proxy-jroute` format. The `Proxy-jroute` header field is used by the `set-origin-server` function and some Servlet containers to implement session stickiness.

Parameters

The following table describes parameters for the `forward-jroute` function.

TABLE 7-52 `forward-jroute` Parameters

Parameter	Description
<code>hdr</code>	(Optional) Name of the HTTP request header used to communicate the request routing information. The default value is <code>Proxy-jroute</code> .
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

See Also

- [“block-jroute” on page 174](#)
- [“set-origin-server” on page 201](#)

forward-keysize

The `forward-keysize` function instructs the server to forward the size of the client's SSL/TLS key to remote servers.

Parameters

The following table describes parameters for the `forward-keysize` function.

TABLE 7-53 `forward-keysize` Parameters

Parameter	Description
<code>hdr</code>	(Optional) Name of the HTTP request header used to communicate the size of the client's SSL/TLS key. The default value is <code>Proxy-keysize</code> .
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

See Also

[“block-keysize” on page 174](#)

forward-proxy-agent

The `forward-proxy-agent` function instructs the server to forward its version information to remote servers.

Parameters

The following table describes parameters for the `forward-proxy-agent` function.

TABLE 7-54 `forward-proxy-agent` Parameters

Parameter	Description
<code>hdr</code>	(Optional) Name of the HTTP request header used to communicate server version. The default value is <code>Proxy-agent</code> .
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

See Also

- [“block-proxy-agent” on page 175](#)
- [“http-client-config” on page 186](#)

forward-proxy-auth

The `forward-proxy-auth` instructs the server to forward the client's proxy authentication credentials, that is, the client's Proxy-authorization HTTP request header to remote servers.

Parameters

The following table describes parameter for the `forward-proxy-auth` function.

TABLE 7-55 forward-proxy-auth Parameter

Parameter	Description
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
ObjectType fn="forward-proxy-auth"
```

See Also

[“block-proxy-auth” on page 176](#)

forward-secret-keysize

The `forward-secret-keysize` function instructs the server to forward the size of the client's SSL/TLS secret key to remote servers.

Parameters

The following table describes parameters for the `forward-secret-keysize` function.

TABLE 7-56 forward-secret-keysize Parameters

Parameter	Description
hdr	(Optional) Name of the HTTP request header used to communicate the size of the client's SSL/TLS secret key. The default value is <code>Proxy-secret-keysize</code> .
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

See Also

[“block-secret-keysize” on page 176](#)

forward-ssl-id

The `forward-ssl-id` function instructs the server to forward the client's SSL/TLS session ID to remote servers.

Parameter

The following table describes parameters for the `forward-ssl-id` function.

TABLE 7-57 `forward-ssl-id` Parameters

Parameter	Description
<code>hdr</code>	(Optional) Name of the HTTP request header used to communicate the client's SSL/TLS session ID. The default value is <code>Proxy-ssl-id</code> .
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

See Also

[“block-ssl-id” on page 177](#)

forward-user-dn

The `forward-user-dn` function instructs the server to forward the distinguished name of the subject of the client's SSL/TLS certificate to remote servers.

Parameters

The following table describes parameters for the `forward-user-dn` function.

TABLE 7-58 `forward-user-dn` Parameters

Parameter	Description
<code>hdr</code>	(Optional) Name of the HTTP request header used to communicate the distinguished name of the subject of the client's SSL/TLS certificate. The default value is <code>Proxy-user-dn</code> .
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

See Also

[“block-user-dn” on page 177](#)

forward-via

The `forward-via` function instructs the server to forward information about request routing to remote servers using the HTTP/1.1 `Via` format. The HTTP/1.1 `Via` header field records the proxy servers and protocol versions that were involved in routing a request.

Parameters

The following table describes parameters for the `forward-via` function.

TABLE 7-59 `forward-via` Parameters

Parameter	Description
<code>hdr</code>	(Optional) Name of the HTTP request header used to communicate routing information. The default value is <code>Via</code> .
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

See Also

[“block-via” on page 178](#)

http-client-config

The `http-client-config` function configures the server’s HTTP client.

Parameters

The following table describes parameters for the `http-client-config` function.

TABLE 7-60 `http-client-config` Parameters

Parameter	Description
<code>keep-alive</code>	(Optional) Indicates whether the HTTP client should attempt to use persistent connections. The default value is <code>true</code> .
<code>keep-alive-timeout</code>	(Optional) The maximum number of seconds to keep a persistent connection open. The default is value is 29.

TABLE 7-60 http-client-config Parameters (Continued)

Parameter	Description
always-use-keep-alive	(Optional) Indicates whether the HTTP client can reuse existing persistent connections for all types of requests. The default value is <code>false</code> indicating that persistent connections will not be reused for non-GET requests or for requests with a body.
protocol	(Optional) HTTP protocol version string. By default, the HTTP client uses either HTTP/1.0 or HTTP/1.1 based on the contents of the HTTP request. In general, you should not use the protocol parameter unless you encounter specific protocol interoperability problems.
proxy-agent	(Optional) Value of the proxy-agent HTTP request header. The default is a string that contains the web server product name and version.
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
ObjectType fn="http-client-config" keep-alive="false"
```

set-basic-auth

The `set-basic-auth` function sets the HTTP basic authentication credentials used by the server when it sends an HTTP request. Use `set-basic-auth` to authenticate to a remote origin server or proxy server.

Parameters

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

TABLE 7-61 set-basic-auth Parameters

Parameter	Description
user	Name of the user to authenticate.
password	Password of the user to authenticate.
hdr	(Optional) Name of the HTTP request header used to communicate the credentials.
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
ObjectType fn="set-basic-auth"
    user="admin"
    password="secret"
    hdr="proxy-authorization"
```

See Also

- “[ssl-client-config](#)” on page 191
- “[block-auth-cert](#)” on page 171
- “[forward-auth-cert](#)” on page 179
- “[block-proxy-auth](#)” on page 176
- “[forward-proxy-auth](#)” on page 184

set-cache-control

The `set-cache-control` function allows you to specify the HTTP caching policy for the response being sent back to the client.

Parameters

The following table describes parameters for the `set-cache-control` function.

TABLE 7-62 `set-cache-control` Parameters

Parameter	Description
<code>control</code>	HTTP cache control directives. Separate multiple directives by commas.
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “ The bucket Parameter ” on page 138.

The following table describes some of the useful cache control directives defined by the HTTP/1.1 protocol.

TABLE 7-63 Cache Control Directives

Directive	Description
<code>public</code>	The response may be cached by any cache.
<code>private</code>	The response must not be cached by a shared cache (for example, a proxy server).

TABLE 7-63 Cache Control Directives (Continued)

Directive	Description
no-cache	Clients must ask the server for updated content on each access.
max-age= <i>n</i>	The response should not be cached for more than <i>n</i> seconds.

Example

```
ObjectType fn="set-cache-control" control="private,max-age=60"
```

set-cookie

The set-cookie function allows you to set a cookie in the response being sent back to the client.

Parameters

The following table describes parameters for the set-cookie function.

TABLE 7-64 set-cookie Parameters

Parameter	Description
name	Name of the cookie.
value	(Optional) Value of the cookie. The default value is null.
path	(Optional) Base URI to which the cookie applies. The default value is / (slash).
domain	(Optional) The domain name of servers to which the cookie must be sent. If no domain is specified, web browsers send the cookie only to the server that sets the cookie.
max-age	(Optional) Maximum time (in seconds) after which the cookie expires. If max-age is not specified, web browsers delete the cookie when the user closes the web browser.
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
<If not defined $cookie{‘FIRSTVISITTIME’}>
ObjectType fn="set-cookie"
  name="FIRSTVISITTIME"
  value="$time"
  max-age="31536000"
</If>
```

set-default-type

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 are not set for a response, then just before the headers are sent the defaults defined by `set-default-type` are used. 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 7-65 set-default-type Parameters

Parameter	Description
<code>enc</code>	(Optional) Encoding assigned to a matching request (the Content-Encoding header).
<code>lang</code>	(Optional) Language assigned to a matching request (the Content-Language header).
<code>charset</code>	(Optional) Character set for the <code>magnus-charset</code> parameter in <code>rq->srvhdrs</code> . If a browser sends the Accept-Charset header or its User-Agent is Mozilla/1.1 or newer, then append “; charset= <i>charset</i> ” to Content-Type, where <i>charset</i> is the value of the <code>magnus-charset</code> parameter in <code>rq->srvhdrs</code> .
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

shtml-hacktype

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 7-66 `shtml-hacktype` Parameters

Parameter	Description
<code>exec-hack</code>	(Optional, UNIX only) Instructs the function to change the Content-Type only if the execute bit is enabled. The value of the parameter is not important, but the parameter should be provided. The value can be <code>true</code> .
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

ssl-client-config

The `ssl-client-config` function configures options used when the server connects to a remote server using SSL/TLS.

Parameter

The following table describes parameters for the `ssl-client-config` function.

TABLE 7-67 `ssl-client-config` Parameters

Parameter	Description
<code>client-cert-nickname</code>	(Optional) Nickname of the client certificate to present to the remote server. The default is not to present a client certificate.
<code>validate-server-cert</code>	(Optional) Boolean that indicates whether the server validates the certificate presented by the remote server. The default value is <code>true</code> , indicating that remote servers must present valid certificates that were issued by a trusted certificate authority.
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
ObjectType fn="ssl-client-config" validate-server-cert="false"
```

See Also

- “set-basic-auth” on page 187
- “block-auth-cert” on page 171
- “forward-auth-cert” on page 179
- “block-proxy-auth” on page 176
- “forward-proxy-auth” on page 184

type-by-exp

The `type-by-exp` function matches the current path with a wildcard expression. If they match, the `type` parameter information is applied to the file. This is the same as `type-by-extension`, except that 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 7-68 `type-by-exp` Parameters

Parameter	Description
<code>exp</code>	Wildcard pattern of paths for which this function is applied.
<code>type</code>	(Optional) Type assigned to a matching request (the <code>Content-Type</code> header).
<code>enc</code>	(Optional) Encoding assigned to a matching request (the <code>Content-Encoding</code> header).
<code>lang</code>	(Optional) Language assigned to a matching request (the <code>Content-Language</code> header).
<code>charset</code>	(Optional) The character set for the <code>magnus-charset</code> parameter in <code>rq->srvhdrs</code> . If a browser sends the <code>Accept-Charset</code> header or its <code>User-Agent</code> is Mozilla/1.1 or newer, then append “; charset= <i>charset</i> ” to <code>Content-Type</code> , where <i>charset</i> is the value of the <code>magnus-charset</code> parameter in <code>rq->srvhdrs</code> .
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “ The bucket Parameter ” on page 138.

Example

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


See Also

- [“type-by-extension” on page 193](#)
- [“force-type” on page 178](#)

type-by-extension

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. The MIME type is added to the `Content-Type` header that is sent back to the client.

The table of MIME type mappings is created by a `mime-file` element in the `server.xml` file, which loads a MIME types file or list and creates the mappings.

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

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

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

Parameters

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

TABLE 7-69 `type-by-extension` Parameters

Parameter	Description
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

See Also

- [“type-by-exp” on page 192](#)
- [“mime-file” on page 74](#)
- [“force-type” on page 178](#)

type-j2ee

The `type-j2ee` function sets the Content-Type for requests that map to resources in a Java web application. When configuring the server to host Servlet or JSP-based web applications, `type-j2ee` must be the first `ObjectType` SAF in `obj.conf`. This is to ensure that `web.xml` MIME type mappings take precedence over the default MIME type mappings.

Parameters

The following table describes parameter for the `type-j2ee` function.

TABLE 7-70 `type-j2ee` Parameter

Parameter	Description
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
ObjectType fn="type-j2ee"
```

See Also

- [“ntrans-j2ee” on page 148](#)
- [“service-j2ee” on page 232](#)
- [“error-j2ee” on page 240](#)
- [“find-index-j2ee” on page 160](#)

Input

The Input directives allow you to select filters that will process incoming request data read by the `Service` stage. For more information, see [“Input” on page 131](#).

The following Input-class filter is described in detail in this section:

- [“sed-request” on page 195](#)

In addition, the following common SAFs are valid for the Input directive:

- [“insert-filter” on page 243](#)
- [“match-browser” on page 244](#)
- [“remove-filter” on page 248](#)
- [“set-variable” on page 252](#)

Every Input directive has the following optional parameters.

TABLE 7-71 Input Directive's Optional Parameters

Optional Parameters	Description
type	(Optional) Specifies a wildcard pattern of MIME types for which this function will be executed.
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.

sed-request

The sed - request filter applies the sed edit commands to an incoming request entity body, for example, an uploaded file or submitted form.

Parameters

The following table shows the sed - request parameters:

TABLE 7-72 sed - request Parameters

Parameter	Description
sed	Specifies a sed command script. When multiple sed parameters are provided, the sed edit commands are evaluated in the order they appear.

Example

The following obj . conf code instructs sed - request to encode any (<) and (>) characters posted in an HTML form:

```
Input fn="insert-filter"
      method="POST"
      filter="sed-request"
      sed="s/</\&lt;/g"
      sed="s/%3C/\&lt;/g"
      sed="s/%3C/\&lt;/g"
      sed="s/>/\&gt;/g"
      sed="s/%3E/\&gt;/g"
      sed="s/%3E/\&gt;/g"
```

Because POST bodies are usually URL-encoded, it is important to check for URL-encoded forms when editing POST bodies. %3C is the URL-encoded form of (<) and %3E is the URI-encoded form of (>).

See Also

- [“insert-filter” on page 243](#)
- [“sed-response” on page 200](#)

Output

The Output stage allows you to select filters that will process outgoing data. For more information, see [“Output” on page 131](#).

Every Output directive has the following optional parameters:

TABLE 7-73 Output Directive’s Optional Parameters

Optional Parameters	Description
type	(Optional) Specifies a wildcard pattern of MIME types for which this function will be executed.
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.

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

- [“compress-file” on page 196](#)
- [“http-compression” on page 198](#)
- [“sed-response” on page 200](#)

In addition, the following common SAFs are valid for the Output directive:

- [“insert-filter” on page 243](#)
- [“match-browser” on page 244](#)
- [“redirect” on page 246](#)
- [“remove-filter” on page 248](#)
- [“set-variable” on page 252](#)

compress-file

For dynamic compression of static files, `compress-file` SAF along with `find-compressed` SAF is used in Web Server 7.0 Update 3.

When a request is sent to the URI for the first time, the `compress-file` function creates a compressed file in the specified subdirectory, provided the file size is between `min-size` and `max-size`.

If `check-age` parameter is set to true, the `compress-file` function checks if the available compressed file version is as recent as the non-compressed file version. If not, it recreates a compressed file.

The `find-compressed` function checks if the compressed version of the requested file is available. It changes the path and directs to the compressed file under the following conditions:

- A compressed version is available.
- The compressed file version is as recent as the non-compressed version.
- The client supports compression.
- The HTTP method is GET or HEAD.

The following table describes parameter for the `compress-file` filter.

TABLE 7-74 `compress-file` Parameter

Parameter	Description
<code>subdir</code>	(Optional) A directory name, relative to the directory in which the original non-compressed file is located. To overwrite a pre-compressed, compressed file lying in <code>docroot</code> , set the <code>subdir</code> to period (<code>.</code>). The default value is period (<code>.</code>), that is, current directory.
<code>check-age</code>	(Optional) Specifies whether to check if the compressed version is older than the non-compressed version. The values can be true or false. The default value is true.
<code>vary</code>	(Optional) The values can be true or false. The default value is true.
<code>compression-level</code>	(Optional) Controls the compression level used by the compression library. The values are from 1 to 9. A value of 1 results in the best speed. A value of 9 results in the best compression. The default value is 6.
<code>min-size</code>	(Optional) The values are from 0 to <code>INT_MAX</code> . The default value is 256.
<code>max-size</code>	(Optional) The values are from <code>min-size</code> to <code>INT_MAX</code> . The default value is 1048576

Example

```
<Objectname="default">
NameTrans fn="assign-name" from="*.html" name="find-compressed"
...
Service method=(GET|HEAD|POST) type=*-magnus-internal/* fn=compress-file
subdir=".compressed-files"
Service method=(GET|HEAD|POST) type=*-magnus-internal/* fn=send-file
```

```

...
</Object>

<Objectname="find-compressed">
PathCheck fn="find-compressed"
</Object>

```

See Also

- [“find-compressed” on page 158](#)
- [“http-compression” on page 198](#)

http-compression

The `http-compression` filter compresses outgoing content. If the client does not support compression, or the outgoing content is already compressed, `http-compression` performs no action.

Unlike the `find-compressed` SAF, the `http-compression` filter can compress dynamic content such as the output from SHTML pages, CGI programs, or JSPs. However, for reasons of efficiency, the `find-compressed` SAF is better for static content such as non-parsed HTML files. For more information, see [“find-compressed” on page 158](#).

Parameters

The following table describes parameter for the `http-compression` filter.

TABLE 7-75 `http-compression` Parameter

Parameter	Description
<code>vary</code>	Controls whether the filter inserts a <code>Vary:</code> Accept-encoding header. If <code>vary</code> is absent, the default value is <code>yes</code> . <code>yes</code> tells the filter to insert a <code>Vary:</code> Accept-encoding header when it compresses content. <code>no</code> tells the filter to never insert a <code>Vary:</code> Accept-encoding header.
<code>fragment-size</code>	Size in bytes of the memory fragment used by the compression library to control how much to compress at a time. The default value is 8096.
<code>compression-level</code>	Controls the compression level used by the compression library. Valid values are from 1 to 9. A value of 1 results in the best speed. A value of 9 results in the best compression. The default value is 6.

TABLE 7-75 http-compression Parameter (Continued)

Parameter	Description
window-size	Controls an internal parameter of the compression library. Valid values are from 9 to 15. Higher values result in better compression at the expense of memory usage. The default value is 15.
memory-level	Controls how much memory is used by the compression library. Valid values are from 1 to 9. A value of 1 uses the minimum amount of memory but is slow. A value of 9 uses the maximum amount of memory for optimal speed. The default value is 8.

Example

```
Output fn="insert-filter"
      type="text/*"
      filter="http-compression"
      vary="on"
      compression-level="9"
```

In this example, `type="text/*"` restricts compression to documents that have a MIME type of `text/*` (for example, `text/ascii`, `text/css`, `text/html`, and so on).

Alternatively, you can specifically exclude browsers that do handle compressed content well by using the `Client` tag as follows:

```
<Client match="none" \
  browser="*MSIE [1-3]*" \
  browser="*MSIE [1-5]*Mac*" \
  browser="Mozilla/[1-4]*Nav*">
Output fn="insert-filter" filter="http-compression" type="text/*"
</Client>
```

This example restricts compression to browsers that are *not* any of the following:

- Internet Explorer for Windows earlier than version 4
- Internet Explorer for Macintosh earlier than version 6
- Netscape Navigator/Communicator earlier than version 6

Internet Explorer on Windows earlier than version 4 may request compressed data at times, but does not correctly support it. Internet Explorer on Macintosh earlier than version 6 does the same. Netscape Communicator version 4.x requests compression, but only correctly handles compressed HTML. It does not correctly handle linked CSS or JavaScript[™] from the compressed HTML, so administrators often simply prevent their servers from sending any compressed content to that browser (or earlier).

For more information about the `Client` tag, see “[Client](#)” on page 124.

sed-response

The sed-response filter applies sed edit commands to an outgoing response entity body, for example, an HTML file or output from a Servlet.

Parameter

The following table describes parameter for the sed-response filter

TABLE 7-76 sed-response Parameter

Parameter	Description
sed	Specifies a sed command script. When multiple sed parameters are provided, the sed edit commands are evaluated in the order they appear.

Example

The following `obj.conf` code instructs sed-response to rewrite any occurrence of `http://127.0.0.1/` in an HTML response to `http://server.example.com/`:

```
Output fn="insert-filter"
       type="text/html"
       filter="sed-response"
       sed="s|http://127.0.0.1/|http://server.example.com/|g"
```

See Also

- [“insert-filter” on page 243](#)
- [“sed-request” on page 195](#)

Route

The Route directive specifies information as to where the Web Server should route requests. For more information, see [“Route” on page 132](#).

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

- [“set-origin-server” on page 201](#)
- [“set-proxy-server” on page 202](#)

In addition, the following common SAFs are valid for the Route directive:

- [“match-browser” on page 244](#)
- [“set-variable” on page 252](#)

set-origin-server

The `set-origin-server` function distributes the load across a set of homogeneous HTTP origin servers.

Parameters

The following table describes parameters for the `set-origin-server` function.

TABLE 7-77 `set-origin-server` Parameters

Parameter	Description
<code>server</code>	URL of the origin server. If multiple <code>server</code> parameters are given, the server distributes the load among the specified origin servers.
<code>sticky-cookie</code>	(Optional) Name of a cookie that, when present in a response, will cause subsequent requests to stick to that origin server. The default value is <code>JSESSIONID</code> .
<code>sticky-param</code>	(Optional) Name of a URI parameter to inspect for route information. When the URI parameter is present in a request URI and its value contains a colon (<code>:</code>) followed by a route ID, the request will stick to the origin server identified by that route ID. The default value is <code>jsessionid</code> .
<code>route-hdr</code>	(Optional) Name of the HTTP request header used to communicate route IDs to origin servers. <code>set-origin-server</code> associates each origin server named by a <code>server</code> parameter with a unique route ID. Origin servers may encode this route ID in the URI parameter named by the <code>sticky-param</code> parameter to cause subsequent requests to stick to them. The default value is <code>Proxy-jroute</code> .
<code>route-cookie</code>	(Optional) Name of the cookie generated by the server when it encounters a <code>sticky-cookie</code> in a response. The <code>route-cookie</code> parameter stores a route ID that enables the server to direct subsequent requests back to the same origin server. The default value is <code>JROUTE</code> .
<code>rewrite-host</code>	(Optional) Indicates whether the host HTTP request header is rewritten to match the host specified by the <code>server</code> parameter. The default value is <code>false</code> indicating that the host header is not rewritten.
<code>rewrite-location</code>	(Optional) Indicates whether the <code>Location</code> HTTP response header that matches the <code>server</code> parameter should be rewritten. The default value is <code>true</code> , indicating that the matching <code>Location</code> headers are rewritten.
<code>rewrite-content-location</code>	(Optional) Indicates whether the <code>Content-Location</code> HTTP response header that matches the <code>server</code> parameter should be rewritten. The default value is <code>true</code> , indicating that the matching <code>Content-Location</code> headers are rewritten.

