Programmer's Guide

iPlanet Web Server, Enterprise Edition

Version 6.0

8xx-xxxx-xx
May 2001

Contents

This book was last modified on 5/14/01

About This Book

This book is a starting point for developers who need information about using the various APIs and programming technologies that are supported by iPlanet™ Web Server, Enterprise Edition 6.0.

This book summarizes each of the APIs and programming technologies, and tells you where to find more information. In general, each API or programming technology is documented in a separate programmer's guide, with the exception of the API for defining customized server-parsed tags, which is discussed in Chapter 3, "Server-Parsed HTML Tags" in this book.

This book has the following chapters:

- Chapter 1, "Overview"

  This chapter discusses the changes in the APIs that are provided with the server from version 3.*x* to 6.0. It also summarizes the various APIs and programming technologies supported by the server and tells you where to look for more information.

- Chapter 2, "Configuration Files"

  This chapter summarizes the configuration files that the iPlanet Web Server uses.

- Chapter 3, "Server-Parsed HTML Tags"

  This chapter discusses how to use server-parsed tags, lists the standard ones, and explains how to define your own.

- Appendix A, "Configuration Changes Between iWS 4.x and 6.0"

  This appendix summarizes the configuration file changes since iPlanet Web Server 4.*x*.

NOTE	Throughout this manual, all Unix-specific descriptions apply to the Linux operating system as well, except where Linux is specifically mentioned.

Chapter 1

Overview

iPlanet Web Server 6.0 supports a variety of application programming interfaces (APIs) and programming technologies that enable you to do the following:

- generate dynamic content in response to client requests
- modify and extend the behavior of the server
- modify the content stored in the server

This chapter summarizes the various APIs and programming technologies supported by the server. More information on each API or programming technology is provided either in a chapter in this book, or in a separate book.

The sections in this chapter are:

- Configuration Files
- iPlanet Web Server 6.0 APIs
- API Summary
- Changes from Previous Versions

Configuration Files

You can configure iPlanet Web Server using the Server Manager and Class Manager interfaces, or by editing configuration files. Most of the configuration files are in the directory in the *server_root*`/https-`*server_id*`/config` directory. For example, if iPlanet Web Server is installed on a Windows NT machine in `C:\iPlanet\Servers\`, the configuration files for the server `myserver.com` are in:

```
C:\iPlanet\Servers\https-myserver.com\config
```

The main configuration files are `magnus.conf`, `server.xml`, `obj.conf`, and `mime.types`, but there are other configuration files as well. See Chapter 2, "Configuration Files," for an overview of configuration files.

For more detailed information about the files `magnus.conf`, `server.xml`, `obj.conf`, and `mime.types`, see the *NSAPI Programmer's Guide for iPlanet Web Server*.

For information about configuration file changes since iPlanet Web Server 4.*x*, see Appendix A, "Configuration Changes Between iWS 4.x and 6.0."

iPlanet Web Server 6.0 APIs

This section summarizes the various APIs and programming technologies supported by iPlanet Web Server 6.0, discusses how to enable the functionality in iPlanet Web Server 6.0, and mentions where to get more information about them.

The main categories of extensions and modifications you can make to the iPlanet Web Server are:

- Dynamically generating responses (or parts of responses) to requests. The APIs and programming approaches that fall in this category are:
  - Server-Parsed HTML Tags
  - CGI
  - Java Servlets and JavaServer Pages (JSP)
- Modifying the behavior of the server itself by implementing server plugins. Most server plugins are written using Netscape Server API (NSAPI). There are also specialized APIs for writing server plugins, such as the Access Control List API (ACLAPI) which is used for controlling access to server resources.

  The APIs for modifying server behavior are:
  - NSAPI
  - Access Control API
  - Certificate-Mapping API
- Modifying the content of the server, by adding, removing, or modifying resources and directories. To do this, use remote file manipulation.

Server-Parsed HTML Tags

iPlanet Web Server 6.0 provides a C API for defining your own server-side tags. These tags can be used in addition to the standard server-side tags, such as `config`, `include` and so on, in HTML files.

Enabling Server-Parsed Tags

To activate and deactivate the parsing of server-side tags, use the Parse HTML page in the Content Management tab of the Class Manager. This page enables you to switch off parsing of server-side HTML tags, or enable it with or without also enabling the `exec` tag. The page also allows you to specify whether to parse all files or just those with a `.shtml` extension.

The directive in `magnus.conf` that enables the parsing of server-side tags is as follows for Windows NT:

```
Init funcs="shtml_init,shtml_send"
shlib="install_dir/bin/https/bin/Shtml.dll" NativeThread="no"
fn="load-modules"
```

For Unix, the directive is the same except that the file is `Shtml.so`.

The directive in `obj.conf` that enables the parsing of server-side tags is:

```
Service fn="shtml_send" type="magnus-internal/parsed-html"
method="(GET|HEAD)"
```

To enable parsing of server-side tags for files with extensions other than `.shtml`, add the extension to the appropriate line in the `mime.types` file. For example, the following line in `mime.types` indicates that files with either a `.shtml` or `.jbhtml` extension are parsed for server-side tags.

```
type=magnus-internal/parsed-html exts=shtml,jbhtml
```

For More Information

See Chapter 3, “Server-Parsed HTML Tags” for more information about defining and using server-parsed tags.

CGI

Common Gateway Interface (CGI) programs run on the server and generate a response to return to the requesting client. CGI programs can be written in various languages, including C, C++, Java, Perl, and as shell scripts. CGI programs are invoked through URL invocation.

iPlanet Web Server complies with the version 1.1 CGI specification.

Since the server starts up a process each time the CGI script or program runs, this is an expensive method of programming the server.

Enabling CGI

iPlanet Web Server provides two ways to identify CGI programs:

- Specifying CGI Directories. The server treats all files in CGI directories as CGI programs.
- Specifying CGI File Extensions. The server treats all files with the specified extensions as CGI programs.

Specifying CGI Directories

To specify directories that contain CGI programs (and only CGI programs) use the CGI Directory page in the Programs tab of the Class Manager. The server treats all files in these directories as CGI programs.

For each CGI directory, the file `obj.conf` contains a `NameTrans` directive that associates the name `cgi` with each request for a resource in that directory. These directives are automatically added to `obj.conf` when you specify CGI directories in the Class Manager interface, or you can manually add them to `obj.conf` if desired.

For example, the following instruction interprets all requests for resources in `http://`*server-name*`/cgi-local` as requests to invoke CGI programs in the directory `C:/iPlanet/Servers/docs/mycgi`.

```
NameTrans fn="pfx2dir" from="/cgi-local"
dir="C:/iPlanet/Servers/docs/mycgi" name="cgi"
```

The `obj.conf` file must contain the following named object:

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

Do not remove this object from `obj.conf`. If you do, the server will never recognize CGI directories, regardless of whether you specify them in the Class Manager interface or manually add more `NameTrans` directives to `obj.conf`.

Specifying CGI File Extensions

Use the CGI File Type page in the Programs tab of the Class Manager to instruct the server to treat all files with certain extensions as CGI programs, regardless of which directory they reside in. The default CGI extensions are `.cgi`, `.bat` and `.exe`.

To change which extensions indicate CGI programs, modify the following line in `mime.types` to specify the desired extensions. Be sure to restart the server after editing `mime.types`.

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

When the server is enabled to treat all files with an appropriate extensions as CGI programs, the `obj.conf` file contains the following Service directive:

```
Service fn="send-cgi" type="magnus-internal/cgi"
```

Creating Custom Execution Environments for CGI Programs (Unix only)

Before you can create a custom execution environment, you must install the `suid Cgistub` and run it as root:

1. Log in as the superuser.

   ```
   su
   ```

2. Create the `private` directory for `Cgistub`:

   ```
   cd server_root/https-instance
   mkdir private
   ```

3. Copy `Cgistub` to the `private` directory:

   ```
   cd private
   cp ../../bin/https/bin/Cgistub .
   ```

4. Set the owner of `private` to the server user:

   ```
   chown user .
   ```

5. Set the permissions on `private`:

   ```
   chmod 500 .
   ```

6. Set the owner of `Cgistub` to `root`:

   ```
   chown root Cgistub
   ```

7. Set the permissions on `Cgistub`:

   ```
   chmod 4711 Cgistub
   ```

8. You can give each reference to the `send-cgi` SAF in `obj.conf` a `user` parameter. For example:

   ```
   Service fn="send-cgi" user="user"
   ```

   You can use variable substitution. For example, in `server.xml`, give a `VS` (virtual server) element the following `VARS` subelement:

   ```
   <VARS user="user"/>
   ```

   This lets you write the `send-cgi` SAF line in `obj.conf` as follows:

   ```
   Service fn="send-cgi" user="$user"
   ```

   For more information about `send-cgi`, `server.xml`, and `obj.conf`, see the *NSAPI Programmer's Guide for iPlanet Web Server*.