TABLE 7-77 set-origin-server Parameters (Continued)

Parameter	Description
<code>rewrite-headername</code>	(Optional) Indicates whether the <i>headername</i> HTTP response headers that match the server parameter should be rewritten, where <i>headername</i> is a user-defined header name. <i>headername</i> is in lowercase. With the exception of the Location and Content-Location headers, the default value is <code>false</code> , indicating that the <i>headername</i> header is not rewritten.
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
Route fn="set-origin-server"
      server="http://appserver1:8000"
      server="http://appserver2:8000"
```

See Also

- [“map” on page 146](#)
- [“set-proxy-server” on page 202](#)

set-proxy-server

The `set-proxy-server` function directs the server to retrieve the current resource from a particular proxy server.

Parameters

The following table describes parameters for the `set-proxy-server` function.

TABLE 7-78 set-proxy-server Parameters

Parameter	Description
<code>server</code>	URL of the remote proxy server. If multiple server parameters are given, the server distributes load among the specified remote servers.
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
Route fn="set-proxy-server"
      server="http://webcache1.eng.sun.com:8080"
      server="http://webcache2.eng.sun.com:8080"
```

See Also

[“set-origin-server” on page 201](#)

Service

The `Service` directives send the response data to the client. For more information, see [“Service” on page 132](#).

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.

TABLE 7-79 Service Directive's Optional Parameters

Optional Parameters	Description
<code>type</code>	Specifies a wildcard pattern of MIME types for which this function will be executed. The <code>magnus-internal/*</code> MIME types are used only to select a <code>Service</code> function to execute.
<code>method</code>	Specifies a wildcard pattern of HTTP methods for which this function will be executed. Common HTTP methods are GET, HEAD, and POST.
<code>query</code>	Specifies a wildcard pattern of query strings for which this function will be executed.
<code>UseOutputStreamSize</code>	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 – Set this parameter to zero (0) to disable output stream buffering.
<code>flushTimer</code>	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 <code>flushTimer</code> value for an application, further buffering is disabled. This is necessary for monitoring the status of CGI applications that run continuously and generate periodic status update reports. If this parameter is not specified, the default is 3000 milliseconds.
<code>ChunkedRequestBufferSize</code>	Determines the default buffer size, in bytes, for un-chunking request data. If this parameter is not specified, the default is 8192 bytes.
<code>ChunkedRequestTimeout</code>	Determines the default timeout, in seconds, for un-chunking request data. If this parameter is not specified, the default is 60 seconds.

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

The `UseOutputStreamSize`, `ChunkedRequestBufferSize`, and `ChunkedRequestTimeout` parameters also have equivalent `magnus.conf` directives. The `obj.conf` parameters override the `magnus.conf` directives.

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

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

This directive usually comes last in the set of `Service`-class directives to give all other `Service` directives a chance to be invoked. This directive is invoked if the method of the request is `GET`, `HEAD`, or `POST`, and the type does not start with `magnus-internal/`. Note here that the pattern `*~` means “does not match.” For a list of characters that can be used in patterns, see [Appendix B, “Using Wildcard Patterns.”](#)

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

- “`add-footer`” on page 205
- “`add-header`” on page 206
- “`append-trailer`” on page 208
- “`delete-file`” on page 209
- “`imagemap`” on page 210
- “`index-simple`” on page 214
- “`key-toosmall`” on page 215
- “`list-dir`” on page 216
- “`make-dir`” on page 218
- “`proxy-retrieve`” on page 219
- “`remove-dir`” on page 220
- “`rename-file`” on page 221
- “`send-cgi`” on page 222
- “`send-file`” on page 225
- “`send-range`” on page 226
- “`send-shellcgi`” on page 227
- “`send-wincgi`” on page 228
- “`service-dav`” on page 229
- “`service-dump`” on page 231
- “`service-j2ee`” on page 232
- “`service-trace`” on page 234
- “`shtml-send`” on page 235
- “`stats-xml`” on page 236
- “`upload-file`” on page 238

In addition, the following common SAFs are valid for the `Service` directive:

- “`match-browser`” on page 244
- “`remove-filter`” on page 248
- “`send-error`” on page 250

- “set-variable” on page 252

add-footer

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

Parameters

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

TABLE 7-80 `add-footer` Parameters

Parameter	Description
<code>file</code>	(Optional) Path name to the file containing the footer. Specify either <code>file</code> or <code>uri</code> . By default, the path name is relative. If the path name is absolute, set the <code>NSIntAbsFilePath</code> parameter to <code>yes</code> .
<code>uri</code>	(Optional) URI pointing to the resource containing the footer. The value can be <code>file</code> or <code>uri</code> .
<code>NSIntAbsFilePath</code>	(Optional) If the <code>file</code> parameter is specified, the <code>NSIntAbsFilePath</code> parameter determines whether the file name is absolute or relative. The default is relative. Set the value to <code>yes</code> to indicate an absolute file path.
<code>type</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “ Service ” on page 203.
<code>method</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “ Service ” on page 203.
<code>query</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “ Service ” on page 203.
<code>UseOutputStreamSize</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “ Service ” on page 203.
<code>flushTimer</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “ Service ” on page 203.

TABLE 7-80 add-footer Parameters (Continued)

Parameter	Description
ChunkedRequestBufferSize	(Optional) Common to all Service-class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203 .
ChunkedRequestTimeout	(Optional) Common to all Service-class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203 .
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
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/webserver7/https-server/footers/footer1.html"
    NSIntAbsFilePath="yes"
```

See Also

- [“append-trailer” on page 208](#)
- [“add-header” on page 206](#)

add-header

The add-header 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 and hence the header can be dynamically generated.

Parameters

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

TABLE 7-81 add-header Parameters

Parameter	Description
file	(Optional) Path name to the file containing the header. The value can be <code>file</code> or <code>uri</code> . By default, the path name is relative. If the path name is absolute, set the <code>NSIntAbsFilePath</code> parameter as <code>yes</code> .

TABLE 7-81 add-header Parameters (Continued)

Parameter	Description
uri	(Optional) URI pointing to the resource containing the header. The value can be file or uri.
NSIntAbsFilePath	(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.
type	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203.
method	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203.
query	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203.
UseOutputStreamSize	(Optional) Common to all Service-class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203.
flushTimer	(Optional) Common to all Service-class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203.
ChunkedRequestBufferSize	(Optional) Common to all Service-class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203.
ChunkedRequestTimeout	(Optional) Common to all Service-class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203.
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138.

Example

```
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/webserver7/https-server/headers/header1.html"
  NSIntAbsFilePath="yes"
```

See Also

- [“add-footer” on page 205](#)
- [“append-trailer” on page 208](#)

append-trailer

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

Returns

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 7-82 `append-trailer` Parameters

Parameter	Description
<code>trailer</code>	Text to append to HTML documents. The string is unescaped with <code>util_uri_unescape</code> 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 <code>:LASTMOD:</code> which is replaced by the date the file was last modified, you must also specify a time format with <code>timefmt</code> .
<code>timefmt</code>	(Optional) Time format string for <code>:LASTMOD:</code> . If <code>timefmt</code> is not provided, <code>:LASTMOD:</code> will not be replaced with the time.
<code>type</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203 .
<code>method</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203 .
<code>query</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203 .

TABLE 7-82 `append-trailer` Parameters (Continued)

Parameter	Description
<code>UseOutputStreamSize</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203 .
<code>flushTimer</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203 .
<code>ChunkedRequestBufferSize</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203 .
<code>ChunkedRequestTimeout</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203 .
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
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 205](#)
- [“add-header” on page 206](#)

delete-file

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 7-83 delete-file Parameters

Parameter	Description
type	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “ Service ” on page 203.
method	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “ Service ” on page 203.
query	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “ Service ” on page 203.
UseOutputStreamSize	(Optional) Common to all <code>Service</code> -class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “ Service ” on page 203.
flushTimer	(Optional) Common to all <code>Service</code> -class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “ Service ” on page 203.
ChunkedRequestBufferSize	(Optional) Common to all <code>Service</code> -class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “ Service ” on page 203.
ChunkedRequestTimeout	(Optional) Common to all <code>Service</code> -class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “ Service ” on page 203.
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “ The bucket Parameter ” on page 138.

Example

```
Service method="DELETE" fn="delete-file"
```

imagemap

The `imagemap` function responds to requests for `imagemaps`. `Imagemaps` are images that are divided into multiple areas and 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 7-84 imagemap Parameters

Parameter	Description
type	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203 .
method	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203 .
query	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203 .
UseOutputStreamSize	(Optional) Common to all <code>Service</code> -class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203 .
flushTimer	(Optional) Common to all <code>Service</code> -class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203 .
ChunkedRequestBufferSize	(Optional) Common to all <code>Service</code> -class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203 .
ChunkedRequestTimeout	(Optional) Common to all <code>Service</code> -class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203 .
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

index-common

The `index-common` function generates a fancy (or common) list of files in the requested directory. The list is sorted alphabetically. Files beginning with a period (.) are not displayed. Each item appears as an HTML link. This function displays more information than `index-simple`, including the size, last modified date, and an icon for each file. It may also include a header and readme file in the listing.

The Init-class function `cindex-init` in `magnus.conf` specifies the format for the index list, including where to look for the images. If `obj.conf` contains a call to `index-common` in the Service stage, `magnus.conf` must initialize fancy (or common) indexing by invoking `cindex-init` during the Init stage.

Indexing occurs when the requested resource is a directory that does not contain an index file or a home page, or no index file or home page has been specified by the functions `find-index` or `home-page`.

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

TABLE 7-85 Content-Type Icons

Content-Type	Icon
"text/*"	text.gif
"image/*"	image.gif
"audio/*"	sound.gif
"video/*"	movie.gif
"application/octet-stream"	binary.gif
Directory	menu.gif
All others	unknown.gif

Parameters

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

TABLE 7-86 index-common Parameters

Parameter	Description
header	(Optional) Path (relative to the directory being indexed) and name of a file (HTML or plain text) that is included at the beginning of the directory listing to introduce the contents of the directory. The file is first tried with <code>.html</code> added to the end. If found, it is incorporated near the top of the directory list as HTML. If the file is not found, it is tried without the <code>.html</code> and incorporated as pre-formatted plain text (bracketed by <code><PRE></code> and <code></PRE></code>).

TABLE 7-86 index-common Parameters (Continued)

Parameter	Description
readme	(Optional) Path (relative to the directory being indexed) and name of a file (HTML or plain text) to append to the directory listing. This file might give more information about the contents of the directory, indicate copyrights, authors, or other information. The file is first tried with <code>.html</code> added to the end. If found, it is incorporated at the bottom of the directory list as HTML. If the file is not found, it is tried without the <code>.html</code> and incorporated as pre-formatted plain text (enclosed within the PRE tag).
type	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “ Service ” on page 203.
method	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “ Service ” on page 203.
query	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “ Service ” on page 203.
UseOutputStreamSize	(Optional) Common to all <code>Service</code> -class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “ Service ” on page 203.
flushTimer	(Optional) Common to all <code>Service</code> -class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “ Service ” on page 203.
ChunkedRequestBufferSize	(Optional) Common to all <code>Service</code> -class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “ Service ” on page 203.
ChunkedRequestTimeout	(Optional) Common to all <code>Service</code> -class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “ Service ” on page 203.
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “ The bucket Parameter ” on page 138.

Example

```
Service type="magnus-internal/directory" method="(GET|HEAD)"
fn="index-common" header="hdr" readme="rdme.txt"
```

See Also

- “[index-simple](#)” on page 214
- “[find-index](#)” on page 159
- “[home-page](#)” on page 145

index-simple

The `index-simple` function generates a simple index of the files in the requested directory. It scans a directory and returns an HTML page to the browser displaying a list of the files and directories in the directory. The list is sorted alphabetically. Files beginning with a period (.) are not displayed. Each item appears as an HTML link.

Indexing occurs when the requested resource is a directory that does not contain either an index file or a home page, or no index file or home page has been specified by the functions `find-index` or `home-page`.

Parameters

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

TABLE 7-87 `index-simple` Parameters

Parameter	Description
<code>type</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “ Service ” on page 203.
<code>method</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “ Service ” on page 203.
<code>query</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “ Service ” on page 203.
<code>UseOutputStreamSize</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “ Service ” on page 203.
<code>flushTimer</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “ Service ” on page 203.

TABLE 7-87 index-simple Parameters (Continued)

Parameter	Description
ChunkedRequestBufferSize	(Optional) Common to all Service-class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203.
ChunkedRequestTimeout	(Optional) Common to all Service-class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203.
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138.

Example

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

See Also

- “find-index” on page 159
- “home-page” on page 145

key-toosmall

The key-toosmall function returns a message to the client specifying that the secret key size for SSL communications is too small. This function is designed to be used together with a Client tag to limit access of certain directories to non-exportable browsers.

Note – This function is provided for backward compatibility only and was deprecated in iPlanet™ Web Server 4.x. It is replaced by the PathCheck-class SAF ssl-check.

Parameters

The following table describes parameters for the key-toosmall function.

TABLE 7-88 key-toosmall Parameters

Parameter	Description
type	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203.

TABLE 7-88 key-toosmall Parameters (Continued)

Parameter	Description
method	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203 .
query	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203 .
UseOutputStreamSize	(Optional) Common to all Service-class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203 .
flushTimer	(Optional) Common to all Service-class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203 .
ChunkedRequestBufferSize	(Optional) Common to all Service-class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203 .
ChunkedRequestTimeout	(Optional) Common to all Service-class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203 .
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
<Object ppath=/mydocs/secret/*>
Service fn="key-toosmall"
</Object>
```

See Also

[“ssl-check” on page 168](#)

list-dir

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
```

where:

- *name* 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 ASCII number of the character.
- *type* is a MIME type such as `text/html`. Directories will be of type `directory`. A file for which the server does not have a type will be of type `unknown`.
- *size* is the size of the file, in bytes.
- *mimetype* is the numerical representation of the date of 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 7-89 `list-dir` Parameters

Parameter	Description
<code>type</code>	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203.
<code>method</code>	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203.
<code>query</code>	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203.
<code>UseOutputStreamSize</code>	(Optional) Common to all Service-class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203.
<code>flushTimer</code>	(Optional) Common to all Service-class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203.
<code>ChunkedRequestBufferSize</code>	(Optional) Common to all Service-class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203.
<code>ChunkedRequestTimeout</code>	(Optional) Common to all Service-class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203.

TABLE 7-89 list-dir Parameters (Continued)

Parameter	Description
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
Service method="INDEX" fn="list-dir"
```

make-dir

The `make-dir` function creates a directory when the client sends a request whose method is `MKDIR`. The function fails if the server can not 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 7-90 make-dir Parameters

Parameter	Description
type	(Optional) Common to all <code>Service-class</code> functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203 .
method	(Optional) Common to all <code>Service-class</code> functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203 .
query	(Optional) Common to all <code>Service-class</code> functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203 .
UseOutputStreamSize	(Optional) Common to all <code>Service-class</code> functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203 .
flushTimer	(Optional) Common to all <code>Service-class</code> functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203 .

TABLE 7-90 `make-dir` Parameters (Continued)

Parameter	Description
<code>ChunkedRequestBufferSize</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203 .
<code>ChunkedRequestTimeout</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203 .
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
Service method="MKDIR" fn="make-dir"
```

proxy-retrieve

The `proxy-retrieve` function retrieves a document from a remote server and returns it to the client. This function also enables you to configure the server to allow or block arbitrary methods. This function only works on the HTTP protocol.

Parameters

The following table describes parameters for the `proxy-retrieve` function.

TABLE 7-91 `proxy-retrieve` Parameters

Parameter	Description
<code>type</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203 .
<code>method</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203 .
<code>query</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203 .
<code>UseOutputStreamSize</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203 .

TABLE 7-91 proxy-retrieve Parameters (Continued)

Parameter	Description
flushTimer	(Optional) Common to all Service-class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203 .
ChunkedRequestBufferSize	(Optional) Common to all Service-class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203 .
ChunkedRequestTimeout	(Optional) Common to all Service-class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203 .
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

See Also

- [“set-origin-server” on page 201](#)
- [“set-proxy-server” on page 202](#)

remove-dir

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 does not 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 7-92 remove-dir Parameters

Parameter	Description
type	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203 .
method	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203 .
query	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203 .
UseOutputStreamSize	(Optional) Common to all Service-class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203 .
flushTimer	(Optional) Common to all Service-class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203 .
ChunkedRequestBufferSize	(Optional) Common to all Service-class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203 .
ChunkedRequestTimeout	(Optional) Common to all Service-class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203 .
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
Service fn="remove-dir" method="RMDIR"
```

rename-file

The `rename-file` function renames a file when the client sends a request with a `New-URL` header whose method is `MOVE`. It renames the file indicated by the URL to `New-URL` within the same directory if the user is authorized and the server has the required 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 7-93 `rename - file` Parameters

Parameter	Description
<code>type</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203 .
<code>method</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203 .
<code>query</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203 .
<code>UseOutputStreamSize</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203 .
<code>flushTimer</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203 .
<code>ChunkedRequestBufferSize</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203 .
<code>ChunkedRequestTimeout</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203 .
<code>bucket</code>	(Optional) Common to all <code>obj . conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
Service fn="rename-file" method="MOVE"
```

send-cgi

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.

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

- Adjust the interval between flushes using the `FlushTimer` parameter. For more information, see “Service” on page 203.
- Adjust the buffer size using the `UseOutputStreamSize` parameter. For more information, see “Service” on page 203.
- Force the 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 7-94 `send-cgi` Parameters

Parameter	Description
<code>user</code>	(UNIX only) Specifies the name of the user to execute CGI programs.
<code>group</code>	(UNIX only) Specifies the name of the group to execute CGI programs.
<code>chroot</code>	(UNIX only) Specifies the directory to <code>chroot</code> to before execution begins.
<code>dir</code>	(UNIX only) Specifies the directory to <code>chdir</code> to after <code>chroot</code> , but before execution begins.
<code>rlimit_as</code>	(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 the limits are set to this value.
<code>rlimit_core</code>	(UNIX only) Specifies the maximum CGI program core file size. A value of <code>0</code> 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 the limits are set to this value.
<code>rlimit_nofile</code>	(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 the limits are set to this value.
<code>nice</code>	(UNIX only) Accepts an increment that determines the CGI program's priority relative to the server. Typically, the server is run with a <code>nice</code> value of <code>0</code> and the <code>nice</code> increment would be from <code>0</code> (the CGI program runs at same priority as server) to <code>19</code> (the CGI program runs at much lower priority than server). Do not increase the priority of the CGI program above that of the server by specifying a <code>nice</code> increment of <code>-1</code> .
<code>type</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203.

TABLE 7-94 send-cgi Parameters (Continued)

Parameter	Description
method	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203 .
query	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203 .
UseOutputStreamSize	(Optional) Common to all Service-class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203 .
flushTimer	(Optional) Common to all Service-class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203 .
ChunkedRequestBufferSize	(Optional) Common to all Service-class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203 .
ChunkedRequestTimeout	(Optional) Common to all Service-class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203 .
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

The following example uses variables defined in the `server.xml` file for the `send-cgi` parameters. For more information about defining variables, see [“Variables” on page 38](#).

```
<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-file

The `send-file` function sends the contents of the requested file to the client. It provides the `Content-Type`, `Content-Length`, and `Last-Modified` headers.

Most requests are handled by this function using the following directive (which usually comes last in the list of `Service-class` directives in the default object, so that it acts as a default):

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

This directive is invoked if the method of the request is `GET`, `HEAD`, or `POST`, and the type does *not* start with `magnus-internal/`. Note that the pattern `*~` means “does not match”.

Parameters

The following table describes parameters for the `send-file` function.

TABLE 7-95 `send-file` Parameters

Parameter	Description
<code>nocache</code>	(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.
<code>type</code>	(Optional) Common to all <code>Service-class</code> functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “ Service ” on page 203.
<code>method</code>	(Optional) Common to all <code>Service-class</code> functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “ Service ” on page 203.
<code>query</code>	(Optional) Common to all <code>Service-class</code> functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “ Service ” on page 203.
<code>UseOutputStreamSize</code>	(Optional) Common to all <code>Service-class</code> functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “ Service ” on page 203.
<code>flushTimer</code>	(Optional) Common to all <code>Service-class</code> functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “ Service ” on page 203.

TABLE 7-95 send-file Parameters (Continued)

Parameter	Description
ChunkedRequestBufferSize	(Optional) Common to all Service-class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203 .
ChunkedRequestTimeout	(Optional) Common to all Service-class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203 .
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

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

When the client requests a portion of a document by specifying HTTP byte ranges, the send-range function returns that portion.

Parameters

The following table describes parameters for the send-range function.

TABLE 7-96 send - range Parameters

Parameter	Description
type	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203 .
method	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203 .
query	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203 .
UseOutputStreamSize	(Optional) Common to all Service-class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203 .
flushTimer	(Optional) Common to all Service-class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203 .
ChunkedRequestBufferSize	(Optional) Common to all Service-class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203 .
ChunkedRequestTimeout	(Optional) Common to all Service-class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203 .
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
Service fn="send-range"
```

send-shellcgi

(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.

Parameters

The following table describes parameters for the `send-shellcgi` function.

TABLE 7-97 send-shellcgi Parameters

Parameter	Description
type	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203 .
method	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203 .
query	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203 .
UseOutputStreamSize	(Optional) Common to all <code>Service</code> -class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203 .
flushTimer	(Optional) Common to all <code>Service</code> -class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203 .
ChunkedRequestBufferSize	(Optional) Common to all <code>Service</code> -class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203 .
ChunkedRequestTimeout	(Optional) Common to all <code>Service</code> -class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203 .
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
Service fn="send-shellcgi"
```

```
Service type="magnus-internal/cgi" fn="send-shellcgi"
```

send-wincgi

(Windows only) The `send-wincgi` function runs a file as a Windows CGI program and sends the results to the client.