9. Restart the server so these changes take effect.

NOTE Installing `Cgistub` in the *server_root*`/https-`*instance*`/private` directory is recommended. If you install it anywhere else, you must specify the path to `Cgistub` in the `init-cgi` function in `magnus.conf`. For details, see the *NSAPI Programmer's Guide for iPlanet Web Server*.

NOTE It may not be possible to install the `suid` `Cgistub` program on an NFS mount. If you wish to use an `suid` `Cgistub`, you must install your server instance to a local file system.

`Cgistub` enforces the following security restrictions:

- The user the CGI program executes as must have a uid of 100 or greater. This prevents anyone from using `Cgistub` to obtain root access.
- The CGI program must be owned by the user it is executed as and must not be writable by anyone other than its owner. This makes it difficult for anyone to covertly inject and then remotely execute programs.

- `Cgistub` creates its Unix listen socket with 0700 permissions.

NOTE	Socket permissions are not respected on a number of Unix variants, including current versions of SunOS/Solaris. To prevent a malicious user from exploiting `Cgistub`, change the server's temporary directory (using the `magnus.conf TempDir` directive) to a directory accessible only to the server user. For details, see the *NSAPI Programmer's Guide for iPlanet Web Server*.

After you have installed `Cgistub`, you can create custom execution environments in the following ways:

- Specifying a Unique CGI Directory and Unix User and Group for a Virtual Server
- Specifying a Chroot Directory for a Virtual Server

Specifying a Unique CGI Directory and Unix User and Group for a Virtual Server

To prevent a virtual server's CGI programs from interfering with other users, these programs should be stored in a unique directory and execute with the permissions of a unique Unix user and group.

First, create the Unix user and group. The exact steps required to create a user and group vary by operating system. For help, consult your operating system's documentation.

Next, follow these steps to create a `cgi-bin` directory for the virtual server:

1. Log in as the superuser.

   ```
   su
   ```

2. Change to the virtual server directory.

   ```
   cd vs_dir
   ```

3. Create the `cgi-bin` directory.

   ```
   mkdir cgi-bin
   chown user:group cgi-bin
   chmod 755 cgi-bin
   ```

Now you can set the virtual server's CGI directory, user, and group in one of these ways:

- Use the `dir`, `user`, and `group` parameters of the `send-cgi` Service SAF in the `obj.conf` file (see the *NSAPI Programmer's Guide for iPlanet Web Server*)
- Enter this information using the Settings tab of the Virtual Server Manager (see the *iPlanet Web Server Administrator's Guide*)

Specifying a Chroot Directory for a Virtual Server

To further improve security, these CGI scripts should be prevented from accessing data above and outside of the virtual server directory.

First, set up the chroot environment. The exact steps required to set up the chroot environment vary by operating system. For help, consult your operating system's documentation. The `man` pages for `ftpd` and `chroot` are often a good place to start.

These are the steps required for Solaris versions 2.6 through 8:

1. Log in as the superuser.

   ```
   su
   ```

2. Change to the chroot directory. This is typically the *vs_dir* directory mentioned in the previous section.

   `cd` *chroot*

3. Create `tmp` in the chroot directory:

   ```
   mkdir tmp
   chmod 1777 tmp
   ```

4. Create `dev` in the chroot directory:

   ```
   mkdir dev
   chmod 755 dev
   ```

5. List `/dev/tcp`, and note the major and minor numbers of the resulting output. In this example, the major number is 11 and the minor number is 42:

   ```
   ls -lL /dev/tcp
   crw-rw-rw-   1 root     sys       11, 42 Apr  9  1998 /dev/tcp
   ```

6. Create the `tcp` device using the major and minor numbers:

   ```
   mknod dev/tcp c 11 42
   chmod 666 dev/tcp
   ```

7. Repeat steps 5 and 6 for each of the following devices (each device will have a different major and minor combination):

   ```
   /dev/udp
   /dev/ip
   /dev/kmem
   /dev/kstat
   /dev/ksyms
   /dev/mem
   /dev/null
   /dev/stderr
   /dev/stdin
   /dev/stdout
   /dev/ticotsord
   /dev/zero
   ```

8. Set permissions on the devices in dev in the chroot directory:

   ```
   chmod 666 dev/*
   ```

9. Create and populate `lib` and `usr/lib` in the chroot directory:

   ```
   mkdir usr
   ```

   ```
   mkdir usr/lib
   ```

   ```
   ln -s /usr/lib
   ```

   ```
   ln /usr/lib/* usr/lib
   ```

   You can ignore the messages this command generates.

   If the `/usr/lib` directory is on a different file system, replace the last command with the following:

   ```
   cp -rf /usr/lib/* usr/lib
   ```

10. Create and populate `bin` and `usr/bin` in the chroot directory:

    ```
    mkdir usr/bin
    ```

    ```
    ln -s /usr/bin
    ```

    ```
    ln /usr/bin/* usr/bin
    ```

    You can ignore the messages this command generates.

    If the `/usr/bin` directory is on a different file system, replace the last command with the following:

    ```
    cp -rf /usr/bin/* usr/bin
    ```

11. Create and populate `etc` in the chroot directory:

```
mkdir etc

ln /etc/passwd /etc/group /etc/netconfig etc
```

12. Test the chroot environment:

```
chroot chroot bin/ls -l
```

The output should look something like this:

```
total 14
lrwxrwxrwx   1 root     other      8 Jan 13 03:32 bin -> /usr/bin
drwxr-xr-x   2 user     group    512 Jan 13 03:42 cgi-bin
drwxr-xr-x   2 root     other    512 Jan 13 03:28 dev
drwxr-xr-x   2 user     group    512 Jan 13 03:26 docs
drwxr-xr-x   2 root     other    512 Jan 13 03:33 etc
lrwxrwxrwx   1 root     other      8 Jan 13 03:30 lib -> /usr/lib
drwxr-xr-x   4 root     other    512 Jan 13 03:32 usr
```

Now you can set the virtual server's chroot directory in one of these ways:

- Use the `chroot` parameter of the `send-cgi` Service SAF in the `obj.conf` file (see the *NSAPI Programmer's Guide for iPlanet Web Server*)
- Enter this information using the Settings tab of the Virtual Server Manager (see the *iPlanet Web Server Administrator's Guide*)

Adding CGI Programs to the Server

To add CGI programs to the iPlanet Web Server, simply do one of the following:

- Drop the program file in a CGI directory (if there are any).
- Give it a file name that the server recognizes as a CGI program and put it in any directory at or below the document root (if CGI file type recognition has been activated).

For Unix, make sure the program file has execute permissions set.

Windows NT CGI and Shell CGI Programs

For information about installing CGI and shell CGI programs on Windows NT using the Class Manager interface, see the *iPlanet Web Server Administrator's Guide.*

Perl CGI Programs

You cannot run CGIs using Perl 5.6.*x* with the `-w` flag. Instead, include the following code in the file:

```
use warnings;
```

CGI Variables

In addition to the standard CGI variables, you can use the iPlanet Web Server CGI variables in Table 1-1 in CGI programs to access information about the client certificate if the server is running in secure mode. The CLIENT_CERT and REVOCATION variables are available only when client certificate based authentication is enabled.

Table 1-1 CGI Variables

Variable	Description
SERVER_URL	The URL of the server that the client requested
HTTP_*xxx*	An incoming HTTP request header, where *xxx* is the name of the header
HTTPS	ON if the server is in secure mode and OFF otherwise
HTTPS_KEYSIZE	The keysize of the SSL handshake (available if the server is in secure mode)
HTTPS_SECRETKEYSIZE	The keysize of the secret part of the SSL handshake (available if the server is in secure mode)
HTTPS_SESSIONID	The session ID for the connection (available if the server is in secure mode)
CLIENT_CERT	The certificate that the client provided (binary DER format)
CLIENT_CERT_SUBJECT_DN	The Distinguished Name of the subject of the client certificate
CLIENT_CERT_SUBJECT_OU	The Organization Unit of the subject of the client certificate
CLIENT_CERT_SUBJECT_O	The Organization of the subject of the client certificate
CLIENT_CERT_SUBJECT_C	The Country of the subject of the client certificate
CLIENT_CERT_SUBJECT_L	The Location of the subject of the client certificate