Parameters

The following table describes parameters for the `send-wincgi` function.

TABLE 7-98 send-wincgi Parameters

Parameter	Description
type	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203 .
method	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203 .
query	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203 .
UseOutputStreamSize	(Optional) Common to all <code>Service</code> -class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203 .
flushTimer	(Optional) Common to all <code>Service</code> -class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203 .
ChunkedRequestBufferSize	(Optional) Common to all <code>Service</code> -class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203 .
ChunkedRequestTimeout	(Optional) Common to all <code>Service</code> -class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203 .
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
Service fn="send-wincgi"
```

```
Service type="magnus-internal/cgi" fn="send-wincgi"
```

service-dav

The `service-dav` function services a request to a WebDAV-enabled URI. In response to a request for a WebDAV resource, the `service-dav` function services the static content and restarts the request with the `sourceuri` for dynamic content. The `sourceuri` is identified by the `magnus-internal` setting. If no `sourceuri` is defined for dynamic content, an HTTP error message is returned.

Requests to WebDAV resources are authenticated and authorized by the AuthTrans and PathCheck NSAPI stages, respectively. By default, all access to sourceuri is restricted by the PathCheck entry in the dav object.

OPTIONS on a WebDAV-enabled URI are always handled by the service-dav directive of the default object. 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 [Chapter 10, “Web Publishing With WebDAV,”](#) in *Sun Java System Web Server 7.0 Update 3 Administrator’s Guide*.

Parameters

The following table describes parameters for the service-dav function.

TABLE 7-99 service-dav Parameters

Parameter	Description
type	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203.
method	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203.
query	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203.
UseOutputStreamSize	(Optional) Common to all Service-class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203.
flushTimer	(Optional) Common to all Service-class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203.
ChunkedRequestBufferSize	(Optional) Common to all Service-class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203.
ChunkedRequestTimeout	(Optional) Common to all Service-class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203.

TABLE 7-99 service-dav Parameters (Continued)

Parameter	Description
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
<Object name="default">
  . . . .
  Service method="(OPTIONS|PUT|DELETE|COPY|MOVE|PROPFIND|PROPPATCH|LOCK|UNLOCK|MKCOL)"
    fn="service-dav"
</Object>
```

See Also

- [“ntrans-dav” on page 147](#)
- [“pcheck-dav” on page 166](#)

service-dump

The service-dump function creates a performance report based on collected performance bucket data. To read the report, point the browser to:

```
http://server_id:portURI
```

For example:

```
http://sun.com:80/.perf
```

Parameters

The following table describes parameters for the service-dump function.

TABLE 7-100 service-dump Parameters

Parameter	Description
type	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203 .
method	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203 .

TABLE 7-100 service-dump Parameters (Continued)

Parameter	Description
query	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203 .
UseOutputStreamSize	(Optional) Common to all Service-class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203 .
flushTimer	(Optional) Common to all Service-class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203 .
ChunkedRequestBufferSize	(Optional) Common to all Service-class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203 .
ChunkedRequestTimeout	(Optional) Common to all Service-class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203 .
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
<Object name="default">
<If uri = "/.perf">
Service fn="service-dump"
</If>
...
</Object>
```

See Also

[“stats-xml” on page 236](#)

service-j2ee

The service-j2ee function services requests made to Java web applications.

Parameters

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

TABLE 7-101 service-j2ee Parameters

Parameter	Description
type	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203 .
method	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203 .
query	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203 .
UseOutputStreamSize	(Optional) Common to all Service-class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203 .
flushTimer	(Optional) Common to all Service-class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203 .
ChunkedRequestBufferSize	(Optional) Common to all Service-class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203 .
ChunkedRequestTimeout	(Optional) Common to all Service-class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203 .
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

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

See Also

- [“ntrans-j2ee” on page 148](#)
- [“error-j2ee” on page 240](#)

- [“find-index-j2ee” on page 160](#)
- [“type-j2ee” on page 194](#)

service-trace

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 7-102 `service-trace` Parameters

Parameter	Description
<code>type</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203 .
<code>method</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203 .
<code>query</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203 .
<code>UseOutputStreamSize</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203 .
<code>flushTimer</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203 .
<code>ChunkedRequestBufferSize</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203 .
<code>ChunkedRequestTimeout</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203 .
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
<Object name="default">
...
Service method="TRACE" fn="service-trace"
...
</Object>
```

shtml-send

The `shtml - send` function parses an HTML document, scanning for embedded commands. These commands may provide information from the server, include the contents of other files, or execute a CGI program.

See [Chapter 2, “Server-Parsed HTML Tags,” in *Sun Java System Web Server 7.0 Update 3 Developer’s Guide*](#) for server-parsed HTML commands.

Parameters

The following table describes parameters for the `shtml - send` function.

TABLE 7-103 `shtml - send` Parameters

Parameter	Description
<code>ShtmlMaxDepth</code>	Maximum depth of include nesting allowed. The default value is 10.
<code>addCgiInitVars</code>	(UNIX only) If present and set to yes, adds the environment variables defined in the <code>init - cgi</code> SAF to the environment of any command executed through the SHTML exec tag. The default is no.
<code>type</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203 .
<code>method</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203 .
<code>query</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203 .
<code>UseOutputStreamSize</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203 .

TABLE 7-103 shtml-send Parameters (Continued)

Parameter	Description
flushTimer	(Optional) Common to all Service-class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203 .
ChunkedRequestBufferSize	(Optional) Common to all Service-class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203 .
ChunkedRequestTimeout	(Optional) Common to all Service-class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203 .
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

stats-xml

The stats-xml function creates a performance report in XML format. If performance buckets are defined, this performance report includes them.

The report is generated at:

```
http://server_id:portURI
```

For example:

```
http://sun.com:80/stats-xml
```

For more information about tuning the server using the stats-xml information, see [Sun Java System Web Server 7.0 Update 3 Performance Tuning, Sizing, and Scaling Guide](#).

Parameters

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

TABLE 7-104 stats-xml Parameters

Parameter	Description
type	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203 .
method	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203 .
query	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203 .
UseOutputStreamSize	(Optional) Common to all Service-class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203 .
flushTimer	(Optional) Common to all Service-class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203 .
ChunkedRequestBufferSize	(Optional) Common to all Service-class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203 .
ChunkedRequestTimeout	(Optional) Common to all Service-class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203 .
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
<Object name="default">
<If uri = "/stats-xml/*">
Service fn="stats-xml"
</If>
...
</Object>
```

See Also

[“service-dump” on page 231](#)

upload-file

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 7-105 `upload-file` Parameters

Parameter	Description
<code>type</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203 .
<code>method</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203 .
<code>query</code>	(Optional) Common to all <code>Service</code> -class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203 .
<code>UseOutputStreamSize</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203 .
<code>flushTimer</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203 .
<code>ChunkedRequestBufferSize</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203 .
<code>ChunkedRequestTimeout</code>	(Optional) Common to all <code>Service</code> -class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203 .
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
Service fn="upload-file"
```

AddLog

The AddLog directives are executed to record information about the transaction. For more information, see [“AddLog” on page 134](#).

The following AddLog-class function is described in detail in this section:

- [“flex-log” on page 239](#)

In addition, the following common SAFs are valid for the AddLog directive:

- [“match-browser” on page 244](#)
- [“set-variable” on page 252](#)

flex-log

The flex-log function records request-specific data in a flexible log format. It can also record requests in the common log format. There is a log analyzer, flexanlg, in the /bin directory for Web Server. There are also a number of free statistics generators for the common log format.

Specify the log format using the format subelement of the access-log element in server.xml. For more information, see [“access-log” on page 43](#). For more information on the log format, see [Appendix C, “Using the Custom Log File Format.”](#)

Parameters

The following table describes parameters for the flex-log function.

TABLE 7-106 flex-log Parameters

Parameter	Description
name	(Optional) Specifies the name of a log file. The name must previously been defined by an access-log element in server.xml. If no name is given, the entry is recorded in the default log file.
iponly	(Optional) Instructs the server to log the IP address of the remote client rather than looking up and logging the DNS name. This improves performance if DNS is turned off. The value of iponly has no significance, as long as it exists; you may use iponly=1.
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
# Log all accesses to the default 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>
```

Error

If a SAF results in an error, the server stops executing all other directives and immediately starts executing the Error directives. For more information, see [“Error” on page 134](#).

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

- [“error-j2ee” on page 240](#)
- [“qos-error” on page 241](#)

In addition, the following common SAFs are valid for the Error directive:

- [“match-browser” on page 244](#)
- [“query-handler” on page 245](#)
- [“redirect” on page 246](#)
- [“remove-filter” on page 248](#)
- [“restart” on page 249](#)
- [“send-error” on page 250](#)
- [“set-variable” on page 252](#)

error-j2ee

The error-j2ee function handles errors that occur during execution of web applications deployed to the Web Server individually or as part of full Java EE applications.

Parameters

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

TABLE 7-107 error-j2ee Parameter

Parameter	Description
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

See Also

- [“find-index-j2ee” on page 160](#)
- [“ntrans-j2ee” on page 148](#)
- [“service-j2ee” on page 232](#)
- [“type-j2ee” on page 194](#)

qos-error

The `qos-error` function returns an error page stating the quality of service that caused the error, and the value of the QOS statistic.

Parameters

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

TABLE 7-108 qos-error Parameters

Parameter	Description
code	<p>(Optional) Three-digit number representing the HTTP response status code, such as <code>401</code> or <code>407</code>. The recommended value is <code>503</code>.</p> <p>This can be any HTTP response status code or reason phrase according to the HTTP specification.</p> <p>A list of common HTTP response status codes and reason strings is as follows:</p> <ul style="list-style-type: none"> ▪ <code>401</code> Unauthorized ▪ <code>403</code> Forbidden ▪ <code>404</code> Not Found ▪ <code>500</code> Server Error
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

```
Error fn="qos-error" code="503"
```

See Also

[“qos-handler” on page 141](#)

Common SAFs

This section lists SAFs that are common to multiple directives.

TABLE 7-109 Common SAFs

Server Application Functions	Directives
“insert-filter” on page 243	<ul style="list-style-type: none"> ▪ “Input” on page 194 ▪ “Output” on page 196
“match-browser” on page 244	<ul style="list-style-type: none"> ▪ “AuthTrans” on page 138 ▪ “NameTrans” on page 142 ▪ “PathCheck” on page 153 ▪ “ObjectType” on page 170 ▪ “Input” on page 194 ▪ “Output” on page 196 ▪ “Route” on page 200 ▪ “Service” on page 203 ▪ “AddLog” on page 239 ▪ “Error” on page 240
“query-handler” on page 245	<ul style="list-style-type: none"> ▪ “Service” on page 203 ▪ “Error” on page 240
“redirect” on page 246	<ul style="list-style-type: none"> ▪ “NameTrans” on page 142 ▪ “Output” on page 196 ▪ “Error” on page 240
“remove-filter” on page 248	<ul style="list-style-type: none"> ▪ “Input” on page 194 ▪ “Output” on page 196 ▪ “Service” on page 203 ▪ “Error” on page 240
“restart” on page 249	<ul style="list-style-type: none"> ▪ “NameTrans” on page 142
“send-error” on page 250	<ul style="list-style-type: none"> ▪ “Service” on page 203 ▪ “Error” on page 240

TABLE 7-109 Common SAFs (Continued)

Server Application Functions	Directives
“set-variable” on page 252	<ul style="list-style-type: none"> ■ “AuthTrans” on page 138 ■ “NameTrans” on page 142 ■ “PathCheck” on page 153 ■ “ObjectType” on page 170 ■ “Input” on page 194 ■ “Output” on page 196 ■ “Route” on page 200 ■ “Service” on page 203 ■ “AddLog” on page 239 ■ “Error” on page 240

insert-filter

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 is 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 7-110 `insert-filter` Parameters

Parameter	Description
<code>filter</code>	Specifies the name of the filter to insert. For more information about predefined filters, see “Input” on page 194 and “Output” on page 196 .
<code>type</code>	(Optional) Common to all <code>Input-class</code> and <code>Output-class</code> functions. Specifies a wildcard pattern of MIME types for which this function will be executed.
<code>method</code>	(Optional) Common to all <code>Input-class</code> and <code>Output-class</code> functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. Common HTTP methods are GET, HEAD, and POST.

TABLE 7-110 `insert-filter` Parameters (Continued)

Parameter	Description
<code>query</code>	(Optional) Common to all <code>Input-class</code> and <code>Output-class</code> functions. Specifies a wildcard pattern of query strings for which this function will be executed.
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

The above directive instructs the `insert-filter` function to add a custom filter, that is, `http-decompression` to the filter stack. The `http-decompression` filter will decompress the incoming HTTP request data, before it goes to the service stage. For more information about predefined filters, see [“Input” on page 194](#) and [“Output” on page 196](#). For more information, Chapter 3, [“Creating Custom Filters,” in *Sun Java System Web Server 7.0 Update 3 NSAPI Developer’s Guide*](#)

See Also

- [“sed-request” on page 195](#)
- [“sed-response” on page 200](#)

match-browser

The `match-browser` function matches specific strings in the `User-Agent` string supplied by the browser. It then modifies the behavior of Sun Java System Web Server based on the results by setting values for specified variables. This function is applicable in all directives.

Syntax

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

Parameters

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

TABLE 7-111 match-browser Parameters

Value	Description
<i>stage</i>	Stage directive used in <code>obj.conf</code> processing. The <code>match-browser</code> function is applicable in all stage directives.
<i>string</i>	Wildcard pattern to compare with the User-Agent header (for example, <code>*Mozilla*</code>).
<i>name</i>	Variable to be changed. The <code>match-browser</code> function indirectly invokes the <code>set-variable</code> function.
<i>value</i>	New value for the specified variable.

Example

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

If a browser's User-Agent header contains the string `Broken` or `broken`, the above `AuthTrans` directive instructs the server to do the following:

- Not send the SSL3 and TLS `close_notify` packet
- Not honor requests for HTTP Keep-Alive
- Use the HTTP/1.0 protocol rather than HTTP/1.1

For more information on the variables used in this example, such as `ssl-unclean-shutdown`, see [“set-variable” on page 252](#).

See Also

[“set-variable” on page 252](#)

query-handler

The `query-handler` function runs a CGI program instead of referencing the path requested.

Note – This function is provided for backward compatibility only and is used mainly to support the obsolete `ISINDEX` tag. Use an HTML form instead.

Parameters

The following table describes parameters for the `query-handler` function.

TABLE 7-112 query-handler Parameters

Parameter	Description
path	Full path and file name of the CGI program to run.
type	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203 .
method	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203 .
query	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203 .
UseOutputStreamSize	(Optional) Common to all Service-class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203 .
flushTimer	(Optional) Common to all Service-class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203 .
ChunkedRequestBufferSize	(Optional) Common to all Service-class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203 .
ChunkedRequestTimeout	(Optional) Common to all Service-class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203 .
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

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

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

redirect

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.

The `redirect` function inspects the URL to which the client will be redirected. If the URL matches the URL the client has requested (same scheme, hostname, port, and path), this function does not perform the redirect and instead returns `REQ_NOACTION`.

Parameters

The following table describes parameters for the `redirect` function.

TABLE 7-113 `redirect` Parameters

Parameter	Description
<code>from</code>	(Optional) Specifies the prefix of the requested URI to match. If <code>from</code> is not specified, it defaults to "".
<code>url</code>	(Optional) Specifies a complete URL to return to the client. If you use this parameter, do not use <code>url-prefix</code> .
<code>url-prefix</code>	(Optional) The new URL prefix to return to the client. The <code>from</code> prefix is replaced by this URL prefix. If you use this parameter, do not use <code>url</code> .
<code>escape</code>	(Optional) Indicates whether the value of the <code>url</code> or <code>url-prefix</code> parameter needs to be escaped. The default is <code>yes</code> , indicating that the server will escape the value. The value <code>no</code> indicates that the URL or URL prefix value has already been escaped. An example of an escaped value is one where any <code>%</code> characters have been replaced with <code>%25</code> and any spaces have been replaced with <code>%20</code> . For more information about escaping URIs, see the description of <code>util_uri_escape</code> , in the <i>Sun Java System Web Server 7.0 Update 3 NSAPI Developer's Guide</i> .
<code>status</code>	(Optional) Customizes the HTTP status code. If <code>status</code> is not specified, it defaults to <code>302</code> .
<code>type</code>	(Optional) Common to all <code>Output</code> -class functions. Specifies a wildcard pattern of MIME types for which this function will be executed.
<code>method</code>	(Optional) Common to all <code>Output</code> -class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. Common HTTP methods are <code>GET</code> , <code>HEAD</code> , and <code>POST</code> .
<code>query</code>	(Optional) Common to all <code>Output</code> -class functions. Specifies a wildcard pattern of query strings for which this function will be executed.
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The <code>bucket</code> Parameter” on page 138.

Example

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/`.

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

See Also

[“restart” on page 249](#)

remove-filter

The `remove-filter` SAF is used to remove a filter from the filter stack. If the filter is inserted multiple times, only the topmost instance is removed. In general, it is not necessary to remove filters with `remove-filter`, as they are removed automatically at the end of a 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 7-114 `remove-filter` Parameters

Parameter	Description
<code>filter</code>	Specifies the name of the filter to remove.
<code>type</code>	(Optional) Common to all <code>Input-class</code> , <code>Output-class</code> , and <code>Service-class</code> functions. Specifies a wildcard pattern of MIME types for which this function will be executed. The <code>magnus-internal/*</code> MIME types are used only to select a <code>Service</code> function to execute.

TABLE 7-114 `remove-filter` Parameters (Continued)

Parameter	Description
<code>method</code>	(Optional) Common to all <code>Input-class</code> , <code>Output-class</code> , and <code>Service-class</code> functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. Common HTTP methods are GET, HEAD, and POST.
<code>query</code>	(Optional) Common to all <code>Input-class</code> , <code>Output-class</code> , and <code>Service-class</code> functions. Specifies a wildcard pattern of query strings for which this function will be executed.
<code>UseOutputStreamSize</code>	(Optional) Common to all <code>Service-class</code> functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “ Service ” on page 203.
<code>flushTimer</code>	(Optional) Common to all <code>Service-class</code> functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “ Service ” on page 203.
<code>ChunkedRequestBufferSize</code>	(Optional) Common to all <code>Service-class</code> functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “ Service ” on page 203.
<code>ChunkedRequestTimeout</code>	(Optional) Common to all <code>Service-class</code> functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “ Service ” on page 203.
<code>bucket</code>	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “ The bucket Parameter ” on page 138.

Example

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

restart

The `restart` function allows URL rewriting within the server without sending an HTTP redirect to the client. The `restart` function replaces the `uri` and `query` values in `rq->reqpb` with the URI and query string specified by the `uri` parameter and restarts the request by returning `REQ_RESTART`.

If the `uri` parameter contains a `?` character, the value following `?` is used as the query string. Otherwise, the restarted request will not have a query string. Because the new request URI will be passed through the `AuthTrans` and `NameTrans` stages again, avoid creating infinite loops.

Parameters

The following table describes parameters for the `restart` function.

TABLE 7-115 restart Parameters

Parameter	Description
from	(Optional) Wildcard pattern that specifies the path of requests that should be restarted. The default is to match all paths.
uri	URI and query string to use for the restarted request.
bucket	(Optional) Common to all <code>obj.conf</code> functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138 .

Example

The following `obj.conf` code causes the server to service requests for `/index.html` as though they were requests for `/index.jsp`:

```
NameTrans fn="restart" from="/index.html" uri="/index.jsp"
```

send-error

The `send-error` function sends an HTML file to the client in place of a specific HTTP response status. The HTML page may contain images and links to the server home page or other pages.

Parameters

The following table describes parameters for the `send-error` function.

TABLE 7-116 send-error Parameters

Parameter	Description
path	Specifies the full file system path of an HTML file to send to the client. The file is sent as <code>text/html</code> regardless of its name or actual type. If the file does not exist, the server sends a simple default error page.
response	(Optional) Text of one of the reason strings (such as “Unauthorized” or “Forbidden”). The string is not case-sensitive.

TABLE 7-116 send-error Parameters (Continued)

Parameter	Description
code	<p>(Optional) Three-digit number representing the HTTP response status code, such as 401 or 407.</p> <p>This can be any HTTP response status code or reason phrase according to the HTTP specification.</p> <p>The following is a list of common HTTP response status codes and reason strings:</p> <ul style="list-style-type: none"> ■ 401 Unauthorized ■ 403 Forbidden ■ 404 Not Found ■ 500 Server Error
uri	(Optional) URI of the resource to send to the client. The URI can specify any resource on the server, including HTML files, SHTML pages, CGI programs, JSPs, and Servlets. If the specified resource does not exist, the HTML file specified by the path parameter is sent instead.
type	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see “Service” on page 203.
method	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see “Service” on page 203.
query	(Optional) Common to all Service-class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see “Service” on page 203.
UseOutputStreamSize	(Optional) Common to all Service-class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see “Service” on page 203.
flushTimer	(Optional) Common to all Service-class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see “Service” on page 203.
ChunkedRequestBufferSize	(Optional) Common to all Service-class functions. Determines the default buffer size, in bytes, for un-chunking request data. For more information, see “Service” on page 203.
ChunkedRequestTimeout	(Optional) Common to all Service-class functions. Determines the default timeout, in seconds, for un-chunking request data. For more information, see “Service” on page 203.
bucket	(Optional) Common to all obj.conf functions. Adds a bucket to monitor performance. For more information, see “The bucket Parameter” on page 138.

Example