Table 1-1 CGI Variables

Variable	Description
CLIENT_CERT_SUBJECT_ST	The State of the subject of the client certificate
CLIENT_CERT_SUBJECT_E	The E-mail of the subject of the client certificate
CLIENT_CERT_SUBJECT_UID	The UID part of the CN of the subject of the client certificate
CLIENT_CERT_ISSUER_DN	The Distinguished Name of the issuer of the client certificate
CLIENT_CERT_ISSUER_OU	The Organization Unit of the issuer of the client certificate
CLIENT_CERT_ISSUER_O	The Organization of the issuer of the client certificate
CLIENT_CERT_ISSUER_C	The Country of the issuer of the client certificate
CLIENT_CERT_ISSUER_L	The Location of the issuer of the client certificate
CLIENT_CERT_ISSUER_ST	The State of the issuer of the client certificate
CLIENT_CERT_ISSUER_E	The E-mail of the issuer of the client certificate
CLIENT_CERT_ISSUER_UID	The UID part of the CN of the issuer of the client certificate
CLIENT_CERT_VALIDITY_START	The start date of the certificate
CLIENT_CERT_VALIDITY_EXIRES	The expiration date of the certificate
CLIENT_CERT_EXTENSION_*xxx*	The certificate extension, where *xxx* is the name of the extension
REVOCATION_METHOD	The name of the certificate revocation method if it exists
REVOCATION_STATUS	The status of certificate revocation if it exists

For More Information

A myriad of information about writing CGI programs is available. A good starting point is “The Common Gateway Interface” at:

`http://hoohoo.ncsa.uiuc.edu/cgi/overview.html`

Java Servlets and JavaServer Pages (JSP)

iPlanet Web Server 6.0 supports the Java Servlet Specification version 2.2 (including Web Application and WAR file support) and the JavaServer Pages (JSP) Specification version 1.1.

Java servlets are server-side Java programs that can be used to generate dynamic content in response to client requests in much the same way as CGI programs do. Servlets are accessed through URL invocation.

You create servlets using the Servlets API, which was created by Sun Microsystems. iPlanet Web Server 6.0 includes all the files necessary for developing and running Java Servlets. You can compile servlets using any Java compiler you like, so long as the `servlet.jar` file is accessible to your Java compiler. The `servlet.jar` file is in the server installation directory at:

`/bin/https/jar`

For information about using the Servlet API, see the Java Servlet API documentation from Sun Microsystems at:

`http://java.sun.com/products/servlet/index.html`

A JavaServer Page (JSP) is a page, much like an HTML page, that can be viewed in a web browser. However, in addition to HTML tags, it can include a set of JSP tags and directives intermixed with Java code that extend the ability of the web page designer to incorporate dynamic content in a page. These additional features provide functionality such as displaying property values and using simple conditionals.

For more information on using JavaServer Pages, see the JavaServer Pages documentation from Sun Microsystems at:

`http://java.sun.com/products/jsp/index.html`

Enabling Java Servlets and JavaServer Pages

To enable servlets, select the Java tab in the Server manager, then select the Enable / Disable Servlets / JSP tab. Check the Enable Java Globally box to enable servlets for the entire server. Check the Enable Java for Class box to enable servlets for a single virtual server class. You cannot enable servlets for a class unless Java is globally enabled. By default, Java is globally enabled and enabled for each virtual server class.

To enable JSPs, you must also include the `jsp-servlet` element with `enable=true` in the `web-apps.xml` file and add `tools.jar` to the JVM classpath. If you want to run uncompiled JSPs, you must also install the Java Development Kit (JDK).

When you install iPlanet Web Server 6.0, you can choose to install the Java Runtime Environment (JRE), which is provided with iPlanet Web Server, or you can specify a path to a JDK. The JDK is not bundled with the iPlanet Web Server, but you can download it for free from Sun Microsystems at:

`http://java.sun.com/products/jdk/1.2/`

The server can run servlets and precompiled JSPs using the JRE, but it needs the JDK to run uncompiled JSPs.

iPlanet Web Server 6.0 requires you to use an official version of JDK 1.2. For details, see the *Programmer's Guide to Servlets for iPlanet Web Server*.

Regardless of whether you choose to install the JRE or specify a path to the JDK during installation, you can tell the iPlanet Web Server to switch to using either the JRE or JDK at any time by using the "Configure JRE / JDK Paths" page in the Global Settings tab of the Administration Server.

The `magnus.conf` file contains the following `Init` directives. The first one loads the servlets library and makes the servlet-related functions available to the iPlanet Web Server. The other two directives initialize the servlet engine. The `shlib` value shown is for Windows NT.

```
Init shlib="d:/server_root/bin/https/bin/NSServletPlugin.dll"
funcs="NSServletEarlyInit,NSServletLateInit,NSServletNameTrans,N
SServletService" shlib_flags="(global|now)" fn="load-modules"
Init EarlyInit="yes" fn="NSServletEarlyInit"
Init LateInit="yes" fn="NSServletLateInit"
```

For Unix, the `shlib` value is as follows:

`shlib="`*server_root*`/bin/https/lib/libNSServletPlugin.so"`

The file `obj.conf` also has other directives that relate to servlets, and defines several additional objects for processing requests for servlets.

Adding Servlets and JavaServer Pages to the Server

You can make servlets and JSPs accessible to clients in one of these two ways:

- Include the servlets in web applications and deploy those web applications.
- Configure the servlets in the default virtual server. This is provided for backward compatibility with iPlanet Web Server 4.*x*.

For more information, see the *Programmer's Guide to Servlets for iPlanet Web Server*.

For More Information

For more information about using servlets in iPlanet Web Server 6.0, see the book *Programmer's Guide to Servlets for iPlanet Web Server*.

For more information about using the Servlets API to create servlets, see the Java Servlet API documentation from Sun Microsystems at:

`http://java.sun.com/products/servlet/index.html`

For information about creating JSPs, see Sun Microsystem's JavaServer Pages web site at:

`http://java.sun.com/products/jsp/index.html`

NSAPI

Netscape Server Application Programming Interface (NSAPI) is a set of C functions for implementing extensions to the server. These extensions are known as server plugins.

Using NSAPI, you can write plugins to extend the functionality of the iPlanet Web Server. An NSAPI plugin defines one or more Server Application Functions (SAFs). You can develop SAFs for implementing custom authorization, custom logging, or for other ways of modifying how the iPlanet Web Server handles requests.

The file `obj.conf` contains instructions (known as directives) that tell the server how to process requests received from clients. Each instruction is enacted either during server initialization or during a particular stage of the request-handling process. Each instruction invokes a server application function (SAF).

For example, the following instruction is invoked when the request method is GET and the requested resource is of type `text/html`. This instruction calls the `append-trailer` function with a trailer argument of `<H4><font color=green>Served by 6.0</font></H4>`. (The `append-trailer` function simply returns the requested resource, in this case an HTML file, to the client, and appends the given trailer to it.)

```
Service method=GET type="text/html" fn=append-trailer
trailer="<H4><font color=green>Served by 6.0</font></H4>"
```

iPlanet Web Server 6.0 comes with a set of pre-defined SAFs. It also comes with a library of NSAPI functions for developing your own SAFs to modify the way that the server handles requests.

Enabling NSAPI

You don't enable NSAPI as such. You use it to develop server application functions (SAFs) to use in the file `obj.conf`. The file `obj.conf` is essential for the operation of the server -- if it does not exist, the server cannot work, since it has nowhere to look for instructions on how to handle requests.

When defining new SAFs, include the header function `nsapi.h` (which is in *server_root*`/plugins/include`) to get access to all the NSAPI functions.

Installing NSAPI Plugins (SAFs)

To load new NSAPI plugins containing customized SAFs into the server, add an `Init` directive to `magnus.conf` to load the shared library file that defines the new SAFs. This directive must call the `load-modules` function, which takes the following arguments:

- `shlib` -- the shared library to load.
- `funcs` -- the functions to be made available to the server.

For More Information

For information about the following topics, see the *NSAPI Programmer's Guide for iPlanet Web Server*.

- the directives in `obj.conf` and how they determine how the server handles requests
- the pre-defined SAFs that ship with iPlanet Web Server 6.0
- the NSAPI functions available for writing custom SAFs
- how to write custom SAFs
- how to load custom SAFs into the iPlanet Web Server by adding an `Init` directive to `magnus.conf` that calls `load-modules`

Access Control API

The Access Control API is a C API that lets you programmatically control who has access to what on the iPlanet Web Server.

Access control lists (ACLs) determine who has what kind of access privileges to which resources on the server. Each ACL contains a list of access control entries. The following access control entry, for example, says that all access is denied to everyone for any resource having a URI that starts with `/private`.

```
acl "uri=/private/*";
deny (all)
(user = "anyone");
```

To create access control lists, use the Restrict Access page in the Preferences tab of the Server Manager interface. You can also edit the files that contain the ACLs used by the server.

The default access control list resides in the directory *server_root*`/httpacl`. The default ACL file is `generated.https-`*server_id*`.acl`. There is also a file called `genwork.https-`*server_id*`.acl` that is a working copy the server uses until you save and apply your changes when working with the user interface. When editing the ACL file, you might want to work in the `genwork` file and then use the Server Manager to load and apply the changes.

With iPlanet Web Server 6.0, you can configure and reference multiple ACL files. For more information, see the discussion of the `server.xml` file in the *NSAPI Programmer's Guide for iPlanet Web Server*.

With the Access Control API, you can manipulate access control lists (ACLs), read and write ACL files, and evaluate and test access to resources on the server.

You can also define your own attributes for authentication. For example, you might want to authenticate users based on email address or on the URL that referred them to the resource. For example:

```
allow (read) referer="*www.acme.com*"
```

You can also authenticate the client based on your own authentication methods and databases.

Registering New Authentication Services

To tell the server to use your attributes for authentication, you need to define your own Loadable Authentication Service (LAS), which is an NSAPI plugin. You load it into the server in the usual manner by adding the following directives to `magnus.conf`:

- An `Init` directive that invokes the `load-modules` function to load the shared library.
- An `Init` directive that calls the initialization function.

For More Information

For information about using the ACL API, see the *Access Control Programmer's Guide*. For information about the syntax for editing ACL files, see Appendix A in the same book.

For more information about configuring ACL files for virtual servers, see the discussion of the `server.xml` file in the *NSAPI Programmer's Guide for iPlanet Web Server*.

For information about changes to the access control API in iPlanet Web Server 6.0, see the comments in the *server_root*`/plugins/include/nsacl/aclapi.h` file.

Certificate-Mapping API

The Certificate-Mapping API consists of data structures and functions used to manage certificate mapping.

When a user authenticates to an iPlanet server by sending a client certificate to the server, the server uses information in the certificate to search the user directory for the user's entry.

You can configure some parts of this process by editing the file `certmap.conf`. This file specifies:

- How the server searches the directory for the user's entry
- Whether the server goes through an additional step of verifying that the user's certificate matches the certificate presented to the server

For more information about this file, see Chapter 2, "Configuration Files."

You can also modify this “certificate to directory entry” process programmatically. iPlanet servers include a set of API functions (referred to here as the Certificate-Mapping API functions) that allow you to control this process. You can write your own functions to customize how certificate subject entries are found in the directory.

To use this API, you need to have a copy of the Directory SDK. You can download a copy of this SDK from this site:

`http://developer.iplanet.com/`

For More Information

For information about using the certificate-mapping API, see the *Certificate-Mapping Programmer’s Guide*.

API Summary

The following table lists the APIs available in iPlanet Web Server 6.0.

Table 1-2 APIs available in iPlanet Web Server 6.0

API/Interface/Protocol	Language	Documentation
Interfaces for Generating Dynamic Content		
Custom Server-Parsed HTML Tags	C	*Chapter 3, “Server-Parsed HTML Tags”*
Java Servlets	Java	*Programmer’s Guide to Servlets for iPlanet Web Server*
JavaServer Pages	HTML with additional JSP tags	*Programmer’s Guide to Servlets for iPlanet Web Server*
CGI (one process per request)	C, C++, Perl, shell, and other languages	*The Common Gateway Interface*
APIs for Writing Server Plugins		
NSAPI (in-process shared object / DLL)	C, C++	*NSAPI Programmer’s Guide for iPlanet Web Server*
Access Control API	C, C++	*Access Control Programmer’s Guide*
Certificate-Mapping API	C, C++	*Certificate-Mapping Programmer’s Guide*

Changes from Previous Versions

Changes from previous versions of iPlanet Web Server are summarized in the following sections:

- API Changes Since iPlanet Web Server 3.x
- API Changes Since iPlanet Web Server 4.0
- API Changes Since iPlanet Web Server 4.1

For specific information about configuration files, see Appendix A, “Configuration Changes Between iWS 4.x and 6.0.”

API Changes Since iPlanet Web Server 3.x

- New API for defining customized server-parsed tags as NSAPI plugins has been added. For more information, see Chapter 3, “Server-Parsed HTML Tags.”
- Server side Java applets (HttpApplets) are not supported. Use Java servlets instead.
- Agents API is not supported.
- NSAPI has new features.

API Changes Since iPlanet Web Server 4.0

- Java Servlets version 2.2.1 and JavaServer Pages 1.1 are supported.
- HTTP/1.1 cookies are supported.
- Descriptions of CGI variables have been added to the “CGI Variables” section in this chapter.
- You can invoke servlets as SSI in HTML pages by using the `<SERVLET>` tag, as discussed in Chapter 3, “Server-Parsed HTML Tags.”
- NSAPI has new features.

API Changes Since iPlanet Web Server 4.1

- Programs such as servlets modify a virtual server instead of the server as a whole. (To add programs as in iPlanet Web Server 4.1, you can configure only one virtual server.)
- Web applications are now supported as described in the Java Servlet 2.2 API Specification.
- NSAPI has new features. For details, see the *NSAPI Programmer's Guide for iPlanet Web Server*.
- Some configuration files have changed. For details, see Appendix A, "Configuration Changes Between iWS 4.x and 6.0."
- The access control API has changed. For details, see the comments in the *server_root*`/plugins/include/nsacl/aclapi.h` file.

Chapter 2

Configuration Files

Configuration files control how iPlanet Web Server operates. This appendix summarizes the **Purpose**, **Location**, and **Contents** or **Syntax** of each configuration file, then briefly describes all directives or parameters allowed in the file (if any) in a table. Cross references are listed after **See Also** headings when other manuals describe some of the directives or parameters in more detail.

For information about configuration file changes since iPlanet Web Server 4.*x*, see Appendix A, “Configuration Changes Between iWS 4.x and 6.0.”

The following configuration files are described in alphabetical order:

- backups.conf
- certmap.conf
- cjava.properties
- cluster.xml
- contexts.properties
- cron.conf
- dbswitch.conf
- iwsstats.xml
- jvm12.conf
- magnus.conf
- mime.types
- ns-cron.conf
- nsfc.conf
- obj.conf

- password.conf
- rules.properties
- server.xml
- servers.lst
- servlets.properties
- web.xml
- web-apps.xml

backups.conf

Purpose

Tracks backups of configuration files.

Location

server_root/https-admserv/conf_bk

server_root/https-*server_id*/conf_bk

Syntax

file:*path_to_backup*:*version*:*timestamp*:*original_path*
...

backup_version_history
...

Contents

```
backups.conf:Version 4.0

https-admserv.acl:httpacl/genwork.https-admserv.acl:2:952103058:httpacl/genwork.https-admserv.acl
magnus.conf:https-admserv/conf_bk/magnus.conf:2:952103070:https-admserv/config/magnus.conf
obj.conf:https-admserv/conf_bk/obj.conf:2:952103060:https-admserv/config/obj.conf
mime.types:https-admserv/conf_bk/mime.types:2:952103060:https-admserv/config/mime.types
jvm12.conf:https-admserv/conf_bk/jvm12.conf:2:952103068:https-admserv/config/jvm12.conf
servlets.properties:https-admserv/conf_bk/servlets.properties:2:952103068:https-admserv/config/servlets.properties
contexts.properties:https-admserv/conf_bk/contexts.properties:2:952103068:https-admserv/config/contexts.properties
rules.properties:https-admserv/conf_bk/rules.properties:2:952103068:https-admserv/config/rules.properties

952103058:https-admserv.acl/1::
952103060:https-admserv.acl/1:magnus.conf/1::
952103060:https-admserv.acl/1:magnus.conf/1:obj.conf/1::
952103060:https-admserv.acl/1:magnus.conf/1:obj.conf/1:mime.types/1::
952103068:https-admserv.acl/1:magnus.conf/1:obj.conf/1:mime.types/1:jvm12.conf/1::
952103068:https-admserv.acl/1:magnus.conf/1:obj.conf/1:mime.types/1:jvm12.conf/1:servlets.properties/1::
952103068:https-admserv.acl/1:magnus.conf/1:obj.conf/1:mime.types/1:jvm12.conf/1:servlets.properties/1:
     contexts.properties/1::
```

```
952103068:https-admserv.acl/1:magnus.conf/1:obj.conf/1:mime.types/1:jvm12.conf/1:servlets.properties/1:
     contexts.properties/1:rules.properties/1::
952103068:https-admserv.acl/2:magnus.conf/2:obj.conf/2:mime.types/2:jvm12.conf/2:servlets.properties/2:
     contexts.properties/2:rules.properties/2::Added ExtraPath for Java.--EOF--
```

Table 2-1 backup.conf

Item	Description
file	The name of the file for which backups are made. Examples are `server.xml`, `obj.conf`, and so on.
path_to_backup	The path to the backup of the file.
version	The version of the file.
timestamp	The timestamp of the backup.
original_path	The path to the file that is backed up.
backup_version_history	A version history listing for the files.

certmap.conf

Purpose

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

Location

server_root`/bin/https/install/misc`

server_root`/userdb`

Syntax

`certmap` *name issuerDN*
name`:`*property1* [*value1*]
name`:`*property2* [*value2*]
`...`

The default certificate is named `default`, and the default *issuerDN* is also named `default`. Therefore, the first certmap defined in the file must be as follows:

```
certmap default default
```

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

See Also

iPlanet Web Server Administrator's Guide

Table 2-2 certmap.conf

Property	Allowed Values	Default Value	Description
`DNComps`	See Description	Commented out	Used to form the base DN for performing an LDAP search while mapping the cert to a user entry. Values are as follows: • Commented out: takes the user's DN from the cert 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 cert to a user entry. Values are as follows: • 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`	LDAP attribute name	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`.

cjava.properties

Purpose

Defines servlet and JVM error messages.