```
Error fn="send-error" code="401" path="/sun/server7/docs/errors/401.html"
```

set-variable

The `set-variable` function enables you to change server settings based upon conditional information in a request. This function is applicable in all directives.

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

The `set-variable` function recognizes many predefined variables as parameters. Additionally, when a `set-variable` parameter name begins with `$` but is not the name of a predefined variable, the parameter and its value are stored in the `rq->vars` *pblock*. This functionality allows you to define or override the `$variable` values at the request time.

`set-variable` accepts both the `$variable` and `${variable}` forms, but the name of the parameter stored in the `rq->vars` *pblock* is always in the `$variable` form.

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 7-117 set-variable Parameters

Value	Description
<i>pblock</i>	<p>Specifies one of the following session or request parameter block names:</p> <ul style="list-style-type: none"> ■ <code>client</code>: Contains the IP address of the client machine and the DNS name of the remote machine. ■ <code>vars</code>: Contains the server's working variables, which includes anything not specifically found in the <code>reqpb</code>, <code>headers</code>, or <code>srvhdrs</code> pblocks. The contents of this pblock differ, depending on the specific request and the type of SAF. ■ <code>reqpb</code>: Contains elements of the HTTP request, which includes the HTTP method such as GET or POST, the URI, the protocol (generally HTTP/1.0), and the query string. This pblock does not change during the request-response process. ■ <code>headers</code>: 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 does not change during the request-response process. ■ <code>srvhdrs</code>: Contains the response headers (such as Server, Date, Content-Type, Content-Length, and so on) that are to be sent to the client in the HTTP response.
<i>name</i>	The variable to set.
<i>value</i>	The string assigned to the variable specified by <i>name</i> .

Variables

The following tables lists variables supported by the `set-variable` SAF.

TABLE 7-118 Supported Variables

Variable	Description
<code>abort</code>	A value of <code>true</code> indicates that the result code should be set to <code>REQ_ABORTED</code> . Setting the result code to <code>REQ_ABORTED</code> will abort the current request and send an error to the browser. For information about result codes, see Chapter 2, "Creating Custom Server Application Functions," in <i>Sun Java System Web Server 7.0 Update 3 NSAPI Developer's Guide</i> .

TABLE 7-118 Supported Variables (Continued)

Variable	Description
error	<p>Sets the HTTP status code and exits the request by returning REQ_ABORTED. To set the HTTP status code without exiting the request, use the set-variable error parameter along with the noaction parameter. To rewrite an HTTP status code, use a Client tag to match the original status code and an Output directive to set the new status code.</p> <p>For example, the following code will rewrite all 302 Moved Temporarily responses to 301 Moved Permanently responses:</p> <pre><Client code="302"> Output fn="set-variable" error="301 Moved Permanently" noaction="true" </Client></pre> <p>Sets the error code to be returned in the event of an aborted browser request.</p>
escape	A Boolean value signifying whether a URL should be escaped using util_uri_escape. For information about util_uri_escape, see Sun Java System Web Server 7.0 Update 3 NSAPI Developer's Guide .
find-pathinfo-forward	Path information after the file name in a URI. See “find-pathinfo” on page 162 .
http-downgrade	HTTP version number (for example, 1.0).
http-upgrade	HTTP version number (for example, 1.0).
keep-alive	A Boolean value that establishes whether a keep-alive request from a browser will be honored.
name	Specifies an additional named object in the obj.conf file whose directives will be applied to this request. See also “assign-name” on page 143 .
noaction	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 Chapter 2, “Creating Custom Server Application Functions,” in Sun Java System Web Server 7.0 Update 3 NSAPI Developer's Guide .
nostat	Causes the server <i>not</i> to perform the stat() function for a URL when possible. See also “assign-name” on page 143 .
senthdrs	A Boolean value that indicates whether HTTP response headers have been sent to the client.

TABLE 7-118 Supported Variables (Continued)

Variable	Description
<code>ssl-unclean-shutdown</code>	A Boolean value that can be used to alter the way SSL3 connections are closed. Caution – 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.
<code>stop</code>	A value of <code>true</code> indicates the result code should be set to <code>REQ_PROCEED</code> . For <code>AuthTrans</code> , <code>NameTrans</code> , <code>Service</code> , and <code>Error</code> stage SAFs, setting the result code to <code>REQ_PROCEED</code> indicates that no further SAFs in that stage should be allowed to execute. For information about result codes, Chapter 2, “Creating Custom Server Application Functions,” in <i>Sun Java System Web Server 7.0 Update 3 NSAPI Developer’s Guide</i> .
<code>url</code>	Redirect requests to a specified URL.

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 set the same server class to use HTTP/1.0 while the rest of the server classes use HTTP/1.1, the `AuthTrans` directive is:

```
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 specify `srvhdrs` as your `Session` or `Request` parameter block.

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

```
NameTrans fn="set-variable" insert-srvhdrs="P3P"
```

- To terminate processing a request based on certain URIs, use a `Client` tag to specify the URIs and an `AuthTrans` directive that sets the variable `abort` to `true` when there is a match. Your `Client` tag would be as follows:

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

- To use predefined variables so that the server rewrites redirects to host *badname* as redirects to host *goodname*:

```
<If $srvhdrs{'location'} =~ "^(http|https)://badname/(.*)$"
Output fn="set-variable" $srvhdrs{'location'}="$1://goodname/$2"
</If>
```

- To set a \$variable value at request time:

```
<If "$time_hour:$time_min" < "8:30" || "$time_hour:$time_min" > "17:00">
AuthTrans fn="set-variable" $docroot="/var/www/docs/closed"
</If>
...
NameTrans fn="document-root" root="$docroot"
```

Regardless of whether the \$docroot variable has been defined in server.xml, its value is set to /var/www/docs/closed when the server is accessed after 5:00 p.m. and before 8:00 a.m. local time.

See Also

[“match-browser” on page 244](#)

FastCGI Plug-in SAFs

FastCGI plug-in provides the following Server Application Functions (SAFs):

- auth-fastcgi
- responder-fastcgi
- filter-fastcgi
- error-fastcgi

The various parameters and "error-reason" strings for the FastCGI SAFs are described in the Sun Java System Administrator's Guide.

MIME Types

The MIME types file in the `config` directory contains mappings between the Multipurpose Internet Mail Extensions (MIME) 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 Web Server receives a request from a client, it uses the MIME type mappings to determine the kind of resource that is 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 `mime.types`.

This chapter discusses the following sections:

- “Determining the MIME Type” on page 257
- “Referencing MIME Types Files in `server.xml`” on page 258
- “Generating the Server Response Using the MIME Type” on page 258
- “Processing the Response in the Client Using the MIME Type” on page 259
- “MIME Types Syntax” on page 259
- “Sample MIME Types File” on page 259

Determining the MIME Type

During the `ObjectType` stage in the request handling process, the server determines the MIME type attributes of the resource requested by the client. You can use different SAFs to determine the MIME type. The most commonly used SAF is `type-by-extension`, which tells the server to look up the MIME type according to the requested resource’s file extension in the MIME types table. The MIME types table is stored in a MIME type file. For more information on the format of this file, see “MIME Types Syntax” on page 259.

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, the MIME types table is not used for that particular request.

For more details, see [“ObjectType” on page 170](#).

Referencing MIME Types Files in server.xml

If you create MIME type files, you must reference them in `server.xml` using the `mime-file` element. Because the `mime-file` element can appear as a child element of both the `server` and `virtual-server` elements, you can create MIME types files that apply to the entire server or only to specific virtual servers. For more information, see [“mime-file” on page 74](#).

Generating the Server Response Using the MIME Type

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 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 whichever `Service` directive in `obj.conf` matches the specified `type`. For example, if the `type` is `magnus-internal/imagemap`, the server uses the `imagemap` function to generate the response to the client, as indicated by the following directive:

```
Service method=(GET|HEAD) type=magnus-internal/imagemap fn=imagemap
```

Processing the Response in the Client Using the MIME Type

The Service function generates the data and sends it to the client that made the request. When the server sends the data to the client, it also sends headers. These headers include whichever MIME type attributes are known (which is usually type).

When the client receives the data, it uses the MIME type to decide what to do with the data. For browser clients, the browser usually displays 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.

MIME Types Syntax

The first line in the MIME types file identifies the file format:

```
#--Sun Microsystems MIME Information
```

Other uncommented 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

A sample of the MIME types file is as follows:

```
#--Sun Microsystems Inc. MIME Information
# Do not delete the above line. It is used to identify the file type.
#
# Copyright (c) 2006 Sun Microsystems, Inc. All rights reserved.
# Use is subject to license terms.
#

type=application/octet-stream          exts=bin
type=application/astound                exts=asd,asn
type=application/fastman                exts=lcc
```

type=application/java-archive	exts=jar
type=application/java-serialized-object	exts=ser
type=application/java-vm	exts=class
type=application/mac-binhex40	exts=hqx
type=application/x-stuffit	exts=sit
type=application/mbedlet	exts=mbd
type=application/msword	exts=doc,dot,wiz,rtf
type=application/oda	exts=oda
type=application/pdf	exts=pdf
type=application/postscript	exts=ai,eps,ps
type=application/studiom	exts=smp
type=application/timbuktu	exts=tb
type=application/vnd.ms-excel	exts=xls,xlw,xla,xlc,xlm,xlt
type=application/vnd.ms-powerpoint	exts=ppt,pps,pot
type=application/vnd.ms-project	exts=mpp
type=application/winhelp	exts=hlp
type=application/x-javascript	exts=js
type=application/x-javascript;charset=UTF-8	exts=jsu
type=application/x-java-jnlp-file	exts=jnlp
type=application/x-aim	exts=aim
type=application/x-asap	exts=asp
type=application/x-csh	exts=csh
type=application/x-dvi	exts=dvi
type=application/x-earthtime	exts=etc
type=application/x-envoy	exts=evy
type=application/x-gtar	exts=gtar
type=application/x-cpio	exts=cpio
type=application/x-hdf	exts=hdf
type=application/x-latex	exts=latex
type=application/x-javascript-config	exts=jsc
type=application/x-maker	exts=fm
type=application/x-mif	exts=mif,mi
type=application/x-mocha	exts=mocha,moc
type=application/x-msaccess	exts=mdb
type=application/x-mscardfile	exts=crd
type=application/x-msclip	exts=clp
type=application/x-msmediaview	exts=m13,m14
type=application/x-msmetafile	exts=wmf
type=application/x-msmoney	exts=mny
type=application/x-mspublisher	exts=pub
type=application/x-msschedule	exts=scd
type=application/x-msterminal	exts=trm
type=application/x-mswrite	exts=wri
type=application/x-NET-Install	exts=ins
type=application/x-netcdf	exts=nc,cdf
type=application/x-ns-proxy-autoconfig	exts=proxy
type=application/x-salsa	exts=slc
type=application/x-sh	exts=sh

type=application/x-shar	exts=shar
type=application/x-sprite	exts=spr,sprite
type=application/x-tar	exts=tar
type=application/x-tcl	exts=tcl
type=application/x-perl	exts=pl
type=application/x-tex	exts=tex
type=application/x-texinfo	exts=texinfo, texi
type=application/x-timbuktu	exts=tbp
type=application/x-tkined	exts=tki,tkined
type=application/x-troff-man	exts=man
type=application/x-troff-me	exts=me
type=application/x-troff-ms	exts=ms
type=application/x-troff	exts=t, tr, roff
type=application/x-wais-source	exts=src
type=application/zip	exts=zip
type=application/pre-encrypted	exts=enc
type=application/x-pkcs7-crl	exts=crl
type=application/x-forTEZZa-ckl	exts=ckl
type=application/xml-dtd	exts=dtd
type=audio/basic	exts=au,snd
type=audio/echospeech	exts=es,esl
type=audio/midi	exts=midi,mid
type=audio/x-aiff	exts=aif,aiff,aifc
type=audio/x-wav	exts=wav
type=audio/x-pn-realaudio	exts=ra,ram
type=audio/x-pac	exts=pac
type=audio/x-epac	exts=pae
type=audio/x-liveaudio	exts=lam
type=drawing/x-dwf	exts=dwf
type=image/fif	exts=fif
type=image/x-icon	exts=ico
type=image/gif	exts=gif
type=image/ief	exts=ief
type=image/ifs	exts=ifs
type=image/jpeg	exts=jpeg,jpg,jpe,jfif,pjpeg,pjp
type=image/png	exts=png
type=image/tiff	exts=tiff,tif
type=image/vnd	exts=dwg,svf
type=image/wavelet	exts=wi
type=image/bmp	exts=bmp
type=image/x-photo-cd	exts=pcd
type=image/x-cmu-raster	exts=ras
type=image/x-portable-anymap	exts=pnm
type=image/x-portable-bitmap	exts=pbm
type=image/x-portable-graymap	exts=pgm

type=image/x-portable-pixmap	exts=ppm
type=image/x-rgb	exts=rgb
type=image/x-xbitmap	exts=xbm
type=image/x-xpixmap	exts=xpm
type=image/x-xwindowdump	exts=xwd
type=text/css	exts=css
type=text/html	exts=htm,html
type=text/plain	exts=txt
type=text/richtext	exts=rtx
type=text/tab-separated-values	exts=tsv
type=text/x-setext	exts=etx
type=text/x-speech	exts=talk
type=text/xml	exts=xml
type=text/xul	exts=xul
type=video/isivideo	exts=fvi
type=video/mpeg	exts=mpeg,mpg,mpe,mpv,vbs,mpegv
type=video/x-mpeg2	exts=mpv2,mp2v
type=video/msvideo	exts=avi
type=video/quicktime	exts=qt,mov,moov
type=video/vivo	exts=viv,vivo
type=video/wavelet	exts=ww
type=video/x-sgi-movie	exts=movie
type=x-world/x-svr	exts=svr
type=x-world/x-vrml	exts=wrl
type=x-world/x-vrt	exts=vrt
type=x-conference/x-cooltalk	exts=ice
enc=x-gzip	exts=gz
enc=x-compress	exts=z
enc=x-uencode	exts=uu,uue
type=magnus-internal/imapmap	exts=map
type=magnus-internal/parsed-html	exts=shtml
type=magnus-internal/cgi	exts=cgi,exe,bat
type=application/x-x509-ca-cert	exts=cacert
type=application/x-x509-server-cert	exts=scert
type=application/x-x509-user-cert	exts=ucert
type=application/x-x509-email-cert	exts=ecert
type=application/vnd.sun.xml.writer	exts=sxw
type=application/vnd.sun.xml.writer.template	exts=stw

type=application/vnd.sun.xml.calc	exts=sxc
type=application/vnd.sun.xml.calc.template	exts=stc
type=application/vnd.sun.xml.draw	exts=sxd
type=application/vnd.sun.xml.draw.template	exts=std
type=application/vnd.sun.xml.impress	exts=sxi
type=application/vnd.sun.xml.impress.template	exts=sti
type=application/vnd.sun.xml.writer.global	exts=sxg
type=application/vnd.sun.xml.math	exts=sxm
type=application/vnd.stardivision.writer	exts=sdw
type=application/vnd.stardivision.writer-global	exts=sgl
type=application/vnd.stardivision.calc	exts=sdc
type=application/vnd.stardivision.draw	exts=sda
type=application/vnd.stardivision.impress	exts=sdd
type=application/vnd.stardivision.impress-packed	exts=sdp
type=application/vnd.stardivision.math	exts=smf, sdf
type=application/vnd.stardivision.chart	exts=sds
type=application/vnd.stardivision.mail	exts=sdm

ACL Files

This chapter describes the access control list (ACL) files and their syntax. ACL files are text files containing lists that define who can access resources stored on Web Server. By default, Web Server uses one ACL file that contains the access list. You can, however, create multiple ACL files and reference them in the `obj.conf` file.

After installation, a default access control list is written to the `default.acf` file in the `instance_dir/config` directory. You can change access control rules by editing this file or by creating additional ACL files.

When you make changes to ACL files, you must restart or reconfigure the server for the changes to take effect. For information on reconfiguring the server without restarting, see [“Dynamic Reconfiguration” on page 30](#).

This chapter contains the following sections:

- [“Referencing ACL Files in `server.xml` and `obj.conf`” on page 265](#)
- [“ACL File Syntax” on page 266](#)
- [“Sample ACL File” on page 271](#)

Referencing ACL Files in `server.xml` and `obj.conf`

If you create ACL files, you must reference them in `server.xml` using the `acl-file` element. Because the `acl-file` element can appear as a child element of both `server` and `virtual-server` elements, you can create ACL files that apply to the entire server or only to specific virtual servers. For more information, see [“acl-file” on page 46](#).

If you have named ACLs, you can reference them in the `obj.conf` file. You can do this in the `PathCheck` directive using the `check-acl` function. The line has the following syntax:

```
PathCheck fn="check-acl" acl="aclname"
```

where *aclname* is a unique name of the ACL as it appears in an ACL file.

For example, you can add the following lines to your `obj.conf` file if you want to restrict access to a directory using the ACL named `testacl`:

```
<Object ppath="/var/htdocs/test/*">
PathCheck fn="check-acl" acl="testacl"
</Object>
```

In the above example, the first line is the object that states which server resource to restrict access to. The second line is the `PathCheck` directive that uses the `check-acl` function to bind the named ACL `testacl` to the object in which the directive appears. For more information, see [“check-acl” on page 154](#).

The `testacl` ACL can be defined in any ACL file referenced in `server.xml`.

ACL File Syntax

All ACL files must follow a specific format and syntax. The ACL file is a text file containing one or more ACLs. All ACL files must begin with the version number they use. There can be only one version line and it can appear after any comment line. Web Server uses version 3.0. For example:

```
version 3.0;
```

You can include comments in the file by beginning the comment line with the `#` sign.

Each ACL in the file begins with a statement that defines its type. ACLs can follow one of the three types:

- Path ACLs specify an absolute path to the resource they affect.
- URI ACLs specify a directory or file relative to the server’s document root.
- Named ACLs specify a name that is referenced in the `obj.conf` file. Web Server comes with a default named resource that allows *read access* to all users and *write access* to users in the LDAP directory. Even though you can create a named ACL from the Web Server user interface, you must manually reference the named ACLs with resources in the `obj.conf` file.

Path and URI ACLs can include a wildcard at the end of the entry, for example: `/a/b/*`. Wildcards placed anywhere except at the end of the entry will not work.

The type line begins with the letters `acl` and includes the type information in double-quotation marks followed by a semicolon. Each type information for all ACLs must be a unique name even among different ACL files. The following lines are examples of several different types of ACLs:

```
acl "default";
...
```

```
acl "path=C:/docs/mydocs/";  
...  
  
acl "uri=/mydocs/";  
...
```

After you define the type of ACL, you can have one or more statements that define the method used with the ACL (authentication statements) and the users and computers who are allowed or denied access (authorization statements). The following sections describe the syntax for these statements.

This section includes the following topics:

- [“General Syntax” on page 267](#)
- [“Authentication Methods” on page 267](#)
- [“Authorization Statements” on page 268](#)
- [“Hierarchy of Authorization Statements” on page 269](#)

General Syntax

Input strings can contain the following characters:

- letters a through z
- numbers 0 through 9
- period (.) and underscore (_)

If you use any other characters, add double-quotation marks around the characters. A single statement can be placed on its own line, and terminated with a semicolon. Multiple statements are placed within braces. A list of items must be separated by commas and enclosed in double-quotation marks.

Authentication Methods

ACLs can optionally specify the authentication method that the server must use when processing the ACL. There are three methods:

- `basic`
- `digest`
- `ssl`

The `basic` and `digest` methods require users to enter a user name and password before accessing a resource. The `ssl` method requires the user to have a client certificate. The Web Server must have the encryption turned on, and the user’s certificate issuer must be in the list of trusted certificate authorities (CAs) to be authenticated.

By default, the server uses the `basic` method for any ACL that does not specify a method. If you use the `digest` method, the server's authentication database must be able to handle digest authentication. Authentication databases are configured in `server.xml` with the `auth-db` element. For more information, see [“auth-db” on page 47](#).

Each `authenticate` line must specify the attribute (`users`, `groups`, or both `users` and `groups`) that the server authenticates. The following authentication statement, which appears after the ACL type line, specifies basic authentication with users matched to individual users in the database or directory:

```
authenticate (user) { method = "basic"; };
```

The following example uses `ssl` as the authentication method for users and groups:

```
authenticate (user, group) { method = "ssl"; };
```

The following example allows any user whose user name begins with `sales`:

```
authenticate (user)
allow (all)
user = sales*
```

If the last line is changed to `group = sales`, then the ACL will fail because the `group` attribute is not authenticated.

Authorization Statements

Each ACL entry can include one or more authorization statements. Authorization statements specify who is allowed or denied access to a server resource. Use the following syntax to write authorization statements:

```
allow|deny [absolute] (right[,right...]) attribute expression;
```

Start each line with either `allow` or `deny`. Because of the hierarchy rules, it is usually a good practice to deny access to everyone in the first rule and then specifically allow access for users, groups, or computers in subsequent rules. That is, if you allow anyone access to a directory called `/my_stuff`, and you have a subdirectory `/my_stuff/personal` that allows access to a few users, the access control on the subdirectory will not work because anyone allowed access to the `/my_stuff` directory will also be allowed access to the `/my_stuff/personal` directory. To prevent this, create a rule for the subdirectory that first denies access to anyone and then allows it for the few users who need access.

In some cases, if you set the default ACL to deny access to everyone, your other ACL rules do not need a `deny all` rule.

The following line denies access to everyone:

```
deny (all) user = "anyone";
```

Hierarchy of Authorization Statements

ACLs have a hierarchy that depends on the resource. For example, if the server receives a request for the document (URI) `/my_stuff/web/presentation.html`, the server builds a list of ACLs that apply for this URI. The server first adds ACLs listed in `check-acl` statement of its `obj.conf` file. Then the server appends matching URI and PATH ACLs.