Location

server_root`/bin/https/res`

Syntax

error = *message*

Errors are not listed here because you should not edit them. You can edit the messages, but this is not recommended.

cluster.xml

Purpose

Defines a cluster of servers for backups and failover in a server farm. This file is present only if at least one cluster has been defined.

Location

server_root/https-admserv/config

Syntax

Most of the file has the following basic XML syntax, with nested elements:

```
<ELEMENT attribute="value" attribute="value" ... >
    <SUBELEMENT attribute="value" attribute="value" ... />
</ELEMENT>
```

In Table 2-3, elements are in bold to distinguish them from attributes.

See Also

iPlanet Web Server Administrator's Guide

Table 2-3 cluster.xml

Element/Attribute	Allowed Subelements or Values	Description
CLUSTER	**MASTER**	Defines a cluster of web servers.
id	A text string	The ID of the cluster.
MASTER	**SLAVE**	Defines the master server in the cluster.
id	A text string	The ID of the master.
hostname	Usually the *server_id*	The host name of the master.
adminport		The administration port of the master.
instance	https-*server_id*	The name of the server instance on the master.
SLAVE	(none)	Defines a slave server in the cluster.
id	A text string	The ID of the slave.

Table 2-3 cluster.xml

Element/Attribute	Allowed Subelements or Values	Description
`hostname`	Usually the *server_id*	The host name of the slave.
`adminport`		The administration port of the slave.
`instance`	`https-`*server_id*	The name of the server instance on the slave.
`protocol`	`http`, `https`	The protocol used for communication with the client.
`substitute`	A master or slave `id` or `null`	The ID of a substitute server if this server is down.

contexts.properties

Purpose

Provided for backward compatibility with iPlanet Web Server 4.*x*. Using `web-apps.xml` instead to configure servlets is recommended.

Defines contexts, which allow multiple servlets to exchange data and access each other's fields. Contexts are useful for defining virtual servers or for code isolation. The default context is `global`. In iPlanet Web Server 6.0, supported for the default virtual server only.

Location

server_root`/https-admserv/config`

server_root`/https-admserv/conf_bk`

server_root`/https-`*server_id*`/config`

server_root`/https-`*server_id*`/conf_bk`

Syntax

`context.`*context_name*`.`*property*`=`*value*

Table 2-4 lists the properties and their possible values.

See Also

Programmer's Guide to Servlets for iPlanet Web Server

The `server.xml` and `web-apps.xml` files

Appendix A, "Configuration Changes Between iWS 4.x and 6.0"

The Servlet 2.2 API specification at:

`http://java.sun.com/products/servlet/index.html`

Table 2-4 contexts.properties

Property	Allowed Value(s)	Default Value	Description
`sessionmgr`	A session manager object	`com.iplanet.server.http.session.IWSSessionMan-ager` (all on one line, no dash)	The name of the session manager for the context. Some session managers, such as `MMapSessionManager`, can only be instantiated once within the server.
`sessionmgr.initArgs`	Comma separated *name=value* pairs	Depends on session manager	A list of parameters specific to the session manager. For more information, see the *Programmer's Guide to Servlets for iPlanet Web Server.*
`initArgs`	Comma separated *name=value* pairs	`initial=0`	A list of additional context attributes.
`respondCookieVersion`	A cookie version number	`0`	Tells the server whether to respond with a specific cookie version.
`tempDir`	A path	`/tmp`	Sets up the Servlet API 2.2 property for the temporary directory. Use forward slashes only.
`reloadInterval`	Number of seconds	`5`	The time interval within which the server checks for JSP and servlet files being modified. Applies to the `global` context only.
`bufferSize`	Number of bytes	`4096`	The initial HTTP output stream buffer size.
`docRoot`	A path with forward slashes	Web server's document root	The document root for the context. If `docRoot` is not specified, the web server's document root is used.

Table 2-4 contexts.properties

Property	Allowed Value(s)	Default Value	Description
`inputStreamLengthCheck`	`true`, `false`	`true`	Tells a `ServletInputStream` to stop reading data when `Content-Length` number of bytes are read.
`outputStreamFlushTimer`	Number of seconds	`0`	Forces the stream to flush the data if the specified number of seconds has elapsed since the last flush. If set to `0`, this property is ignored.
`uri`	A URI	`/`	An additional URI prefix which serves as a context base.
`authdb`	A database name	`default`	The name of the authentication database. This database must also be defined in the `server.xml` file in the `database` attribute of a `USERDB` element, and in the `dbswitch.conf` file.
`classpath`	A path		The global classpath for this context.
`singleClassLoader`	`true`, `false`	`false`	Tells the servlet engine whether to use a single class loader for all servlets in the context.
`serverName`	A server instance name		Used to specify the server instance that runs the servlets in the context.
`contentTypeIgnoreFromSSI`	`true`, `false`	`true`	Ignores `setContentType` when invoked from SSI if `true`.

Table 2-4 contexts.properties

Property	Allowed Value(s)	Default Value	Description
parameterEncoding	none, auto, responseCT, or a specific *encoding* such as utf8 or Shift_JIS	auto	Advises the web server on how to decode parameters from forms: • *encoding*: uses the specified encoding. • none: uses the system default encoding. • auto: tries to figure out the encoding from, in order, 1) the charset , 2) the parameter Encoding attribute, then 3) a hidden form field, such as j_encoding. Otherwise same as none. • responseCT: tries to figure out the encoding from the response content type if it is available, otherwise, same as none.
isModifiedCheckAggressive	true, false	false	Determines whether the servlet loader aggressively checks dependencies to reload modified servlets.

cron.conf

Purpose

Allows you to program the server to perform maintenance activities at regular intervals, such as back up log files. The ns-cron.conf file controls whether or not the cron.conf file is activated.

Location

server_root/https-admserv/config

Syntax

```
<Object name=name>
Command "command"
User user
Time nn:nn
Days day day ...
</Object>
```

The following is an example of a `cron.conf` file that manages log rotation.

```
<Object name=https-server_id_rotatelg0>
Command "server_root/bin/https/httpadmin/bin/rotlog https-server_id"
User LocalSystem
Time 03:00
Days Sun Mon Tue Wed Thu Fri Sat
</Object>
```

Table 2-5 cron.conf

Directive	Allowed Values	Description
`name`		An object name for the maintenance activity.
`Command`		The command or script that performs the maintenance activity. This can be any command or executable file.
`User`		The name of the system user.
`Time`	A 24-hour time	The time of day at which the activity takes place.
`Days`	`Sun`, `Mon`, `Tue`, `Wed`, `Thu`, `Fri`, `Sat`	The days of the week on which the activity takes place.

dbswitch.conf

Purpose

Specifies the LDAP directory that iPlanet Web Server uses.

Location

server_root`/userdb`

Syntax

```
directory name LDAP_URL
name:property1 [value1]
name:property2 [value2]
...
```

The default contents of this file are as follows:

```
directory default null:///none
```

Edit the file as follows for anonymous binding over SSL:

```
directory default ldaps://directory.netscape.com:636:/dc%3Dcom
```

Edit the file as follows for anonymous binding *not* over SSL:

```
directory default ldap://directory.netscape.com:389:/dc%3Dcom
```

See Also
NSAPI Programmer's Guide for iPlanet Web Server, Chapter 8

Table 2-6 dbswitch.conf

Property	Allowed Values	Default Value	Description
`nsessions`	A positive integer	`8`	The number of LDAP connections for the database.
`dyngroups`	`off`, `on`, `recursive`	`on`	Determines how dynamic groups are handled. If `off`, dynamic groups are not supported. If `on`, dynamic groups are supported. If `recursive`, dynamic groups can contain other groups.
`binddn`	A valid DN		The DN used for connecting to the database. If both `binddn` and `bindpw` are not present, binding is anonymous.
`bindpw`			The password used for connecting to the database. If both `binddn` and `bindpw` are not present, binding is anonymous.
`dcsuffix`	A valid DN (relative to the LDAP URL)	(none)	If present, the default value of the base DN for the request's virtual server is determined by a DC tree search of the connection group's `servername` attribute, starting at the `dcsuffix` DN. Otherwise, the default value of the base DN is the `base DN` value in the LDAP URL. The `basedn` attribute of a `USERDB` element in the `server.xml` file overrides this value.
`digestauth`	`off`, `on`	`off`	Specifies whether the database can do digest authentication. If `on`, a special Directory Server plugin is required. For information about how to install this plugin, see the *iPlanet Web Server Administrator's Guide*.

iwsstats.xml

Purpose

Reports server performance statistics. Configured via the `stats-xml` SAF in `obj.conf`, and present only if this SAF is used. This file is intended to be read but not modified.

Location

Located here, dynamically generated:

server_root`/https-`*server_id*`/stats-xml/iwsstats.xml`

You can view it here:

`http://`*server_id*`:`*port*`/stats-xml/iwsstats.xml`

Syntax

The file has the following basic XML syntax, with nested elements:

```
<ELEMENT attribute="value" attribute="value" ... >
    <SUBELEMENT attribute="value" attribute="value" ... />
</ELEMENT>
```

In Table 2-7, elements are in bold to distinguish them from attributes.

See Also

NSAPI Programmer's Guide for iPlanet Web Server, Chapter 3

Table 2-7 iwsstats.xml

Element/Attribute	Subelements or Values	Description
`stats`	**`server`**	The top-level statistics element. All `stats-xml` statistics information is contained within this element.
`enabled`	`0` (off), `1` (on)	Indicates whether statistics collection is enabled (on).
`versionMajor`		The major version of the statistics format. In this version of iPlanet Web Server, the value is frozen at 1.
`versionMinor`		The minor version of the statistics format.
`server`	**`connection-queue`**, **`thread-pool`**, **`profile`**, **`process`**, **`virtual-server`**	Describes a server instance.

Table 2-7 iwsstats.xml

Element/Attribute	Subelements or Values	Description
id		The server instance ID (for example `https-www.iplanet.com`).
versionServer		A string describing the iPlanet Web Server version (for example `iPlanet-WebServer-Enterprise/6.0 B1-12/20/2000 13:56 (SunOS DOMESTIC)`).
timeStarted	A number of seconds after 00:00:00 1/1/1970	The time this server instance was started.
secondsRunning		The number of seconds since this server instance started.
ticksPerSecond		The number of ticks in a second. This value is system-dependent.
maxProcs		The maximum number of processes.
maxThreads		The maximum number of request processing threads.
maxVirtualServers		The maximum number of virtual servers tracked.
flagProfilingEnabled	`0` (off), `1` (on)	Indicates whether NSAPI performance profiling is enabled (on).
flagVirtualServer Overflow	`0` (no), `1` (yes)	Indicates whether more than `maxVirtualServers` are configured (yes). If this attribute is set to `1`, statistics are not being tracked for all virtual servers.
connection-queue	(none)	Describes a connection queue (the queue in which requests are enqueued prior to being serviced). There is only one connection queue in iPlanet Web Server 6.0. Subsequent versions may introduce multiple connection queues.
id		The connection queue ID.
thread-pool	(none)	Describes a thread pool as defined in the `magnus.conf` file.
id		The thread pool ID.
name		The symbolic name of the thread pool.

Table 2-7 iwsstats.xml

Element/Attribute	Subelements or Values	Description
profile	(none)	Describes an NSAPI performance profile bucket as defined in the magnus.conf file.
id		The NSAPI performance profile bucket ID.
name		The symbolic name of the NSAPI performance profile bucket.
description		The description of the NSAPI performance profile bucket.
process	**connection-queue-bucket**, **thread-pool-bucket**, **dns-bucket**, **keepalive-bucket**, **cache-bucket**, **thread**	Describes a single server process within a server instance.
pid		The operating system process identifier that uniquely identifies this process.
mode	unknown, active	Displays active when this process is active.
timeStarted	A number of seconds after 00:00:00 1/1/1970	The time this process was started.
countConfigurations		The number of times a configuration has been loaded, or 0 if this information is not available.
connection-queue-bucket	(none)	Tracks statistics pertaining to a specific connection-queue.
connection-queue		The ID of a connection-queue element.
countTotalConnections		The total number of new connections that have been accepted.
countQueued		The number of connections currently enqueued.
peakQueued		The largest number of connections that have been in the queue simultaneously.
maxQueued		The maximum number of connections that can be in the queue.
countOverflows		The number of times the queue has been too full to accommodate a connection.

Table 2-7 iwsstats.xml

Element/Attribute	Subelements or Values	Description
countTotalQueued		The total number of connections that have been queued. A given connection may be queued multiple times, so countTotalQueued may be greater than or equal to countTotalConnections.
ticksTotalQueued	A tick is a system-dependent unit of time; see ticksPerSecond	The total number of ticks connections have spent in the queue.
thread-pool-bucket	(none)	Tracks statistics pertaining to a specific thread-pool.
thread-pool		The ID of a thread-pool element.
countThreadsIdle		The number of request processing threads currently idle.
countThreads		The number of request processing threads.
maxThreads		The maximum number of request processing threads that can exist concurrently.
countQueued		The number of requests queued for processing by this thread pool.
peakQueued		The largest number of requests that have been in the queue simultaneously.
maxQueued		The maximum number of requests that can be in the queue.
dns-bucket	(none)	Tracks DNS (Domain Name System) statistics.
flagCacheEnabled	0 (off), 1 (on)	Indicates whether the DNS cache is enabled (on).
countCacheEntries		The number of DNS entries presently in the cache.
maxCacheEntries		The maximum number of DNS entries the cache can accommodate.
countCacheHits		The number of times a DNS cache lookup has succeeded.
countCacheMisses		The number of times a DNS cache lookup has failed.

Table 2-7 iwsstats.xml

Element/Attribute	Subelements or Values	Description
flagAsyncEnabled	0 (off), 1 (on)	Indicates whether asynchronous DNS lookups are enabled (on).
countAsyncNameLookups		The total number of asynchronous DNS name lookups performed.
countAsyncAddrLookups		The total number of asynchronous DNS address lookups performed.
countAsyncLookups InProgress		The number of asynchronous DNS lookups currently in progress.
keepalive-bucket	(none)	Tracks keepalive (persistent connection) statistics.
countConnections		The number of connections currently in keepalive mode.
maxConnections		The maximum number of simultaneous keepalive connections.
countHits		The total number of times connections in keepalive mode have subsequently made a valid request.
countFlushes		The number of times keepalive connections have been closed by the server.
secondsTimeout		The number of seconds before the server closes an idle keepalive connection.
cache-bucket	(none)	Tracks file cache (NSFC) statistics.
flagEnabled	0 (off), 1 (on)	Indicates whether the file cache is enabled (on).
secondsMaxAge	Number of seconds	The maximum age of a file cache entry.
countEntries		The number of entries currently in the file cache.
maxEntries		The maximum number of cache entries the file cache can accommodate simultaneously.
countOpenEntries		The number of entries associated with an open file.
maxOpenEntries		The maximum number of cache entries associated with an open file that the file cache can accommodate simultaneously.

Table 2-7 iwsstats.xml

Element/Attribute	Subelements or Values	Description
sizeHeapCache	Number of bytes	The amount of heap used by cached file content.
maxHeapCacheSize	Number of bytes	The maximum amount of heap the file cache uses for cached file content.
sizeMmapCache	Number of bytes	The amount of address space used by memory mapped file content.
maxMmapCacheSize	Number of bytes	The maximum amount of address space that the file cache uses for memory mapped file content.
countHits		The number of times a cache entry lookup has succeeded.
countMisses		The number of times a cache entry lookup has failed.
countInfoHits		The number of times a file information lookup has succeeded.
countInfoMisses		The number of times a file information lookup has failed.
countContentHits		The the number of times a file content lookup has succeeded.
countContentMisses		The the number of times a file content lookup has failed.
thread	**request-bucket**, **profile-bucket**	Describes a request processing thread.
mode	unknown, idle, DNS, request, processing, response, updating	The thread's last known status.
timeStarted	A number of seconds after 00:00:00 1/1/1970	The time this thread was started.
connection-queue		The ID of the connection-queue the thread is servicing.
virtual-server		The ID of the virtual-server the thread most recently serviced.
virtual-server	**request-bucket**, **profile-bucket**	Describes a virtual server.
id		The virtual server ID.