The server processes this list in the same order. Unless absolute ACL statements are present, all statements are evaluated in order. If an absolute `allow` or `absolute deny` statement evaluates to `true`, the server stops processing and accepts this result.

If there is more than one ACL that matches, the server uses the last statement that matches. However, if you use an absolute statement, the server stops looking for other matches and uses the ACL containing the absolute statement. If you have two absolute statements for the same resource, the server uses the first one in the file and stops looking for other resources that match.

```
version 3.0;
acl "default";
authenticate (user, group) {
    prompt = "Sun Java System Web Server";
};
allow (read, execute, info) user = "anyone";
allow (list, write, delete) user = "all";

acl "uri=/my_stuff/web/presentation.html";
deny (all) user = "anyone";
allow (all) user = "user1";
```

Expression Attribute

Attribute expressions define who is allowed or denied access based on their user name, group name, host name, or IP address. The following are examples of allowing access to different users or computers:

- `user = "anyone"`
- `user = "smith*"`
- `group = "sales"`
- `dns = "*.sun.com"`
- `dns = "*.sun.com,*.mozilla.com"`
- `ip = "198.*"`
- `ciphers = "rc4"`

- `ssl = "on"`

You can also restrict access to your server by time of day (based on the local time on the server) by using the `timeofday` attribute. For example, you can use the `timeofday` attribute to restrict access to certain users during specific hours.

Note – Use 24-hour time to specify times. For example, use 0400 to specify 4:00 a.m. or 2230 for 10:30 p.m.

The following example restricts access to a group of users called `guests` between 8:00 a.m. and 4:59 p.m.:

```
allow (read)
```

```
(group="guests") and (timeofday<0800 or timeofday=1700);
```

You can also restrict access by day of the week. Use the following three-letter abbreviations to specify days: Sun, Mon, Tue, Wed, Thu, Fri, and Sat.

The following statement allows access for users in the `premium` group any day and any time. Users in the `discount` group get access all day on weekends and on weekdays, any time except 8 a.m. to 4:59 p.m.

```
allow (read) (group="discount" and dayofweek="Sat,Sun") or (group="discount" and  
(dayofweek="mon,tue,wed,thu,fri" and (timeofday<0800 or timeofday=1700))) or  
(group="premium");
```

Expression Operators

You can use various operators in an expression. Parentheses delineate the operator order of precedence. With `user`, `group`, `dns`, and `ip`, you can use the following operators:

- `and`
- `or`
- `not`
- `=` (equals)
- `!=` (not equal to)

With `timeofday` and `dayofweek`, you can use:

- `>` (greater than)
- `<` (less than)
- `>=` (greater than or equal to)
- `<=` (less than or equal to)

Sample ACL File

After installation, the *instance_dirconfig/default.ac* file provides default settings for the server. When editing an ACL file, you can make changes in the default file, then deploy the changes. You can also create additional ACL files.

A sample *default.ac* file is as follows:

```
version 3.0;
acl "default";
authenticate (user, group) {
    prompt = "Sun Java System Web Server";
};
allow (read, execute, info) user = "anyone";
allow (list, write, delete) user = "all";

acl "es-internal";
allow (read, execute, info) user = "anyone";
deny (list, write, delete) user = "anyone";
```

Note – The above access control rules allow anyone to read resources on the server but restrict listing, writing, and deleting resources to authenticated users.

Other Server Configuration Files

This chapter summarizes the configuration files that are not discussed in other chapters. Configuration files that should never be modified are not listed in this chapter. The following configuration files are described in detail:

- “certmap.conf” on page 273
- “sun-web.xml” on page 275
- “login.conf” on page 275
- “server.policy” on page 275
- “default-web.xml” on page 276

certmap.conf

The `certmap.conf` file configures how a certificate is mapped to an LDAP entry designated by *issuerDN*.

The following table describes the `certmap.conf` file properties.

TABLE 10-1 certmap.conf Properties

Attribute	Allowed Values	Default Value	Description
DNComps	See description	Commented out	Used to form the base DN for performing an LDAP search while mapping the certificate to a user entry. Values are as follows: <ul style="list-style-type: none"> ■ 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
FilterComps	See description	Commented out	Used to form the filter for performing an LDAP search while mapping the certificate to a user entry. Values are as follows: <ul style="list-style-type: none"> ■ Commented out or empty – Sets the filter to "objectclass=*" ■ Comma-separated attributes – Forms the filter
verifycert	on or off	off (commented out)	Specifies whether certificates are verified.
CmapLdapAttr	Name of the LDAP attribute	certSubjectDN (commented out)	Specifies the name of the attribute in the LDAP database that contains the DN of the certificate.
library	Path to shared lib or dll	None	Specifies the library path for custom certificate mapping code.
InitFn	Name of initialization function	None	Specifies the initialization function in the certificate mapping code referenced by library.

Location

instance_dir/config

Syntax

```
certmap name issuerDNname:property1 [value1]
name:property2 [value2]
...
```

The default certificate is named `default`, and the default `issuerDN` is also named `default`. Therefore, the first `certmap.conf` defined in the file must be as follows:

```
certmap default default
```

Use `#` at the beginning of a line to indicate a comment.

See Also

Sun Java System Web Server 7.0 Update 3 Administrator's Guide

sun-web.xml

The `sun-web.xml` file configures the features specific to the Web Server for deployed web applications. For more information about `sun-web.xml`, see *Sun Java System Web Server 7.0 Update 3 Developer's Guide to Java Web Applications*.

Location

The `META-INF` or `WEB-INF` directory of a module or application

login.conf

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

Location

`instance_dir/config`

server.policy

The `server.policy` file controls the access that applications have to the resources. This file is the standard Java SE policy file. In Web Server, the Java SE `SecurityManager` (the Java component that enforces the policy) is not active by default. The policies granted in this policy file do not have any effect unless the `SecurityManager` is turned on in `server.xml`.

To use the Java SE `SecurityManager`, turn it on by adding the following JVM options to `server.xml`, using the `jvm-options` subelement of the `jvm` element:

```
<jvm-options>-Djava.security.manager</jvm-options>
<jvm-options>-Djava.security.policy=instance_dir/config/server.policy</jvm-options>
```

You can also add JVM options using the Admin Console or the `wadm set -jvm-props` command.

Location

`instance_dir/config`

Syntax

```
grant [codeBase "path"] {
    permission permission_class "package", "permission_type";
    ...
};
```

See Also

[Sun Java System Web Server 7.0 Update 3 Developer's Guide to Java Web Applications](#)

<http://java.sun.com/docs/books/tutorial/security1.2/tour2/index.html>

default-web.xml

The `default-web.xml` is a global web deployment descriptor file that is shared by deployed web applications. There is one `default-web.xml` per server instance that is shared by all web applications deployed on the server instance.

Location

`instance_dir/config`

See Also

[Sun Java System Web Server 7.0 Update 3 Developer's Guide to Java Web Applications](#)

Using Variables, Expressions, and String Interpolation

This appendix describes variables, expressions, and string interpolation, and has the following sections:

- [“Variables” on page 277](#)
- [“Expressions” on page 281](#)
- [“String Interpolation” on page 296](#)

Variables

The Web Server includes a set of variables predefined by the server, as well as the capability for you to define custom variables. This section includes the following sections:

- [“Predefined Variables” on page 277](#)
- [“Custom Variables” on page 280](#)
- [“Resolving Variables” on page 280](#)

Predefined Variables

Predefined variables are implicitly defined by the server. The following table lists the predefined variables and their descriptions:

TABLE A-1 Predefined Variables

Variable	Description
<code>\$n</code>	Regular expression backreference (value of the <i>n</i> th capturing subpattern, <i>n</i> = 1...9), for example, \$1. Regular expression backreferences are only available within the body of <code>If</code> and <code>ElseIf</code> containers, and only if the container expressions includes one or more regular expressions. For more information on <code>If</code> and <code>ElseIf</code> , see “If, ElseIf, and Else” on page 126 .
<code>\$\$</code>	Value that matched a regular expression. Regular expression backreferences are only available within the body of <code>If</code> and <code>ElseIf</code> containers, and only if the container expressions includes one or more regular expressions. For more information on <code>If</code> and <code>ElseIf</code> , see “If, ElseIf, and Else” on page 126 .
<code>\$auth_group</code>	Authenticated user's group (alias for <code>\$vars{ 'auth-group' }</code>).
<code>\$auth_type</code>	Authentication method (alias for <code>\$vars{ 'auth-type' }</code>).
<code>\$auth_user</code>	Authenticated user name (alias for <code>\$vars{ 'auth-user' }</code>).
<code>\$browser</code>	Web browser version (alias for <code>\$headers{ 'user-agent' }</code> if the client sent a <code>User-Agent</code> header or an empty string).
<code>\$chunked</code>	Boolean variable that indicates whether request body was sent using chunked encoding.
<code>\$code</code>	Response status code.
<code>\$cookie{ 'name' }</code>	Value of the cookie <i>name</i> from request.
<code>\$dns</code>	Alias for <code>\$client{ 'dns' }</code> .
<code>\$env{ 'name' }</code>	Value of the environment variable <i>name</i> (includes CGI/SHTML environment variables).
<code>\$headers{ 'name' }</code>	Value of <i>name</i> from <code>rq->headers</code> , that is, value of the request header name where <i>name</i> is a lowercase string.
<code>\$id</code>	Virtual server ID as specified by the <i>name</i> subelement of the <code>virtual-server</code> element in <code>server.xml</code> . For more information, see “virtual-server” on page 97 .
<code>\$internal</code>	Boolean that indicates whether request was internally generated.
<code>\$ip</code>	Alias for <code>\$client{ 'ip' }</code> .
<code>\$keep_alive</code>	Boolean that indicates whether the connection will be kept open.
<code>\$keysize</code>	Alias for <code>\$client{ 'keysize' }</code> .
<code>\$method</code>	Request method (alias for <code>\$reqpb{ 'method' }</code>).

TABLE A-1 Predefined Variables (Continued)

Variable	Description
\$path	Requested path (either URI, partial path, or file system path depending on stage). The predefined variable path is the value of path from <code>rq->vars</code> . If path isn't set in <code>rq->vars</code> (for example, if <code>NameTrans</code> hasn't completed), path gets the value of <code>ppath</code> from <code>rq->vars</code> .
\$path_info	Alias for <code>\$vars{'path-info'}</code> .
\$ppath	Alias for <code>\$vars{'ppath'}</code> .
\$protocol	Request protocol (alias for <code>\$reqpb{'protocol'}</code>).
\$query	Request query string (alias for <code>\$reqpb{'query'}</code>).
\$reason	Response reason phrase.
\$referer	Alias for <code>\$headers{'referer'}</code> .
<code>\$reqpb{'name'}</code>	Value of <code>name</code> from <code>rq->reqpb</code> .
\$restarted	Boolean that indicates whether request has been restarted.
\$secret_keysize	Alias for <code>\$client{'secret-keysize'}</code> .
\$server_url	Prefix for self-referencing URLs.
\$time	Time the request was received as the number of seconds since 00:00:00 UTC, January 1, 2006.
\$time_day	Day of the month when the request was received. Value can be from 01 to 31.
\$time_hour	Hours since midnight when the request was received. Value can be from 00 to 23.
\$time_min	Minutes after the hour when the request was received. Value can be from 00 to 59.
\$time_mon	Month of the year when the request was received. Value can be from 01 to 12.
\$time_sec	Seconds after the minute when the request was received. Value can be from 00 to 61.
\$time_wday	Day of the week when the request was received. Value can be from 0 to 6, where 0 corresponds to Sunday.
\$time_year	Four-digit year when the request was received.
\$type	Alias for <code>\$srvhdrs{'content-type'}</code> .
\$uri	URI of the requested resource (alias for <code>\$reqpb{'uri'}</code>).

TABLE A-1 Predefined Variables (Continued)

Variable	Description
<code>\$url</code>	URL of the requested resource.
<code>\$urlhost</code>	Host name to which the client connected.
<code>\$vars{'\$headers{'name'}}</code>	Value of <i>name</i> from <code>rq->vars</code> .
<code>\$security</code>	Boolean that indicates whether a secure transport was used.
<code>\$senthdrs</code>	Boolean that indicates whether response headers have been sent.
<code>\$srvhdrs{'\$headers{'name'}}</code>	Value of <i>name</i> from <code>rq->srvhdrs</code> , that is, value of response header name where <i>name</i> is a lowercase string.

Custom Variables

You can define custom variables in the `server.xml` file using the `variables` element. These variables can then be used in function parameters in `obj.conf` functions. You can also define variables at request time using the `set-variables` function in `obj.conf`.

For more information, see [“variable” on page 96](#) and [“set-variable” on page 252](#).

Note – Because predefined variables take precedence over custom variables, it is a best practice to use uppercase names for custom variables. Using uppercase avoids conflicts with the lowercase predefined variables, should the list of predefined variables be extended in the future.

Resolving Variables

The server uses the following order when attempting to resolve a `$variable`:

1. Predefined variables
2. Variables defined at request time using `set-variable` in `obj.conf`
3. Variables defined by the `virtual-server` element's `variable` subelement in `server.xml`
4. Variables defined by the `server` element's `variable` subelement in `server.xml`

When you define a `$variable` at request time, it is stored as a name-value pair in the `rq->vars` `pblock`. These variables are given a higher precedence than `server.xml` variables so that `server.xml` variables can be overridden at request time.

Expressions

Expressions allow you to dynamically construct SAF parameters and to select which SAFs to execute on a request-by-request basis. Expressions are constructed from literals, variables, functions, and operators. Use expressions in `If` and `ElseIf` tags (see “[If, ElseIf, and Else](#)” on page 126), in log format strings (see [Appendix C, “Using the Custom Log File Format”](#)), and SAF parameters (see “[String Interpolation](#)” on page 296).

This section contains the following sections:

- “[Expression Syntax](#)” on page 281
- “[Expression Results as Booleans](#)” on page 282
- “[Expression Literals](#)” on page 282
- “[Expression Variables](#)” on page 283
- “[Expression Operators](#)” on page 284
- “[Expression Functions](#)” on page 286
- “[Regular Expressions](#)” on page 295

Expression Syntax

The expression syntax is similar to the syntax used in Perl. Expressions are constructed from literals, variables, functions, and operators.

The following example shows an expression used in an `If` tag:

```
<If not $internal
    and $uri =~ "^/private/(.*)$"
    and $referer !~ "https://example.com/">
NameTrans fn="redirect"
    url="http://example.com/denied.jsp?file=$1"
</If>
```

This example expression checks to see if a request meets certain criteria, for example if it is an internal request. If it does not meet the criteria, the server redirects the request to a request denied URL.

The expression contains the following components:

- Literals – “`^/private/(.*)$`” and “`https://example.com/`”
- Variables – `$internal`, `$uri`, and `$referer`
- Operators – `not`, `and`, `==`, and `!~`

Expression Results as Booleans

In some circumstances, for example, after evaluating an `If` or `ElseIf` expression, the server must treat the result of an expression as a Boolean. The server uses the following rules when converting a numeric value to a Boolean:

- Numeric 0 (zero) evaluates to false
- All other numeric values evaluate to true

The server uses the following rules when converting a string to a Boolean:

- Zero-length strings evaluate to false
- The string 0 (zero) evaluates to false
- All other strings evaluate to true

Expression Literals

Expression literals are divided into string and numeric literals.

String Literals

A string literal is bracketed by either single quotes (`'`) or double quotes (`"`). When single quotes bracket a string literal, the value of the literal is the value within the quotes. When double quotes are used, any references to variables or expressions within the quotes are interpolated. For more information, see [“String Interpolation” on page 296](#).

The following expression examples show the use of single and double quotes.

```
# This expression evaluates to true
('foo' eq "foo")

# This expression evaluates to false
('foo' eq "bar")

# This expression evaluates to true
('foo' eq "f$(lc('O'))o")

# This expression may evaluate to true or false,
# depending on the value of the variable $foo
('$foo' eq "$foo")
```

To include an uninterpolated `$` character in a double quote string, use the `$$` or `\$` escape sequences.

When a double quote character appears within a literal bracketed by double quotes, it must be prefixed with a backslash. When a single backslash (`\`) character appears within a literal bracketed by double quotes, it must be prefixed with a backslash. When a single quote character appears within a literal bracketed by single quotes, it must be prefixed with a backslash.

The following examples show valid and invalid literals:

```
# The following are examples of valid literals
'this string literal is bracketed by single quotes'
"this string literal is bracketed by double quotes"
"the backslash, \, escapes characters in double quote string literals"
'it\'s easy to use strings literals'

# The following are examples of invalid literals
'it's important to escape quote characters'
"any \ characters in double quote string literals must be escaped"
```

Numeric Literals

A numeric literal can consist of decimal digits and an optional decimal point, a leading zero followed by octal digits, or a leading `0x` prefix followed by hexadecimal digits. Hexadecimal and octal numbers are automatically converted to decimal form.

The following examples show expressions that use numeric literals:

```
# The following expressions evaluate to true
(1 < 2)
(0x10 == "16")
(1 == 1.00)

# The following expressions evaluate to false
(1 > 2)
("0x10" == 16)
(1 != 1.00)
```

Expression Variables

Any `$variable` can be used as a variable in an expression. To mirror the `Client` tag syntax, the `$` prefix is optional for predefined variable names in expressions. For example, the following three portions of `obj.conf` are all semantically equivalent:

```
<If $uri = "*.html">
...
</If>

<If uri = "*.html">
...
</If>

<Client uri = "*.html">
...
</Client>
```

Any variable names you define must use the \$ prefix. For example, the following expression is invalid even if `somecustomvariable` is defined in a `server.xml` variable element:

```
<If somecustomvariable = "foo">
...
</If>
```

To make this expression valid, add the dollar sign prefix:

```
<If $somecustomvariable = "foo">
...
</If>
```

Expression Operators

The following table lists the operators that are used in expressions.

TABLE A-2 List of Expression Operators

Operator Symbol	Operator Name
!	C-style logical not
=	Wildcard pattern match
=~	Regular expression match
!~	Regular expression mismatch
+	Addition or unary plus
-	Subtraction or unary minus
.	String concatenation
defined	Value is defined
-d	Directory exists
-e	File or directory exists
-f	File exists
-l	Symbolic link exists
-r	File is readable
-s	File size
-U	URI maps to accessible file or directory

TABLE A-2 List of Expression Operators (Continued)

Operator Symbol	Operator Name
<	Numeric less than
<=	Numeric less than or equal to
>	Numeric greater than
>=	Numeric greater than or equal to
lt	String less than
le	String less than or equal to
gt	String greater than
ge	String greater than or equal to
==	Numeric equal
!=	Numeric not equal
eq	String equal
ne	String not equal
^	C-style exclusive or
&&	C-style logical and
	C-style logical or
not	Logical not
and	Logical and
or	Logical or
xor	Logical exclusive or

The following table lists the precedence of operators within expressions from highest to lowest precedence.

TABLE A-3 Operator Precedence

Symbol	Operands	Associativity	Description
(), []	0	Left to right	Parentheses
!, unary +, unary -	1	Right to left	Sign operators
=, ==, !=	2	Non-associative	Pattern matching operators
+, -, .	2	Non-associative	Additive operators

TABLE A-3 Operator Precedence (Continued)

Symbol	Operands	Associativity	Description
defined, -d, -f, -l, -r, -s, -U	1	Right to left	Named operators
<, lt, <=, le, >, gt, >=, ge	2	Non-associative	Relational operators
==, eq, !=, ne	2	Non-associative	Equality operators
^	2	Left to right	C-style exclusive or operator
&&	2	Left to right	C-style logical and operator
	2	Left to right	C-style logical or operator
not	1	Right to left	Logical not operator
and	2	Left to right	Logical and operator
or, xor	2	Left to right	Logical or operators

The numeric operators (<, <=, >, >=, ==, and !=) are intended to operate on numbers and not strings. To facilitate comparing numbers, dates, and timestamps, the numeric operators ignore any white space, colons, slashes, and commas in their arguments. Dashes after the first digit are also ignored.

Note – It is generally incorrect to use the numeric operators on non-numeric values.

For example, the following expression evaluates to true:

```
# The following expression evaluates to true because both
# "foo" and "bar" are numerically equivalent to 0
("foo" == "bar")
```

Expression Functions

Expression functions manipulate data for use in expressions. Expression functions are different from SAFs. While SAFs perform the actual work associated with an HTTP request, expression functions are used to select which SAFs run and what parameters to pass to the SAFs.

Some expression functions require one or more arguments. An expression function's argument list is bracketed by parentheses (()) and the individual arguments are separated by commas (,).

The individual expression functions are listed in the following sections:

- [“atime” on page 287](#)
- [“choose” on page 287](#)

- “ctime” on page 288
- “escape” on page 288
- “external” on page 289
- “httpdate” on page 290
- “lc” on page 291
- “length” on page 291
- “lookup” on page 292
- “mtime” on page 293
- “owner” on page 293
- “uc” on page 294
- “unescape” on page 294
- “uuid” on page 295

atime

The `atime` function returns the time of the last access for the specified file or directory.

Syntax

`atime(path)`

Arguments

The following table describes the argument for the expression function.

TABLE A-4 `atime` Argument

Argument	Description
<i>path</i>	The absolute path to the directory or file name for which you are requesting the last access

See Also

- “ctime” on page 288
- “mtime” on page 293

choose

The `choose` function parses pipe-separated values from *values* and returns one at random.

Syntax

`choose(values)`

Arguments

The following table describes the argument for the expression function.

TABLE A-5 choose Argument

Argument	Description
<i>values</i>	The list of values to choose from, separated by the pipe character ()

Example

The following `obj.conf` code demonstrates the use of `choose` to randomly select one of three images:

```
NameTrans fn="rewrite"
           from="/images/random"
           path="/images/${choose('iwsvi.jpg|0061.jpg|webservervii.jpg')}")"
```

ctime

The `ctime` function returns the time of the last status change for the specified file or directory.

Syntax

`ctime(path)`

Arguments

The following table describes the argument for the expression function.