Table 2-7 iwsstats.xml

Element/Attribute	Subelements or Values	Description
mode	unknown, active	Displays active when this virtual server is active.
hosts		The software virtual server hostnames serviced by this virtual server (for example www.foo.com foo.com foo.isp.com).
interfaces		The interfaces (listen sockets) the virtual server is configured for (for example 192.168.1.2:80 192.168.1.2:443).
request-bucket	(none)	Tracks request-related statistics.
method		The method (for example GET) of the most recently serviced request.
uri		The URI (for example /index.html) of the most recently serviced request.
countRequests		The number of requests serviced.
countBytesReceived		The number of bytes received, or 0 if this information is not available.
countBytesTransmitted		The number of bytes transmitted, or 0 if this information is not available.
rateBytesTransmitted	Bytes per second	The rate at which data was transmitted over some server-defined interval, or 0 if this information is not available.
countOpenConnections		The number of open connections, or 0 if this information is not available.
count2xx		The number of 200-level responses sent.
count3xx		The number of 300-level responses sent.
count4xx		The number of 400-level responses sent.
count5xx		The number of 500-level responses sent.
countOther		The number of responses sent that were not 200, 300, 400, or 500 level.
count200		The number of 200 responses sent.
count302		The number of 302 responses sent.
count304		The number of 304 responses sent.
count400		The number of 400 responses sent.

Table 2-7 iwsstats.xml

Element/Attribute	Subelements or Values	Description
count401		The number of 401 responses sent.
count403		The number of 403 responses sent.
count404		The number of 404 responses sent.
count503		The number of 503 responses sent.
profile-bucket	(none)	Tracks statistics pertaining to a `profile` element.
profile		The ID of a `profile` element.
countCalls		The number of calls to NSAPI SAFs.
countRequests		The number of requests processed.
ticksDispatch	A tick is a system-dependent unit of time; see `ticksPerSecond`	The number of ticks spent dispatching requests.
ticksFunction	A tick is a system-dependent unit of time; see `ticksPerSecond`	The number of ticks spent in NSAPI SAFs.

jvm12.conf

Purpose

Allows you to change Java Virtual Machine settings.

Location

server_root`/https-admserv/config`

server_root`/https-admserv/conf_bk`

server_root`/https-`*server_id*`/config`

server_root`/https-`*server_id*`/conf_bk`

Syntax

```
[JVMConfig]
setting=value
...
```

See Also

Programmer's Guide to Servlets for iPlanet Web Server

Table 2-8 jvm12.conf

Setting	Allowed Values	Default Value	Description
variable	Any JVM environment variable		A JVM environment variable can be included in `jvm.conf` and given a value, for example (all on one line): `org.omg.CORBA.ORBClass= com.inprise.vbroker.orb .ORB`
`jvm.minHeapSize`		`1048576` (1 MB)	The minimum heap size allocated to Java. For Solaris, change this value to `3145278` (3 MB). For HPUX, change this value to `4194304` (4 MB). For all other operating systems, 1 MB is ideal.
`jvm.maxHeapSize`		`16777216` (16 MB)	The maximum heap size allocated to Java.
`jvm.enableClassGC`	`0` (off), `1` (on)	`0`	Enables or disables class garbage collection.
`jvm.verboseMode`	`0` (off), `1` (on)	`0`	Enables or disables JVM verbose mode. If on, the JVM logs a commentary on what it is doing, such as loading classes. The commentary appears in the error log.
`jvm.enableDebug`	`0` (off), `1` (on)	`0`	Enables or disables JVM remote debugging.
`jvm.printErrors`	`0` (off), `1` (logs to log file), `2` (logs to `stderr`)	`0`	Enables or disables reporting of errors through `vfprintf`.
`jvm.option`			Allows you to set vendor JVM options.

Table 2-8 jvm12.conf

Setting	Allowed Values	Default Value	Description
`jvm.profiler`			Specifies the profiler. If you use the `optimizeit` profiler from Intuitive Systems, you must also set the `OPTIDIR` setting.
`jvm.disableThreadRecycling`	`0` (off), `1` (on)	`0`	Enables or disables thread recycling. If on, the server always creates a global scope thread to execute servlets. Otherwise a global scope thread is created only when the request handling thread is not in the global scope.
`jvm.serializeAttach`	`0` (off), `1` (on)	`0`	If on, threads that attach to the JVM are serialized. By default (if off), threads can attach to the JVM in parallel.
`jvm.stickyAttach`	`0` (off), `1` (on)	`0`	Setting the value of this parameter to 1 causes threads to remember that they are attached to the JVM.
`jvm.trace`		`5`	Determines the trace level. For servlet and JSP debugging, the recommended level is `7`. Level 5 displays servlet engine messages. Level 6 displays servlet and JSP engine messages. Level 7 displays these and other exceptions in the browser.
`jvm.allowExit`	`0` (off), `1` (on)	`0`	Enables or disables exit from the process.
`java.compiler`		`NONE`	Specifies the Java compiler. See your JVM documentation for options that turn the JIT (just in time) compiler on and off. This should be set to `NONE` when `jvm.enableDebug` is on.
`OPTITDIR`	A path	*	Specifies the path to the profiler if the profiler is `optimizeit`.

Table 2-8 jvm12.conf

Setting	Allowed Values	Default Value	Description
`nes.jsp.enabledebug`	`0` (off), `1` (on)	`1`	Enables or disables verbose JSP compilation tracing.
`jvm.include.CLASSPATH`	`0` (off), `1` (on)	`1`	Specifies whether to include the `CLASSPATH` environment variable value in the `jvm.classpath` setting.
`nes.jsp.forkjavac`	`0` (off), `1` (on)	`0`	If on, Java compilation of JSPs runs in a separate process.
`jvm.serializeFirstRequest`	`0` (off), `1` (on)	`1` for Linux, AIX, and Compaq (DEC); `0` for other platforms	If on, ensures that only one request thread loads and constructs a servlet object. Once a servlet is loaded and initialized, new requests to the same servlet happen in parallel. This setting must be on for Linux, AIX, and Compaq (DEC).
`jvm.classpath`	A path with forward slashes only		Specifies the path(s) to JAR files dependent on the JVM. Enter additional classpath values as needed.

* *N*`:/App/IntuitiveSystems/OptimizeIt30D`, where *N* is the drive on which OptimizeIt is installed.

magnus.conf

Purpose

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

Location

server_root`/https-admserv/config`

server_root`/https-admserv/conf_bk`

server_root`/https-`*server_id*`/config`

server_root`/https-`*server_id*`/conf_bk`

Syntax

`Init` functions have the following syntax:

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

In Table 2-9, functions are in bold to distinguish them from parameters.

Directives have the following syntax:

directive value

See Also

NSAPI Programmer's Guide for iPlanet Web Server, Chapter 7

Appendix A, "Configuration Changes Between iWS 4.x and 6.0"

Init Functions

Table 2-9 magnus.conf Init functions

Function/Parameter	Allowed Values	Default Value	Description
`cindex-init`			Changes the default characteristics for fancy indexing.
`opts`	`s`	(None)	(optional) is a string of letters specifying the options to activate. Currently there is only one possible option: • `s` tells the server to scan each HTML file in the directory being indexed for the contents of the HTML `<TITLE>` tag to display in the description field. The `<TITLE>` tag must be within the first 255 characters of the file.

Table 2-9 magnus.conf Init functions

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

Table 2-9 magnus.conf Init functions

Function/Parameter	Allowed Values	Default Value	Description
`define-perf-bucket`			Creates a performance bucket, which you can use to measure the performance of SAFs in `obj.conf` (see "The bucket Parameter," on page 72). This function works only if the `perf-init` function is enabled.
`name`			A name for the bucket, for example `cgi-bucket`.
`description`			A description of what the bucket measures, for example `CGI Stats`.
`dns-cache-init`			Configures DNS caching.
`cache-size`	`32` to `32768` (32K)	`1024`	(optional) Specifies how many entries are contained in the cache.
`expire`	`1` to `31536000` seconds (1 year)	`1200` seconds (20 minutes)	(optional) specifies how long (in seconds) it takes for a cache entry to expire.
`flex-init`			Initializes the flexible logging system.
logFileName	A path or file name		The full path to the log file or a file name relative to the server's `logs` directory. In this example, the log file name is `access` and the path is `/logdir/access`: `access="/logdir/access"`
`format.`*logFileName*			Specifies the format of each log entry in the log file. See the *NSAPI Programmer's Guide for iPlanet Web Server* for more information.
`buffer-size`	Number of bytes	`8192`	Specifies the size of the global log buffer.