TABLE A-6 ctime Argument

Argument	Description
<i>path</i>	The absolute path to the directory or file name for which you are requesting the last status change

See Also

- “`atime`” on page 287
- “`mtime`” on page 293

escape

The `escape` function encodes the URI using `util_uri_escape`, converting special octets to their %-encoded equivalents, and returns the result.

Syntax

`escape(uri)`

Arguments

The following table describes the argument for the expression function.

TABLE A-7 escape Argument

Argument	Description
<i>uri</i>	The URI that the expression function converts

See Also

[“unescape” on page 294](#)

external

The external function passes a value to an external rewriting program and returns the result.

Each invocation of `external` results in a single newline-terminated line being written to the external rewriting program's `stdin`. For each line of input, the program must produce a single line of output. When developing an external rewriting program, it is important to avoid buffering `stdout`. In Perl, for example, `$| = 1;` should be used to disable buffering. Because the server expects the external rewriting program to produce one line of output for each line of input, the server can hang if the external rewriting program buffers its output.

Syntax

`external(program, value)`

Arguments

The expression function has the following arguments.

TABLE A-8 external Arguments

Argument	Description
<i>program</i>	<p>The <i>program</i> argument is the file name of an external rewriting program. Because <i>program</i> is executed using the operating system's default shell (<code>/bin/sh</code> on Unix/Linux) or the command interpreter (<code>CMD.EXE</code> on Windows), <i>program</i> should be an absolute path or the name of a program in the operating system's <code>PATH</code>. The server starts the external rewriting program on demand. A given server process never executes more than one instance of the program at a time.</p> <p>Note – The server may start multiple instances of a given external rewriting program when the server is running in multiprocess mode.</p>

TABLE A-8 external Arguments (Continued)

Argument	Description
<i>value</i>	The value passed to the rewrite program.

Example

The following is an example of an external rewriting program `rewrite.pl`, used to change the prefix `/home/` to `/u/`:

```
#!/usr/bin/perl
$| = 1;
while (<STDIN>) {
    s|^/home/|/u/|;
    print $_;
}
```

In this example, the external expression function used to invoke `rewrite.pl` is as follows:

```
NameTrans fn="rewrite" path="$(external('rewrite.pl', $path))"
```

httpdate

The `httpdate` function returns an RFC 1123 date/time stamp for use in HTTP header fields such as `Expires`.

Syntax

```
httpdate(time)
```

Arguments

The following table describes the argument for the expression function.

TABLE A-9 httpdate Argument

Argument	Description
<i>time</i>	The time value

Example

The following `obj.conf` code could be used to set an `Expires` header that indicates a response is not cached for more than one day:

```
ObjectType fn="set-variable"
    insert-srvhdrs="$(httpdate($time + 86400))"
```

lc

The `lc` function converts all the US ASCII characters in the string to lowercase and returns the result.

syntax

`lc(string)`

Arguments

The following table describes the argument for the expression function.

TABLE A-10 `lc` Argument

Argument	Description
<i>string</i>	The string the expression function converts to lowercase

Example

The following `obj.conf` code can be used to redirect clients who erroneously used uppercase characters in the request URI to the equivalent lowercase URI:

```
<If code == 404 and not -e path and -e lc(path)>
Error fn="redirect" uri="${lc($uri)}"
</If>
```

See Also

[“uc” on page 294](#)

length

The `length` function returns the length of its argument, that is, a number representing the length of the string.

Syntax

`length(string)`

Arguments

The following table describes the argument for the expression function.

TABLE A-11 Length Argument

Argument	Description
<i>string</i>	The string for which the expression function computes the length.

Example

The following `obj.conf` code can be used to send a 404 Not found error to clients that request URIs longer than 255 bytes:

```
<If length($uri) > 255>
PathCheck fn="deny-existence"
</If>
```

lookup

The lookup function inspects a text file for a name-value pair with name *name* and returns the corresponding value. The name-value pairs in the file are separated by white space.

If the file does not contain a name-value pair with the specified name, this function returns the value of *defaultvalue*, if specified, or returns an empty string.

Syntax

```
lookup(filename, name, defaultvalue)
```

Arguments

The expression function has the following arguments:

TABLE A-12 Lookup Arguments

Argument	Description
<i>filename</i>	<i>filename</i> is the name of a text file that contains one name-value pair per line. <i>filename</i> can be an absolute path or a path relative to the server's <code>config</code> directory. Names and values are separated by white space. Lines beginning with <code>#</code> are ignored.
<i>name</i>	The name of the name-value pair for which the function looks in the text file.
<i>defaultvalue</i>	The value returned by the function if <i>filename</i> exists but does not contain a name-value pair with a name matching the value of <i>name</i> . If <i>defaultvalue</i> is not specified, it defaults to an empty string.

Example

The following example shows a text file called `urimap.conf` that could be used with the `lookup` function to map shortcut URIs to URIs:

```
# This file contains URI mappings for Web Server.
# Lines beginning with # are treated as comments.
# All other lines consist of a shortcut URI, whitespace, and canonical URI.
/webserver /software/products/web_srvr/home_web_srvr.html
/solaris   /software/solaris/
/java      /software/java/
```

Using the sample text file above, you could use the following lookup expression to implement shortcut URIs for commonly accessed resources:

```
<If lookup('urimap.conf', uri)>
NameTrans fn="redirect" url="$(lookup('urimap.conf', uri))"
</If>
```

mtime

The `mtime` function returns the time of the last data modification for the specified file or directory.

Syntax

`mtime(path)`

Arguments

The following table describes the argument for the expression function.

TABLE A-13 `mtime` Argument

Argument	Description
<i>path</i>	The absolute path to the directory or file name for which you are requesting the last data modification

See Also

- [“atime” on page 287](#)
- [“ctime” on page 288](#)

owner

The `owner` function returns the owner of a file.

Syntax

`owner(path)`

Arguments

The following table describes the argument for the expression function.

TABLE A-14 `owner` Argument

Argument	Description
<i>path</i>	The absolute path to the directory or file name for which you are requesting the last data modification

uc

The `uc` function converts all the US ASCII characters in string to uppercase and returns the result.

Syntax

`uc(string)`

Arguments

The following table describes the argument for the expression function.

TABLE A-15 `uc` Argument

Argument	Description
<i>string</i>	The string that the expression function converts to uppercase

See Also

[“lc” on page 291](#)

unescape

The `unescape` function decodes the URI using `util_uri_unescape`, converting %-encoded octets to their unencoded form, and returns the result.

Syntax

`unescape(uri)`

Arguments

The following table describes the argument for the expression function.

TABLE A-16 unescape Argument

Argument	Description
<i>uri</i>	The URI that the function converts

See Also

[“escape” on page 288](#)

uuid

The `uuid` function returns a UUID as a string. No two calls to `uuid` return the same UUID. Because they are guaranteed to be unique, UUIDs are useful for constructing client-specific cookie values.

Syntax

```
uuid()
```

Regular Expressions

The `If` and `ElseIf` expressions may evaluate regular expressions using the `==` and `!~` regular expression matching operators. These regular expressions use the Perl-compatible syntax implemented by Perl-compatible Regular Expressions (PCRE).

By default, regular expressions are case sensitive. The `(?i)` option flag can be added to the beginning of a regular expression to request case insensitivity. For example:

```
$uri == '^/[Ff][Ii][Ll][Ee][Nn][Aa][Mm][Ee]$'
```

```
$uri == '(?i)~/filename$'
```

When an `If` or `ElseIf` expression contains a regular expression, regular expression backreferences can appear within arguments in the container body. Regular expression backreferences are of the form `$n` where `n` is an integer between 1 and 9 corresponding to the capturing subpattern of the regular expression.

For example:

```
<If $path == '^(.*)\\.html|\\.htm$'>
NameTrans fn="rewrite" path="$1.shtml"
</If>
```

In the above example, two subpatterns are used in the `If` expression, so `$1` and `$2` can be used as backreferences. In the example, the value of the first capturing subpattern is used within a `NameTrans fn="rewrite"` parameter. The value of the second capturing subpattern is ignored.

An `If` or `ElseIf` expression can contain backreferences to earlier regular expressions in that same `If` or `ElseIf` expression.

For example:

```
<If "foo" =~ "(.*)" and $1 eq "foo">
# Any contained directives will be executed
# since $1 will evaluate to "foo"
...
</If>
```

The contents of the above `If` expression are executed, because the given `If` expression always evaluates to true.

However, `If` and `Elseif` expressions, and contained directives, can't contain backreferences to regular expressions in parent containers. For example, the following `obj.conf` entry is invalid:

```
<If $path =~ '(.)\.css'>
<If $browser = "*MSIE*">
# This example is invalid as $1 is not defined
AuthTrans fn="rewrite" path="$1-msie.css"
</If>
</If>
```

You can use `&` to obtain the value that last successfully matched a regular expression. Use the following `obj.conf` entry to redirect requests for HTML files to another server:

```
<If $path =~ '\.html$' or $path =~ '\.htm$' >
NameTrans fn="redirect" url="http://docs.example.com&"
</If>
```

String Interpolation

Strings that contain references to variables or expressions are called interpolated strings. When you use interpolated strings, the embedded expressions and variables are evaluated and the result is inserted into the string. The act of inserting data into a string is called string interpolation.

Use interpolated strings in expressions, log formats, and `obj.conf` parameters. In expressions, only string literals bracketed by double quotes are interpolated. For more information, see [“Expression Literals” on page 282](#).

Using Variables in Interpolated Strings

To include the value of a variable in a string, prefix the name of the variable with the dollar-sign (\$). For example, the following `format` element in `server.xml` logs the client IP address, requested URI, and corresponding file system path for each HTTP request:

```
<access-log>
  <file>access</file>
  <format>$ip "$uri" $path</format>
</access-log>
```

In this example, `$ip`, `$uri`, and `$path` are predefined variables. For more information, see [“Variables” on page 277](#).

For more information on access logs and log format, see [Appendix C, “Using the Custom Log File Format.”](#) For more information on the `access-log` element in `server.xml`, see [“access-log” on page 43](#).

If the name of the variable is ambiguous, add curly braces, `{}`, to the name. For example, the following string contains a reference to the predefined `$path` variable:

```
"${path}html"
```

Without the curly braces, the string instead contains a reference to a hypothetical variable named `pathhtml`.

Using Expressions in Interpolated Strings

To include the result of an expression in a string, prefix the expression with `$` (and follow it with `.`). For example, the following two strings are identical after interpolation:

```
"$(2 + 2)"
"4"
```

When an interpolated string is used as an `obj.conf` parameter, the string is interpolated each time the corresponding instruction is executed. For example, the following lines could be used in `obj.conf` to redirect clients based on the requested URI and the contents of the file `redirect.conf`:

```
<Object ppath="/redirect/*">
NameTrans fn="redirect" url="$(lookup('redirect.conf', $uri, '/'))"
</Object>
```

In this example, the expression `lookup('redirect.conf', $uri, '/')` is evaluated each time the `NameTrans` directive is invoked, and the result is passed to the `redirect` SAF in its `url`

parameter. For more information on the `redirect` SAF, see [“redirect” on page 246](#). For more information on the `lookup` expression function, see [“lookup” on page 292](#).

Using Wildcard Patterns

This appendix describes the wildcard patterns used by the Sun Java System Web Server. Wildcards use special characters and are applicable in the `obj.conf` file, built-in SAFs, and NSAPI functions. To use a wildcard character without any special meaning, precede it with a backslash (`\`) character.

This appendix has the following sections

- “Wildcard Patterns” on page 299
- “Wildcard Examples” on page 300

Wildcard Patterns

The following table describes wildcard patterns, listing the patterns and their uses.

TABLE B-1 Wildcard Patterns

Pattern	Use
*	Match zero or more characters.
?	Match exactly one occurrence of any character.
	An or expression. The substrings used with this operator can contain other special characters such as * or \$. The substrings must be enclosed in parentheses, for example, (a b c), but the parentheses cannot be nested.
\$	Match the end of the string. This is useful in or expressions.
[abc]	Match one occurrence of the characters a, b, or c. Within these expressions, the only character that needs to be treated as a special character is] ; all others are not special.

TABLE B-1 Wildcard Patterns (Continued)

Pattern	Use
[a-z]	Match one occurrence of a character between a and z.
[^az]	Match any character except a or z.
*~	This expression, followed by another expression, removes any pattern matching the second expression.

Wildcard Examples

The following table provides wildcard examples, listing the pattern and the result.

TABLE B-2 Wildcard Examples

Pattern	Result
*.sun.com	Matches any string ending with the characters .sun.com.
(quark energy).sun.com	Matches either quark.sun.com or energy.sun.com.
198.93.9[23].???	Matches a numeric string starting with either 198.93.92 or 198.93.93 and ending with any 3 characters.
.	Matches any string with a period in it.
~sun-	Matches any string except those starting with sun-.
*.sun.com~quark.sun.com	Matches any host from domain sun.com except for a single host quark.sun.com.
*.sun.com~(quark energy neutrino).sun.com	Matches any host from domain .sun.com except for hosts quark.sun.com, energy.sun.com, and neutrino.sun.com.
.com~.sun.com	Matches any host from domain .com except for hosts from sub-domain sun.com.
type=~magnus-internal/*	Matches any type that does not start with magnus-internal/. This wildcard pattern is used in the file obj.conf in the catch-all Service directive.
~.gif*	Matches any string except those including gif.

Using the Custom Log File Format

This chapter contains information about the log format used by Web Server. Use these format options to customize the format of your log files. You can enter them through the Admin Console, or edit the format subelement of the `access-log` element in `server.xml`. For more information, see [“access-log” on page 43](#).

You can use variables and expressions in log formats with the syntax `$variable` and `$(expression)`. For more information, see [“Variables” on page 277](#), and [“Expressions” on page 281](#).

Custom Log File Format

When creating a custom log format, anything contained between percent signs (%) is recognized as the name portion of a name-value pair stored in a parameter block in the server. Any additional text is treated as literal text, so you can add to the line to make it more readable. The one exception to the percent sign rule is the `%SYSDATE%` component, which delivers the current system date. `%SYSDATE%` is formatted using the time format `%d/%b/%Y:%H:%M:%S` plus the offset from GMT.

If no format parameter is specified for a log file, the common log format is used:

```
"%Ses->client.ip% - %Req->vars.auth-user% [%SYSDATE%]  
\ "%Req->reqpb.clf-request%\ " %Req->srvhdrs.clf-status%  
%Req->srvhdrs.content-length%"
```

Typical components of log file format are listed in the following table. Because certain components could resolve to values that contain spaces, they are enclosed in escaped quotes (\").

TABLE C-1 Typical Components of Custom Log Formatting

Option	Component
Client host name (unless <code>iponly</code> is specified in <code>flex-log</code> or DNS name is not available) or IP address	<code>%Ses->client.ip%</code>
Client DNS name	<code>%Ses->client.dns%</code>
System date	<code>%SYSDATE%</code>
Full HTTP request line	<code>\ "%Req->reqpb.clf-request%\ "</code>
Status	<code>%Req->srvhdrs.clf-status%</code>
Response content length	<code>%Req->srvhdrs.content-length%</code>
Response content type	<code>%Req->srvhdrs.content-type%</code>
Referer header	<code>\ "%Req->headers.referer%\ "</code>
User-Agent header	<code>\ "%Req->headers.user-agent%\ "</code>
HTTP method	<code>%Req->reqpb.method%</code>
HTTP URI	<code>%Req->reqpb.uri%</code>
HTTP query string	<code>%Req->reqpb.query%</code>
HTTP protocol version	<code>%Req->reqpb.protocol%</code>
Accept header	<code>%Req->headers.accept%</code>
Date header	<code>%Req->headers.date%</code>
If-Modified-Since header	<code>%Req->headers.if-modified-since%</code>
Authorization header	<code>%Req->headers.authorization%</code>
Any header value	<code>%Req->headers.headername%</code>
Name of authorized user	<code>%Req->vars.auth-user%</code>
Value of a cookie	<code>%Req->headers.cookie.name%</code>
Value of any variable in <code>Req->vars</code>	<code>%Req->vars.varname%</code>
Virtual server ID	<code>%vsid%</code>
Duration	<code>%duration%</code> Records the time in microseconds the server spent handling the request. Statistics must be enabled before <code>%duration%</code> can be used.

TABLE C-1 Typical Components of Custom Log Formatting (Continued)

Option	Component
System time	%Time System time in seconds since 00:00:00 UTC, January 1, 1970.
Relative time	%RELATIVETIME% System time in seconds since logging started.
Method number	%Req->method_num% A number representing the HTTP method as used in NSAPI.
HTTP Protocol Version	%Req->protv_num% A number representing the HTTP protocol version as used in NSAPI.
HTTP request line	%Req->reqpb.clf-request.method% The method from the HTTP request line.
HTTP URI	%Req->reqpb.clf-request.uri% The URI from the HTTP request line.
URI Path	%Req->reqpb.clf-request.uri.abs_path% The absolute path component of the URI
URI Query	%Req->reqpb.clf-request.uri.query% The query component of the URI.
user_dn	%Ses->client.user_dn% The SSL client certificate authentication for web security.
HTTP Protocol	%Req->reqpb.clf-request.protocol% The protocol from the HTTP request line.
Protocol Name	%Req->reqpb.clf-request.protocol.name% The name of the protocol.
Protocol Version	%Req->reqpb.clf-request.protocol.version% The version of the protocol.

Using Time Formats

This appendix 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`). For more information about server-parsed HTML, see *Sun Java System Web Server 7.0 Update 3 NSAPI Developer's Guide*.

The formats are similar to those used by the `strftime` C library routine, but not identical. For more information on the NSAPI function, `util_strftime`, see “[util_strftime Function](#)” in *Sun Java System Web Server 7.0 Update 3 NSAPI Developer's Guide*.

Format Strings

The following table describes the format strings for dates and times.

TABLE D-1 Format Strings for Date and Time

Attribute	Allowed Values
%a	Abbreviated day of the week (3 chars)
%d	Day of month as decimal number (01-31)
%S	Second as decimal number (00-59)
%M	Minute as decimal number (00-59)
%H	Hour in 24-hour format (00-23)
%Y	Year with century, as decimal number, up to 2099
%b	Abbreviated month name (3 chars)
%h	Abbreviated month name (3 chars)

TABLE D-1 Format Strings for Date and Time (Continued)

Attribute	Allowed Values
%T	Time in "HH:MM:SS" format
%X	Time in "HH:MM:SS" format
%A	Day of the week, full name
%B	Month, full name
%C	"%a %b %e %H:%M:%S %Y"
%c	Date and time in "%m/%d/%y %H:%M:%S" format
%D	Date in "%m/%d/%y" format
%e	Day of month as decimal number (1-31) without leading zeros
%I	Hour in 12-hour format (01-12)
%j	Day of year as decimal number (001-366)
%k	Hour in 24-hour format (0-23) without leading zeros
%l	Hour in 12-hour format (1-12) without leading zeros
%m	Month as decimal number (01-12)
%n	Line feed
%p	a.m./p.m. indicator for 12-hour clock
%R	Time in "%H:%M" format
%r	Time in "%I:%M:%S %p" format
%t	Tab
%U	Week of year as decimal number, with Sunday as first day of week (00-51)
%w	Weekday as decimal number (0-6; Sunday is 0)
%W	Week of year as decimal number, with Monday as first day of week (00-51)
%x	Date in "%m/%d/%y" format
%y	Year without century, as decimal number (00-99)
%%	Percent sign

Configuration Changes Between Sun ONE Web Server 6.1 and Sun Java System Web Server 7.0

This appendix summarizes the major configuration file changes between the 6.1 and the 7.0 version of Sun Java System Web Server.

- “Element Changes in `server.xml`” on page 307
- “Directive and Init Function Changes in `magnus.conf`” on page 309
- “Other Configuration File Changes” on page 314

Element Changes in `server.xml`

This section summarizes the changes in `server.xml`.

TABLE E-1 `server.xml` Changes

Web Server 6.1	Web Server 7.0	Description
SERVER	Replaced	Replaced by <code>server</code> and <code>qos</code> . For more information, see “ <code>server</code> ” on page 80 and “ <code>qos</code> ” on page 77.
LS	Replaced	Replaced by <code>http-listener</code> . For more information, see “ <code>http-listener</code> ” on page 63.
SSLPARAMS	Replaced	Replaced by <code>ssl</code> . For more information, see “ <code>ssl</code> ” on page 88.
MIME	Replaced	Replaced by <code>mime-file</code> . For more information, see “ <code>mime-file</code> ” on page 74.
ACLFIL	Replaced	Replaced by <code>acl-file</code> . For more information, see “ <code>acl-file</code> ” on page 46.
VSCCLASS	Replaced	Replaced by <code>localization</code> and <code>object-file</code> . For more information, see “ <code>localization</code> ” on page 70 and <code>object-file</code> .

TABLE E-1 server.xml Changes (Continued)

Web Server 6.1	Web Server 7.0	Description
VS	Replaced	Replaced by <code>virtual-server</code> . The <code>virtual-server</code> element includes subelements such as <code>host</code> , <code>http-listener-name</code> , <code>acl-file</code> , <code>mime-file</code> , <code>object-file</code> , <code>default-object-name</code> , and <code>log-file</code> . For more information on <code>virtual-server</code> and its subelements, see “ virtual-server ” on page 97.
QOSPARAMS	Replaced	Replaced by <code>qos-limits</code> . For more information, see “ qos-limits ” on page 77.
USERDB	Replaced	Replaced by <code>auth-db</code> . For more information, see “ auth-db ” on page 47.
DAV	Replaced	Replaced by <code>dav</code> . For more information, see “ dav ” on page 53.
DAVCOLLECTION	Replaced	Replaced by <code>dav-collection</code> . For more information, see “ dav-collection ” on page 54.
SEARCH	Replaced	Replaced by <code>search-app</code> . For more information, see “ search-app ” on page 79.
SEARCHCOLLECTION	Replaced	Replaced by <code>search-collection</code> . For more information, see “ search-collection ” on page 79.
WEBAPP	Replaced	Replaced by <code>web-app</code> . For more information, see “ web-app ” on page 99.
JAVA	Replaced	Replaced by <code>jvm</code> and <code>servlet-container</code> . For more information, see “ jvm ” on page 67 and “ servlet-container ” on page 83.
JVMOPTIONS	Replaced	Replaced by <code>jvm-options</code> . For more information, see “ jvm ” on page 67.
PROFILER	Replaced	Replaced by <code>profiler</code> . For more information, see “ profiler ” on page 74.
SECURITY	Replaced	Replaced by <code>security</code> .
AUTHREALM	Replaced	Replaced by <code>auth-realm</code> . For more information, see “ auth-realm ” on page 48.
RESOURCES	Replaced	Replaced by <code>resources</code> .
CUSTOMRESOURCE	Replaced	Replaced by <code>custom-resource</code> . For more information, see “ custom-resource ” on page 52.

TABLE E-1 server.xml Changes (Continued)

Web Server 6.1	Web Server 7.0	Description
EXTERNALJNDIRESOURCE	Replaced	Replaced by <code>external-jndi-resource</code> . For more information, see “external-jndi-resource” on page 59 .
JDBCRESOURCE	Replaced	Replaced by <code>jdbc-resource</code> . For more information, see “jdbc-resource” on page 66 .
JDBCPOOL	Replaced	Replaced by <code>jdbc-resource</code> . For more information, see “jdbc-resource” on page 66 .
MAILRESOURCE	Replaced	Replaced by <code>mail-resource</code> . For more information, see “mail-resource” on page 73 .
LOG	Replaced	Replaced by <code>log</code> . For more information, see “log” on page 71 .
DESCRIPTION	Replaced	Replaced by <code>description</code> .
DISPLAYNAME	Replaced	Replaced by <code>display-name</code> . For more information, see “display-name” on page 56 .
VARS	Replaced	Replaced by <code>variable</code> . For more information, see “variable” on page 96 .
PROPERTY	Replaced	Replaced by <code>variable</code> and <code>property</code> . For more information, see “variable” on page 96 and “property” on page 75 .

Directive and Init Function Changes in magnus.conf

This section summarizes the changes in `magnus.conf`.

- [“Directive Changes” on page 309](#)
- [“Init Function Changes” on page 313](#)

Directive Changes

The following table summarizes the changes made to `magnus.conf` directives:

TABLE E-2 Directive Changes in magnus.conf

Web Server 6.1	Web Server 7.0	Description
MaxProcs	Deprecated for Java technology-enabled servers	Configures multiprocess mode. Multiprocess mode is deprecated for Java technology-enabled servers.
PidLog	Removed	The pid file is now named pid and stored in the server's temporary directory.
TempDir	Replaced	Replaced by the server.xml temp-path element.
TempDirSecurity	Removed	
DNS	Subsumed	Subsumed by the server.xml dns element. For more information, see “dns” on page 56.
AsyncDNS	Subsumed	Subsumed by the server.xml dns element. For more information, see “dns” on page 56.
HTTPVersion	Subsumed	Subsumed by the server.xml dns element. For more information, see “dns” on page 56.
ServerString	Subsumed	Subsumed by the server.xml http element. For more information, see “http” on page 61.
AcceptTimeout	Subsumed	Subsumed by the server.xml http element. For more information, see “http” on page 61.
Favicon	Subsumed	Subsumed by the server.xml http element. For more information, see “http” on page 61.
HeaderBufferSize	Subsumed	Subsumed by the server.xml http element. For more information, see “http” on page 61.
MaxRqHeaders	Subsumed	Subsumed by the server.xml http element. For more information, see “http” on page 61.
StrictHttpHeaders	Subsumed	Subsumed by the server.xml http element. For more information, see “http” on page 61.
UseOutputStreamSize	Subsumed	Subsumed by the server.xml http element. For more information, see “http” on page 61.
ChunkedRequestBufferSize	Subsumed	Subsumed by the server.xml http element. For more information, see “http” on page 61.
ChunkedRequestTimeout	Subsumed	Subsumed by the server.xml http element. For more information, see “http” on page 61.
ConnQueueSize	Subsumed	Subsumed by the server.xml thread-pool element. For more information, see “thread-pool” on page 95.

Web Server 6.1	Web Server 7.0	Description
RqThrottle	Subsumed	Subsumed by the server.xml thread-pool element. For more information, see “thread-pool” on page 95 .
RqThrottleMin	Subsumed	Subsumed by the server.xml thread-pool element. For more information, see “thread-pool” on page 95 .
StackSize	Subsumed	Subsumed by the server.xml thread-pool element. For more information, see “thread-pool” on page 95 .
KeepAliveQueryMeanTime	Subsumed	Subsumed by the server.xml keep-alive element. For more information, see “keep-alive” on page 69 .
KeepAliveQueryMaxSleepTime	Removed	Keep-alive connection management changes render this directive obsolete.
KeepAliveTimeout	Subsumed	Subsumed by the server.xml keep-alive element. For more information, see “keep-alive” on page 69 .
MaxKeepAliveConnections	Subsumed	Subsumed by the server.xml keep-alive element. For more information, see “keep-alive” on page 69 .
KeepAliveThreads	Subsumed	Subsumed by the server.xml keep-alive element. For more information, see “keep-alive” on page 69 .
Security	Subsumed	Subsumed by the server.xml pkcs11 element. For more information, see “pkcs11” on page 74 .
SSLClientAuthDataLimit	Subsumed	Subsumed by the server.xml ssl element. For more information, see “ssl” on page 88 .
SSLClientAuthTimeout	Subsumed	Subsumed by the server.xml ssl element. For more information, see “ssl” on page 88 .
SSLCacheEntries	Subsumed	Subsumed by the server.xml ssl-session-cache element. For more information, see “ssl-session-cache” on page 93 .
SSLSessionTimeout	Subsumed	Subsumed by the server.xml ssl-session-cache element. For more information, see “ssl-session-cache” on page 93 .
SSL3SessionTimeout	Subsumed	Subsumed by the server.xml ssl-session-cache element. For more information, see “ssl-session-cache” on page 93 .
ACLCacheLifetime	Subsumed	Subsumed by the server.xml acl-cache element. For more information, see “acl-cache” on page 44 .
ACLUserCacheSize	Subsumed	Subsumed by the server.xml acl-cache element. For more information, see “acl-cache” on page 44 .

TABLE E-2 Directive Changes in magnus.conf		(Continued)
Web Server 6.1	Web Server 7.0	Description
ACLGroupCacheSize	Subsumed	Subsumed by the <code>server.xml</code> <code>acl-cache</code> element. For more information, see “acl-cache” on page 44 .
CGIExpirationTimeout	Subsumed	Subsumed by the <code>server.xml</code> <code>cgi</code> element. For more information, see “cgi” on page 49 .
CGIStubIdleTimeout	Subsumed	Subsumed by the <code>server.xml</code> <code>cgi</code> element. For more information, see “cgi” on page 49 .
MinCGIStubs	Subsumed	Subsumed by the <code>server.xml</code> <code>cgi</code> element. For more information, see “cgi” on page 49 .
MaxCGIStubs	Subsumed	Subsumed by the <code>server.xml</code> <code>cgi</code> element. For more information, see “cgi” on page 49 .
WinCGITimeout	Subsumed	Subsumed by the <code>server.xml</code> <code>cgi</code> element. For more information, see “cgi” on page 49 .
CGIWaitPid	Removed	Controlled whether the CGI subsystem uses <code>wait</code> or <code>waitpid</code> to reap child processes. The CGI subsystem will now always use <code>waitpid</code> .
ErrorLogDateFormat	Subsumed	Subsumed by the <code>server.xml</code> <code>log</code> element. For more information, see “log” on page 71 .
ListenQ	Subsumed	Subsumed by the <code>server.xml</code> <code>http-listener</code> element. For more information, see “http-listener” on page 63 .
RcvBufSize	Subsumed	Subsumed by the <code>server.xml</code> <code>http-listener</code> element. For more information, see “http-listener” on page 63 .
SndBufSize	Subsumed	Subsumed by the <code>server.xml</code> <code>http-listener</code> element. For more information, see “http-listener” on page 63 .
LogFlushInterval	Subsumed	Subsumed by the <code>server.xml</code> <code>access-log-buffer</code> element. For more information, see “access-log-buffer” on page 43 .
DefaultLanguage	Subsumed	Subsumed by the <code>server.xml</code> <code>localization</code> element. For more information, see “localization” on page 70 .
ExtraPath	Removed	Server startup/configuration changes render this directive obsolete.
PostThreadsEarly	Removed	Thread management changes render this directive obsolete.
ThreadIncrement	Removed	Thread management changes render this directive obsolete.

TABLE E-2 Directive Changes in magnus.conf (Continued)

Web Server 6.1	Web Server 7.0	Description
UseNativePoll	Removed	Native poll implementation versus NSPR implementation will always be used.
AdminLanguage	Removed	AdminLanguage was deprecated in a previous release.
ClientLanguage	Removed	ClientLanguage was deprecated in a previous release.
NetsiteRoot	Removed	NetsiteRoot was deprecated in a previous release.
ServerID	Removed	ServerID was deprecated in a previous release.
ServerName	Removed	ServerName was deprecated in a previous release.
ServerRoot	Removed	ServerRoot was deprecated in a previous release.

Init Function Changes

The following table summarizes the changes made to magnus.conf Init functions.

TABLE E-3 Init function changes in magnus.conf

Web Server 6.1	Web Server 7.0	Description
dns-cache-init	Deprecated	Superseded by the server.xml dns-cache element. For more information, see “dns-cache” on page 57 .
flex-init	Deprecated	Superseded by the server.xml access-log element. For more information, see “access-log” on page 43 .
perf-init	Deprecated	Superseded by the server.xml stats element. For more information, see “stats” on page 94 .
stats-init	Deprecated	Superseded by the server.xml stats element. For more information, see “stats” on page 94 .
init-cgi	Deprecated	Superseded by the server.xml cgi element. For more information, see “cgi” on page 49 .
init-clf	Deprecated	Superseded by the server.xml access-log element. For more information, see “access-log” on page 43 .
nt-console-init	Deprecated	Superseded by the server.xml log element. For more information, see “log” on page 71 .
flex-rotate-init	Deprecated	Superseded by the server.xml event and log elements. For more information, see “event” on page 58 and “log” on page 71 .

Other Configuration File Changes

This section lists additional configuration file changes in Sun Java System Web Server 7.0.

The following files have been removed and are no longer applicable:

- `dbswitch.conf` - This file configured authentication databases. The functionality of this file is subsumed by the `server.xml auth-db` element. For more information, see [“auth-db” on page 47](#).
- `nsfc.conf` - This optional file configured the Netscape file cache. The functionality of this file is subsumed by the `server.xml file-cache` element. For more information, see [“file-cache” on page 60](#).
- `password.conf` - This optional file containing one or more PKCS #11 PINs allowed unattended restarts of an SSL-enabled server. The functionality of this file is subsumed by the `server.xml pkcs11` element. For more information, see [“pkcs11” on page 74](#).
- `*.clfilter` - The `magnus.conf.clfilter`, `obj.conf.clfilter`, and `server.xml.clfilter` files defined the program used to filter node-specific information from configuration files when propagating configuration changes across a cluster. This filtering is now performed automatically by the Administration Server.

The location of the following files has changed:

- `certmap.conf` - This file has been moved from the `install_dir/userdb` directory to the `instance_dir/config` directory.
- `https-server_id-hostname-cert8.db` - This file has been moved from the `install_dir/alias` directory to the `instance_dir/config` directory and is renamed `cert8.db`.
- `https-server_id-hostname-key3.db` - This file has been moved from the `install_dir/alias` directory to the `instance_dir/config` directory and is renamed `key3.db`.
- `secmod.db` - This file has been moved from the `install_dir/alias` directory to the `instance_dir/config` directory.
- `generated.https-server_id.acl` - This file has been moved from the `install_dir/httpacl` directory to the `instance_dir/config` directory and is renamed `default.acl`.

Web Server Interfaces

This appendix describes the interfaces in Web Server and their stability level. Sun products classify public interfaces according to their expected stability level so that you can make informed decisions when creating dependencies on these interfaces. For example, you can confidently create programmatic dependencies (for example, shell scripts) which rely on stable interfaces, knowing these will not change often (if ever).

Note that the word interface is used in a very broad sense. Any implementation detail on which your code might rely on can be an interface. This includes APIs but also includes aspects such as CLI option names, file system paths, file names and so forth.

In the following table, the stability levels have the following definitions:

- Standard – Interfaces defined by a standard, for example Java Servlet API (JSR 154). Changes track the standard specification and are as stable as the referenced standard.
- Stable – Incompatibilities are exceptional. Incompatible changes can only occur in the next major release and with prior warning. While possible, incompatible changes to stable interfaces are not expected.
- Evolving – Incompatibilities are rare. Incompatible change can only occur in the next minor release and with prior warning.
- Unstable – Experimental or transitional: incompatibilities are common. While future release of the Web Server might attempt to provide either stability or a migration path for unstable interfaces, incompatible changes are possible at any time. If at all possible, avoid creating programmatic dependencies on unstable interfaces or your code might break in a future release. If you need to create programmatic dependencies on unstable interfaces, structure your code in a way which makes it easy to adapt to future changes.
- Obsolete – Obsolete interfaces continue to be supported but might be removed in some (not yet determined) future release. Do not create any new dependencies on obsolete interfaces. If you have existing dependencies on obsolete interfaces, remove those dependencies as soon as possible.

- Private – Private interfaces cannot be relied on for any use. Private interfaces might change incompatibly (or disappear entirely) without prior notice at any time. Sun cannot provide any support or assurance of any use of private interfaces.

Note – Private interfaces are for the most part not listed in this appendix, because all interfaces not documented in the product documentation are by default private. However, some visible but private interfaces are explicitly documented as private in this appendix to highlight the fact that these interfaces cannot be used.

TABLE F-1 Interfaces

Interface Name	Stability Level	Comments
server.xml	Unstable	Avoid creating scripts which read or write to server.xml directly. Instead, use the wadm CLI to reliably modify server.xml.
magnus.conf	Evolving	Where possible, use the wadm CLI to reliably modify configuration files.
default.acl	Evolving	Where possible, use the wadm CLI to reliably modify configuration files.
certmap.conf	Evolving	Where possible, use the wadm CLI to reliably modify configuration files.
obj.conf	Evolving	Where possible, use the wadm CLI to reliably modify configuration files.
mime.types	Evolving	Where possible, use the wadm CLI to reliably modify configuration files.
server.policy	Evolving	Where possible, use the wadm CLI to reliably modify configuration files.
login.conf	Evolving	Where possible, use the wadm CLI to reliably modify configuration files.
Any configuration files not specifically listed above	Private	Naming and contents of any other configuration files are not intended for user manipulation.

TABLE F-1 Interfaces (Continued)

Interface Name	Stability Level	Comments
\$PKGROOT/bin/	Stable	The location of supported public binaries.
\$PKGROOT/include/	Stable	The location of public include files for developers.
\$PKGROOT/plugins/	Stable	The location of documented plug-ins.
\$PKGROOT/samples/	Unstable	Samples are a form of documentation. They are provided for reference, but might change from release to release. Do not build hard dependencies on samples.
\$PKGROOT/lib/	Private	No external use supported.
<i>instance_dir</i> /bin/*	Stable	The location of supported, public, instance-specific binaries: <i>startserv</i> , <i>stopserv</i> , <i>rotate</i> , <i>restart</i> , and <i>reconfig</i> .
<i>instance_dir</i> //logs/access	Location: stable Content: stable	The access log file can be parsed by scripts.
<i>instance_dir</i> //logs/errors	Location: stable Content: not an Interface	The content of the log, for example, the wording of messages, is not an interface suitable for programmatic access and might change from patch to patch. It is intended for visual parsing by human readers only.
Installer CLI and options	Evolving	
Uninstall CLI and options	Evolving	
Silent installer statefile variables	Evolving	
Installer graphical user interface (GUI)	Unstable	GUI screen layouts are generally unstable.
Installer exit values	Evolving	
Configurator/unconfigurator back-end CLIs	Private	No external use supported.
Configurator back-end properties	Private	No external use supported.
Jacl	Private	No external use supported.

TABLE F-1 Interfaces (Continued)

Interface Name	Stability Level	Comments
JLine	Private	No external use supported.
PCRE	Private	No external use supported.
Xalan C++	Private	No external use supported.
Xerces C++	Private	No external use supported.
schema2beans	Private	No external use supported.
Admin Console	Unstable	GUI screen layouts are generally unstable.
wadm CLI and command-line arguments	Evolving	Where possible, use the wadm CLI to reliably modify configuration files.
wadm CLI error codes	Evolving	Where possible, use the wadm CLI to reliably modify configuration files.
wadm password file format	Evolving	Where possible, use the wadm CLI to reliably modify configuration files.
wadm output (stdout and stderr)	Not an interface	Output generated by the CLI only provides messages for a human reader. It is not intended for programmatic parsing or scripting.
.wadmrc file	Evolving	Optional Jacl file residing in the user's home directory or loaded up by <code>--rcfile</code> . It serves as a startup file.
wdeploy CLI	Obsolete	Previously obsoleted, still retained. Replaced by wadm. Will be removed in a future release.
SNMP MIB	Evolving	
JES-MF MBeans	Private	No external use supported.
com.sun.appserv.server.Lifecycle APIs	Evolving	API details in lifecycle spec.
JSR 88 implementation	Evolving	
SUNWwbsvr7 (Solaris) sun-websvr-7.0.0-1.i386.rpm (Linux rpm)	Stable	Main Web Server 7.0 package. The name will change in the next major release.

TABLE F-1 Interfaces (Continued)

Interface Name	Stability Level	Comments
SUNWwbsvr7-dev (Solaris) sun-webserver-dev-7.0.0-1.i386.rpm (Linux rpm)	Stable	Package that contains additional files for developer support (for example, header files). The name will change in the next major release.
SUNWwbsvr7-cli (Solaris) sun-webserver-cli-7.0.0-1.i386.rpm (Linux rpm)	Stable	The CLI package. The CLI can be installed separately on other hosts. The name will change in the next major release.
N1plugin-descriptor.xml	Private	No external use supported.
N1pluginUI.xml	Private	No external use supported.
TCP port 8989	Stable	Default administration HTTP SSL port. IANA registration completed.
TCP port 8800	Stable	Default administration HTTP non-SSL port. IANA registration completed.
WebDAV	Standard	RFC 2518, RFC 3744.
JSTL 1.1	Standard	
MaxProcs mode	Obsolete	MaxProcs mode is obsolete but still supported. It might be removed in a future release.

Alphabetical List of Server Configuration Elements and Predefined SAFs

This appendix provides an alphabetical list of server configuration elements, including server .xml elements, and predefined SAFs in magnus.conf and obj.conf files.

A

- “access-log” on page 43
- “access-log-buffer” on page 43
- “acl-cache” on page 44
- “acl-file” on page 46
- “acl-db” on page 45
- “add-footer” on page 205
- “add-header” on page 206
- “append-trailer” on page 208
- “assign-name” on page 143
- “audit-accesses” on page 46
- “auth” on page 47
- “auth-db” on page 47
- “auth-realm” on page 48

B

- “basic-auth” on page 138
- “basic-ncsa” on page 140
- “block-auth-cert” on page 171

“block-cache-info” on page 172

“block-cipher” on page 172

“block-ip” on page 173

“block-issuer-dn” on page 173

“block-jroute” on page 174

“block-keysize” on page 174

“block-proxy-agent” on page 175

“block-proxy-auth” on page 176

“block-secret-keysize” on page 176

“block-ssl-id” on page 177

“block-user-dn” on page 177

“block-via” on page 178

C

“cgi” on page 49

“check-acl” on page 154

“check-request-limits” on page 155

“cindex-init” on page 108

“cluster” on page 50

“compress-file” on page 196

“connection-creation-property” on page 50

“connection-lease-property” on page 51

“convert” on page 52

“custom-resource” on page 52

D

“dav” on page 53

“dav-collection” on page 54

“default-auth-db-name” on page 55

“default-auth-realm-name” on page 56
“default-soap-auth-provider-name” on page 56
“define-perf-bucket” on page 109
“delete-file” on page 209
“deny-existence” on page 157
“display-name” on page 56
“dns” on page 56
“dns-cache” on page 57
“document-root” on page 144

E

“env-variable” on page 58
“error-j2ee” on page 240
“event” on page 58
“external-jndi-resource” on page 59

F

“file-cache” on page 60
“find-compressed” on page 158
“find-index” on page 159
“find-index-j2ee” on page 160
“find-links” on page 161
“find-pathinfo” on page 162
“flex-log” on page 239
“force-type” on page 178
“forward-auth-cert” on page 179
“forward-cache-info” on page 180
“forward-cipher” on page 180
“forward-ip” on page 181

“forward-issuer-dn” on page 181

“forward-jroute” on page 182

“forward-keysize” on page 183

“forward-proxy-agent” on page 183

“forward-proxy-auth” on page 184

“forward-secret-keysize” on page 184

“forward-ssl-id” on page 185

“forward-user-dn” on page 185

“forward-via” on page 186

G

“get-client-cert” on page 163

“get-sslid” on page 141

H

“http” on page 61

“http-client-config” on page 186

“http-listener” on page 63

“home-page” on page 145

I

“imagemap” on page 210

“index” on page 65

“index-common” on page 211

“index-simple” on page 214

“init-dav” on page 110

“init-filter-order” on page 110

“init-request-limits” on page 112

“init-uhome” on page 112

“insert-filter” on page 243

“instance” on page 65

J

“jdbc-resource” on page 66

“jvm” on page 67

K

“keep-alive” on page 69

“key-toosmall” on page 215

L

“lifecycle-module” on page 69

“list-dir” on page 216

“load-modules” on page 113

“localization” on page 70

“lock-db” on page 71

“log” on page 71

M

“make-dir” on page 218

“mail-resource” on page 73

“map” on page 146

“match-browser” on page 244

“mime-file” on page 74

N

“ntcgicheck” on page 165

“ntrans-dav” on page 147

“ntrans-j2ee” on page 148

“nt-uri-clean” on page 164

P

“pcheck-dav” on page 166

“pfx2dir” on page 148

“pkcs11” on page 74

“pool-init” on page 114

“profiler” on page 74

“property” on page 75

“property-db” on page 76

“proxy-retrieve” on page 219

Q

“qos” on page 77

“qos-error” on page 241

“qos-handler” on page 141

“qos-limits” on page 77

“query-handler” on page 245

R

“redirect” on page 246

“register-http-method” on page 114

“remove-dir” on page 220

“remove-filter” on page 248

“rename-file” on page 221

“request-policy” on page 78

“require-auth” on page 166

“response-policy” on page 78

“restart” on page 249

“reverse-map” on page 150

“rewrite” on page 151

S

“search-app” on page 79

“search-collection” on page 79

“sed-request” on page 195

“sed-response” on page 200

“send-cgi” on page 222

“send-error” on page 250

“send-file” on page 225

“send-range” on page 226

“send-shellcgi” on page 227

“send-wincgi” on page 228

“server” on page 80

“service-dav” on page 229

“service-dump” on page 231

“service-j2ee” on page 232

“service-trace” on page 234

“servlet-container” on page 83

“session-replication” on page 85

“set-basic-auth” on page 187

“set-cache-control” on page 188

“set-cookie” on page 189

“set-default-type” on page 190

“set-origin-server” on page 201

“set-proxy-server” on page 202

“set-variable” on page 252

“set-virtual-index” on page 167

“shtml-hacktype” on page 190

“shtml-send” on page 235

“single-sign-on” on page 86

“snmp” on page 87

“soap-auth-provider” on page 87

“ssl” on page 88

“ssl-check” on page 168

“ssl-client-config” on page 191

“ssl-logout” on page 169

“ssl-session-cache” on page 93

“stats” on page 94

“stats-xml” on page 236

“strip-params” on page 152

T

“thread-pool” on page 95

“thread-pool-init” on page 115

“time” on page 95

“token” on page 96

“type-by-exp” on page 192

“type-by-extension” on page 193

“type-j2ee” on page 194

U

“unix-home” on page 152

“unix-uri-clean” on page 169

“upload-file” on page 238

V

“variable” on page 96

“virtual-server” on page 97

W

“web-app” on page 99

Index

Numbers and Symbols

- != (not equal to), ACL expression operator, 270
- = (equals), ACL expression operator, 270
- = greater than or equal to, ACL expression operator, 270

A

- access-log-buffer element, 43
- access-log element, 43
- acl-cache element, 44
- acl-db element, 45
- acl-file element, 46
- acl parameter, 154
- ACL
 - attribute expressions, 269-270
 - authentication statements, 267-268
 - authorization statements, 268-269
 - default file, 271
 - default file location, 314
 - file syntax, 266-270
 - in server.xml and obj.conf, 265-266
- add-footer function, 205-206
- add-header function, 206-208
- addCgiInitVars parameter, 235
- AddLog, 121
 - flow of control, 134
 - function descriptions, 239-240
- admin-server directory, 27
- all-requests bucket, 138
- always-use-keep-alive parameter, 187

- and, ACL expression operator, 270
- append-trailer function, 208-209
- assign-name function, 143-144
- atime function, 287
- attribute expressions, ACL, operators, 270
- attribute expressions, ACL, attribute, 269-270
- audit-accesses element, 46
- auth-db element, 47
- auth element, 47
- auth-group parameter, 167
- auth-realm element, 48
- auth-type parameter, 139, 140, 167
- auth-user parameter, 167
- authentication statements, ACL syntax, 267-268
- authorization statements, ACL, 268-269
- AuthTrans, 120
 - flow of control, 127
 - function descriptions, 138-142

B

- backreferences, 295
- basic-auth function, 138-140
- Basic authentication method, 267
- basic-ncsa function, 140-141
- bin directory, 28
- block-auth-cert function, 171
- block-cache-info function, 172
- block-cipher function, 172-173
- block-ip function, 173
- block-issuer-dn function, 173-174

- block-jroute function, 174
- block-keysize function, 174-175
- block-proxy-agent function, 175
- block-proxy-auth function, 176
- block-secret-keysize function, 176-177
- block-size parameter, 114
- block-ssl-id function, 177
- block-user-dn function, 177-178
- block-via function, 178
- bong-file parameter, 157, 169
- boolean, expression results, 282
- bucket, 109-110
- bucket parameter, 138
- bucket
 - all request, 138
 - default, 138
- built-in SAFs in obj.conf, 137-256

C

- cache control directives, 188
- cache
 - ACL, 44
 - DNS, 57
 - enabling memory allocation pool, 114
 - file, 60
- case sensitivity in magnus.conf, 102
- case sensitivity in obj.conf, 136
- cert8.db file location, 314
- certmap.conf, 273-275
- certmap.conf file location, 314
- cgi element, 49
- charset parameter, 179, 190, 192
- check-acl function, 154
- check-age parameter, 159
- check-request-limits function, 155-157
- checkFileExistence parameter, 162
- ChildRestartCallback, 103
- choose function, 287-288
- chroot parameter, 223
- ChunkedRequestBufferSize parameter, 203
- ChunkedRequestTimeout parameter, 203
- cindex-init function, 108-109
- clfilter, 314

- client-cert-nickname parameter, 191
- Client tag, 124-126
- cluster element, 50
- CmapLdapAttr property, 274
- code parameter, 241
- comments
 - in magnus.conf, 103
 - in obj.conf, 136
- compression-level parameter, 198
- config directory, 28
- connection-creation-property element, 50
- connection-lease-property element, 51
- content-type icons, 212
- control parameter, 188
- convert element, 52
- core SAFs in obj.conf, 137-256
- ctime function, 288
- custom log file format, 301-303
- custom log file format components, 302
- custom-resource element, 52
- custom variables, 280

D

- dav-collection element, 54
- dav element, 53
- day of month, 305
- dayofweek, ACL expression operator, 270
- dbm parameter, 140
- dbswitch.conf, 314
- deafault-auth-db-name element, 55
- default-auth-realm-name element, 56
- default-bucket, 138
- default object, 122-127
- default-soap-auth-provider-name element, 56
- default-web.xml, 276
- define-perf-bucket function, 109-110
- delete-file function, 209-210
- deny-existence function, 157-158
- deprecated SAFs, 117
- description parameter, 110
- Digest authentication method, 267
- digest directory, 29
- dir parameter, 149, 162, 223

directive changes in magnus.conf, 309-313
 directives
 magnus.conf, 103
 obj.conf, 137-256
 order of in obj.conf, 135
 syntax in obj.conf, 121
 directory structure, 27
 disable parameter, 114, 161
 display-name element, 56
 DNComps property, 274
 dns-cache element, 57
 dns-cache-init function (deprecated), 117
 dns element, 56
 document-root function, 144-145
 domain parameter, 189
 dorequest parameter, 163
 dotdirok parameter, 165, 169
 dynamic link library, loading, 113-114
 dynamic reconfiguration, 30-31

E

element changes in server.xml, 307-309
 elements in the server.xml file, 41-99
 Else tag, 126-127
 Elseif tag, 126-127
 with regular expressions, 295
 enc parameter, 179, 190, 192, 257
 env-variable element, 58
 Error directive, 121
 flow of control, 134
 function descriptions, 240-242
 error-j2ee function, 240-241
 error parameter, 156
 errors, sending customized messages, 241
 escape function, 288-289
 escape parameter, 247
 event element, 58
 evolving interfaces, 315
 examples, wildcard patterns, 300
 exec-hack parameter, 191
 exp parameter, 192
 expressions, 281-296
 ACL, 269-270

expressions (*Continued*)
 functions, 286-295
 in interpolated strings, 297-298
 literals, 282-283
 operators, 284-286
 regular, 295-296
 results as Booleans, 282
 syntax, 281
 variables, 283-284
 extension parameter, 165
 external function, 289-290
 external-jndi-resource element, 59

F

fancy indexing, 108-109
 fastcgi safes, 256
 fastcgi directory, 29
 file-cache element, 60
 file name extensions
 MIME types, 257
 object type, 130
 file parameter, 205, 206
 files, mapping types of, 257
 filter parameter, 243, 248
 FilterComps property, 274
 filters parameter, 111
 filters, ordering, 110-111
 find-compressed function, 158-159
 find-index function, 159-160
 find-index-j2ee function, 160-161
 find-links function, 161-162
 find-pathinfo-forward parameter, 143, 149
 find-pathinfo function, 162-163
 flex-init function (deprecated), 117
 flex-log function, 239-240
 flex-rotate-init function (deprecated), 117
 flow of control in obj.conf, 127-134
 FlushTimer parameter, 203
 fn parameter, in directives in obj.conf, 121
 force-type function, 130, 178-179
 forcing object type, 130-131
 format parameter, 109
 forward-auth-cert function, 179-180

- forward-cache-info function, 180
- forward-cipher function, 180-181
- forward-ip function, 181
- forward-issue-dn function, 181-182
- forward-jroute function, 182
- forward-keysize function, 183
- forward-proxy-agent function, 183
- forward-proxy-auth function, 184
- forward-secret-keysize function, 184
- forward slashes, 102, 136
- forward-ssl-id function, 185
- forward-user-dn function, 185
- forward-via function, 186
- fragment-size parameter, 198
- from parameter, 143, 149, 151, 168, 247, 250
- funcs parameter, 113
- functions
 - common, 242-256
 - expression, 286-295

G

- get-client-cert function, 163-164
- get-sslid function, 141
- greater than, ACL expression operator, 270
- group parameter, 223
- groupdb parameter, 139
- groupfn parameter, 139
- grpfile parameter, 140

H

- header parameter, 212
- hierarchy, ACL authorization statements, 269-270
- home-page function, 145-146
- htaccess directory, 29
- http-client-config function, 186-187
- http-compression filter, 158, 198-199
- http element, 61
- http-listener element, 63
- httpdate function, 290
- HTTP, registering methods, 114-115

I

- icon-uri parameter, 109
- If tag, 126-127
 - with regular expressions, 295
- ignore parameter, 109
- imagemap function, 210-211
- include directory, 29
- include element, 64
- index-common function, 211-214
- index element, 65
- index-names parameter, 160
- index-simple function, 214-215
- indexing, fancy, 108-109
- init-cgi function (deprecated), 117
- init-clf function (deprecated), 117
- init-dav function, 110
- init-filiter-order function, 110-111
- Init function changes, 313-314
- Init SAFs in magnus.conf, 107
- init-uhome function, 112-113
- InitFn property, 274
- Init, function descriptions, 103
- initializing, the WebDAV subsystem, 110
- Input, 121
 - flow of control, 131
 - function descriptions, 194-196
 - optional parameters, 194
- insert-filter function, 243-244
 - with Input directive, 131
 - with Output directive, 131
- instance directory, 28
- instance element, 65
- interfaces
 - evolving, 315
 - obsolete, 315
 - private, 316
 - stable, 315
 - standard, 315
 - unstable, 315
- internal requests, 135
- interpolated strings, 296-298
- interval parameter, 155
- iponly parameter, 239

J

Java SE SecurityManager, 275
 jdbc-resource element, 66
 jdk directory, 29
 jvm element, 67
 JVM profiler, 74

K

keep-alive element, 69
 keep-alive parameter, 186
 keep-alive-timeout parameter, 186
 KernelThreads, 104
 key-toosmall function, 215-216
 key3.db file location, 314

L

lang parameter, 179, 190, 192, 257
 LateInit parameter, 107
 lc function, 291
 length function, 291-292
 lib directory, 29
 library property, 274
 lifecycle-module element, 69
 line continuation
 in magnus.conf, 102
 in obj.conf, 136
 links, finding hard links, 161-162
 list-dir function, 216-218
 literals
 expression, 282-283
 numeric, 283
 string, 282-283
 load-modules function, 113-114
 loadbal directory, 29
 localization element, 70
 lock-db element, 71
 log analyzer, 239
 log element, 71
 log file format, 301-303
 log file, analyzer for, 239
 login.conf, 275

lookup function, 292-293

M

magnus.conf
 case sensitivity, 102
 comments, 103
 common SAFs, 116-117
 deprecated SAFs, 117
 directive changes, 309-313
 forward slashes, 102
 Init function changes, 313-314
 line continuation, 102
 miscellaneous directives, 103
 parameters for directives, 102
 path names, 102
 quotation marks, 102
 SAFs in, 107-117
 separators, 102
 spaces, 102
 mail-resource element, 73
 make-dir function, 218-219
 map function, 146-147
 match-browser function, 244-245
 matching, special characters, 299-300
 max-age parameter, 189
 max-connections parameter, 155
 max-rps parameter, 155
 maxthreads parameter, 116
 memory allocation, pool-init function, 114
 memory-level parameter, 199
 method parameter, 164, 195, 196, 203
 methods parameter, 115
 mime-file element, 74
 MIME types, 257
 mime.types file, 257
 sample of, 259-263
 MIME types
 determining, 257-258
 file syntax, 259
 generating server response, 258
 processing response in the client, 259
 sample file, 259-263
 type-by-extension, 257-258

minthreads parameter, 116
monitor parameter, 156
month name, 305
mtime function, 293

N

name attribute
 in obj.conf objects, 122
 in objects, 123
name parameter, 110, 116, 143, 149, 153, 239
NameTrans, 120
 flow of control, 127-129
 function descriptions, 142-153
native thread pools, defining in obj.conf, 115-116
NativePoolMaxThreads, 104
NativePoolMinThreads, 104
NativePoolQueueSize, 105
NativePoolStackSize, 105
NativeThread parameter, 114, 115
nice parameter, 223
nocache parameter, 225
nondefault objects, processing, 128-129
nostat parameter, 144
not, ACL expression operator, 270
nsfc.conf, 314
NSIntAbsFilePath parameter, 205, 207
nt-console-init function (deprecated), 117
nt-uri-clean function, 164-165
ntgcicheck function, 165
ntrans-base, 143, 144, 149
ntrans-j2ee function, 148
ntras-dav function, 147-148
numeric literals, 283

O

obj.conf, 266
 case sensitivity, 136
 Client tag, 124-126
 comments, 136
 directive syntax, 121
 directives, 121-122, 137-256

obj.conf (*Continued*)
 Else tag, 126-127
 Elseif tag, 126-127
 flow of control, 127-134
 function flow changes, 134-135
 If tag, 126-127
 Object tag, 122-127
 order of directives, 135
 overview, 119-136
 parameters for directives, 135
 processing other objects, 128-129
 syntax rules, 135-136
Object tag, 122-127
 name attribute, 122
 ppath attribute, 122
objects, processing nondefault objects, 128-129
ObjectType, 121
 flow of control, 129-131
 forcing, 130-131
 function descriptions, 170-194
 setting by file extension, 130
obsolete interfaces, 315
operators
 ACL expressions, 270
 expression, 284-286
 precedence, 285-286
opts parameter, 108
or, ACL expression operator, 270
order, of directives in obj.conf, 135
Output, 121
 flow of control, 131-132
 function descriptions, 196-200
 optional parameters, 196
owner function, 293-294

P

parameters
 for magnus.conf directives, 102
 for obj.conf directives, 135
password.conf, 314
path names
 in magnus.conf, 102
 in obj.conf, 136

path parameter, 146, 154, 156, 157, 167, 246, 250
 PathCheck, 121

- flow of control, 129
- function descriptions, 153-170

 pcheck-dav function, 166
 perf-init function (deprecated), 117
 performance bucket, 138
 pfx2dir function, 128, 148-150
 pkcs11 element, 74
 plugins directory, 29
 pool-init function, 114
 pool parameter, 114
 ppath attribute

- in obj.conf objects, 122
- in objects, 123-124

 predefined SAFs in obj.conf, 137-256
 predefined variables, 277-280
 private interfaces, 316
 processing nondefault objects, 128-129
 profiler element, 74
 property-db element, 76
 property element, 75
 protocol parameter, 187
 proxy-agent parameter, 187
 proxy-retrieve function, 219-220
 pwfile parameter, 113, 153

Q

qos element, 77
 qos-error function, 241-242
 qos-handler function, 141-142
 qos-limits element, 77
 quality of service, *See* qos
 query-handler function, 245-246
 query parameter, 195, 196, 203
 queueSize parameter, 116
 quotes

- in magnus.conf, 102
- in obj.conf, 136

R

readme parameter, 213
 realm parameter, 167
 reconfig, 30
 redirect function, 246-248
 register-http-method function, 114-115
 regular expressions, 295-296
 remove-dir function, 220-221
 remove-filter function, 248-249
 rename-file function, 221-222
 request-handling process, 120

- flow of control, 127-134

 request-policy element, 78
 requests

- internal, 135
- restarted, 134

 require-auth function, 166-167
 require parameter, 164
 response-policy element, 78
 restart function, 249-250
 restarted requests, 134
 reverse-map function, 150-151
 rewrite-content-location parameter, 150, 201
 rewrite function, 151
 rewrite-headername parameter, 150, 202
 rewrite-host parameter, 147, 201
 rewrite-location parameter, 150, 201
 rlimit_as parameter, 223
 rlimit_core parameter, 223
 rlimit_nofile parameter, 223
 root element, 80
 root parameter, 145
 Route, 121
 route-cookie parameter, 201
 route-hdr parameter, 201
 Route

- flow of control, 132
- function descriptions, 200-203

 rules, for editing obj.conf, 135-136

S

SAFs

- deprecated, 117

SAFs (*Continued*)

- in magnus.conf, 107-117
- Init, 103
- predefined in obj.conf, 137-256

samples directory, 29

search-app element, 79

search-collection element, 79

secmod.db file location, 314

secret-keysize parameter, 168

sed parameter, 195, 200

sed-request filter, 195-196

sed-response filter, 200

send-cgi function, 222-224

send-error function, 250-252

send-file function, 225-226

send-range function, 226-227

send-shellcgi function, 227-228

send-wincgi function, 228-229

separators

- in magnus.conf, 102
- in obj.conf, 136

server element, 80

server instance directory, 28

server parameter, 201

server.policy, 275-276

server.xml, 33

- editing, 33
- element changes, 307-309
- elements, 41
- overview, 33
- sample, 38
- schema, 33
- validating, 34
- variables defined in, 224

server

- flow of control, 127-134
- instructions in obj.conf, 121-122
- processing nondefault objects, 128-129

Service, 121

service-dav function, 229-231

service-dump function, 231-232

service-j2ee function, 232-234

service-trace function, 234-235

Service

- default directive, 133-134
- examples, 132-133
- flow of control, 132-134
- function descriptions, 203-238
- optional parameters, 203

servlet-container element, 83

session-replication element, 85

set-basic-auth function, 187-188

set-cache-control function, 188-189

set-cookie function, 189

set-default-type function, 190

set-origin-server function, 201-202

set-proxy-server function, 202-203

set-variable function, 252-256

set-virtual-index function, 167-168

setup directory, 29

shared library, loading, 113-114

shlib parameter, 113

shtml-hacktype function, 190-191

shtml_send function, 235-236

ShtmlMaxDepth parameter, 235

single-sign-on element, 86

snmp element, 87

soap-auth-provider element, 87

spaces

- in magnus.conf, 102
- in obj.conf, 136

SSL authentication method, 267

ssl-check function, 168-169

ssl-client-config function, 191-192

ssl element, 88

ssl-logout function, 169

ssl-session-cache element, 93

ssl2-ciphers element, 89

ssl3-tls-ciphers element, 90

stable interfaces, 315

stackSize parameter, 116

standard interfaces, 315

stats element, 94

stats-init function (deprecated), 117

stats-xml function, 236-237

sticky-cookie parameter, 201

sticky-param parameter, 201

string interpolation, 296-298
string literals, 282-283
strip-params function, 152
subdir parameter, 153
sun-web.xml, 275
syntax
 ACL files, 266-270
 directives in obj.conf, 121
 expressions, 281
 for editing obj.conf, 135-136
 mime.types file, 259

T

tags
 Client, 124-126
 Else, 126-127
 Elseif, 126-127
 If, 126-127
 Object, 122-127
TerminateTimeout, 105
thread-pool element, 95
thread-pool-init function, 115-116
thread pools, defining in obj.conf, 115-116
tildeok parameter, 164
time element, 95
time format strings, 305-306
timefmt parameter, 208
timeofday, ACL expression operator, 270
timeout parameter, 112
timezones parameter, 109
token element, 96
trailer parameter, 208
type-by-exp function, 192-193
type-by-extension function, 193, 258
type-j2ee function, 194
type parameter, 179, 192, 195, 196, 203, 257

U

uc function, 294
Umask, 106
unescape function, 294-295

unix-home function, 152-153
unix-uri-clean function, 169-170
unstable interfaces, 315
upload-file function, 238
uri parameter, 205, 207, 250, 251
URI translation, 135
url parameter, 247
url-prefix parameter, 247
URL, mapping to other servers, 148-150
UseOutputStreamSize parameter, 203
user parameter, 223
userdb parameter, 139
userfile parameter, 140
userfn parameter, 139
util_strftime, 305
uuid function, 295

V

validate-server-cert parameter, 191
variable element, 96
variables, 277-280
 custom, 280
 expression, 283-284
 in interpolated strings, 297
 predefined, 277-280
 resolving, 280
 supported by set-variable, 253
vary parameter, 159, 198
verifycert property, 274
virtual-index parameter, 168
virtual-server element, 97

W

web-app element, 99
Web Server interfaces, 315
WebDAV, 53
 ACL database, 45
 authentication, 47
 collection, 54
 initializing, 110
 lock database, 71

WebDAV (*Continued*)

- property-db, 76
- weekday, 305
- widths parameter, 109
- wildcards
 - examples, 300
 - patterns, 299-300
- window-size parameter, 199