Table 2-9 magnus.conf Init functions

Function/Parameter	Allowed Values	Default Value	Description
num-buffers		1000	Specifies the maximum number of logging buffers to use.
flex-rotate-init			Enables rotation for logs.
rotate-start	A 4-digit string indicating the time in 24-hour format		Indicates the time to start rotation. For example, 0900 indicates 9 am while 1800 indicates 9 pm.
rotate-interval	Number of minutes		Indicates the number of minutes to elapse between each log rotation.
rotate-access	yes, no	yes	(optional) determines whether common-log, flex-log, and record-useragent logs are rotated.
rotate-error	yes, no	yes	(optional) determines whether error logs are rotated.
rotate-callback	A path		(optional) specifies the file name of a user-supplied program to execute following log file rotation. The program is passed the post-rotation name of the rotated log file as its parameter.
init-cgi			Changes the default settings for CGI programs.
timeout	Number of seconds	300	(optional) specifies how many seconds the server waits for CGI output before terminating the script.

Table 2-9 magnus.conf Init functions

Function/Parameter	Allowed Values	Default Value	Description
`cgistub-path`			(optional) specifies the path to the CGI stub binary. If not specified, iPlanet Web Server looks in the following directories, in the following order, relative to the server instance's `config` directory: `../private/Cgistub`, then `../../bin/https/bin/Cgistub`. For information about installing an suid Cgistub, see Chapter 1, "Overview."
env-variable			(optional) specifies the name and value for an environment variable that the server places into the environment for the CGI.
`init-clf`			Initializes the Common Log subsystem.
logFileName	A path or file name		Specifies either the full path to the log file or a file name relative to the server's `logs` directory.
`init-uhome`			Loads user home directory information.
`pwfile`			(optional) specifies the full file system path to a file other than `/etc/passwd`. If not provided, the default Unix path (`/etc/passwd`) is used.
`load-modules`			Loads shared libraries into the server.
`shlib`			Specifies either the full path to the shared library or dynamic link library or a file name relative to the server configuration directory.

Table 2-9 magnus.conf Init functions

Function/Parameter	Allowed Values	Default Value	Description
funcs	A comma separated list with no spaces		A list of the names of the functions in the shared library or dynamic link library to be made available for use by other `Init` or `Service` directives. The dash (-) character may be used in place of the underscore (_) character in function names.
NativeThread	`yes`, `no`	`yes`	(optional) specifies which threading model to use. `no` causes the routines in the library to use user-level threading. `yes` enables kernel-level threading.
pool			The name of a custom thread pool, as specified in `thread-pool-init`.
nt-console-init			Enables the NT console, which is the command-line shell that displays standard output and error streams.
stderr	`console`		Directs error messages to the NT console.
stdout	`console`		Directs output to the NT console.
perf-init			Enables system performance measurement via performance buckets.
disable	`true`, `false`	`true`	Disables the function when `true`.
pool-init			Configures pooled memory allocation.
free-size	`1048576` bytes or less		(optional) maximum size in bytes of free block list.
disable	`true`, `false`	`false`	(optional) flag to disable the use of pooled memory if `true`.

Table 2-9 magnus.conf Init functions

Function/Parameter	Allowed Values	Default Value	Description
`register-http-method`			Lets you extend the HTTP protocol by registering new HTTP methods.
`methods`	A comma separated list		Names of the methods you are registering.
`stats-init`			Enables reporting of performance statistics in XML format.
`profiling`	`yes`, `no`	`no`	Enables NSAPI performance profiling using buckets. This can also be enabled through `perf-init`.
`update-interval`	`1` or greater	`5`	The period in seconds between statistics updates within the server.
`virtual-servers`	`1` or greater	`1000`	The maximum number of virtual servers for which statistics are tracked. This number should be set higher than the number of virtual servers configured.
`thread-pool-init`			Configures an additional thread pool.
`name`			Name of the thread pool.
`maxthreads`			Maximum number of threads in the pool.
`minthreads`			Minimum number of threads in the pool.
`queueSize`	Number of bytes		Size of the queue for the pool.
`stackSize`	Number of bytes		Stack size of each thread in the native (kernel) thread pool.

Directives

Table 2-10 magnus.conf directives

Directive	Allowed Values	Default Value	Description
ACLCacheLifetime	Any number of seconds	120	Determines the number of seconds before cache entries expire. Each time an entry in the cache is referenced, its age is calculated and checked against ACLCacheLifetime. The entry is not used if its age is greater than or equal to the ACLCacheLifetime. If this value is set to 0, the cache is turned off.
ACLUserCacheSize		200	Determines the number of users in the User Cache.
ACLGroupCacheSize		4	Determines how many group IDs can be cached for a single UID/cache entry.
AdminLanguage	en (English), fr (French), de (German), ja (Japanese)	en	Specifies the language for the Server Manager.
AsyncDNS	on, off	off	Specifies whether asynchronous DNS is allowed.
CGIExpirationTimeout	Any number of seconds	300 (5 minutes) recommended	Specifies the maximum time in seconds that CGI processes are allowed to run before being killed.
CGIStubIdleTimeout	Any number of seconds	30	Causes the server to kill any CGIStub processes that have been idle for the number of seconds set by this directive. Once the number of processes is at MinCGIStubs, the server does not kill any more processes.
CGIWaitPid	on, off	on	(Unix only) makes the action for the SIGCHLD signal the system default action for the signal. Makes the SHTML engine wait explicitly on its exec cmd child processes.

Table 2-10 magnus.conf directives

Directive	Allowed Values	Default Value	Description
ChildRestartCallback	on, off, yes, no, true, false	no	Forces the callback of NSAPI functions that were registered using the daemon_atrestart function when the server is restarting or shutting down.
ChunkedRequestBufferSize	Any number of bytes	8192	Determines the default buffer size for "un-chunking" request data.
ChunkedRequestTimeout	Any number of seconds	60 (1 minute).	Determines the default timeout for "un-chunking" request data.
ClientLanguage	en (English), fr (French), de (German), ja (Japanese)	en	Specifies the language for client messages (such as File Not Found).
ConnQueueSize	Any number of connections	5000	Specifies the number of outstanding (yet to be serviced) connections that the web server can have.
DefaultCharSet	A valid character set name	iso-8859-1	Specifies the default character set for the server. The default language is used for both the client responses and administration.
DefaultLanguage	en (English), fr (French), de (German), ja (Japanese)	en	Specifies the default language for the server. The default language is used for both the client responses and administration.
DNS	on, off	on	Specifies whether the server performs DNS lookups on clients that access the server.
ErrorLog	A path	(none)	Specifies the directory where the server logs its errors.
ErrorLogDateFormat	See the manual page for the C library function strftime	%d/%b/%Y:%H:%M:%S	The date format for the error log.

Table 2-10 magnus.conf directives

Directive	Allowed Values	Default Value	Description
`ExtraPath`	A path	(none)	Appends the specified directory name to the `PATH` environment variable. This is used for configuring Java on Windows NT. There is no default value; you must specify a value.
`HeaderBufferSize`	Any number of bytes	`8192` (8 KB)	The size (in bytes) of the buffer used by each of the request processing threads for reading the request data from the client. The maximum number of request processing threads is controlled by the `RqThrottle` setting.
`HTTPVersion`	*m.n*; *m* is the major version number and *n* the minor version number	`1.1`	The current HTTP version used by the server.
`IOTimeout`	Any number of seconds	`30` for servers that don't use hardware encryption devices and `300` for those that do	Specifies the number of seconds the server waits for data to arrive from the client. If data does not arrive before the timeout expires then the connection is closed.
`KeepAliveThreads`	Any number of threads	`1`	Specifies the number of threads in the keep-alive subsystem. It is recommended that this number be a small multiple of the number of processors on the system.
`KeepAliveTimeout`	`300` seconds maximum	`30`	Determines the maximum time that the server holds open an HTTP Keep-Alive connection or a persistent connection between the client and the server.
`KernelThreads`	`0` (off), `1` (on)	`0` (off)	If on, ensures that the server uses only kernel-level threads, not user-level threads. If off, uses only user-level threads.

Table 2-10 magnus.conf directives

Directive	Allowed Values	Default Value	Description
ListenQ	Ranges are platform-specific	4096 (AIX), 200 (NT), 128 (all others)	Defines the number of incoming connections for a server socket.
LogFlushInterval	Any number of seconds	30	Determines the log flush interval, in seconds, of the log flush thread.
LogVerbose	on, off	off	If on, logs all server messages including those that are not logged by default.
LogVsId	on, off	off	Determines whether virtual server IDs are displayed in the error log. You should enable LogVsId when multiple virtual servers share the same log file.
MaxCGIStubs	Any number of CGI stubs	10	Controls the maximum number of CGIStub processes the server can spawn. This is the maximum concurrent CGIStub processes in execution, not the maximum number of pending requests.
MaxKeepAliveConnections	0 - 32768	256	Specifies the maximum number of Keep-Alive and persistent connections that the server can have open simultaneously.
MaxProcs	Any number of processes	1	(Unix only) Specifies the maximum number of processes that the server can have running simultaneously.
MaxRqHeaders	0 - 32	32	Specifies the maximum number of header lines in a request.
MinCGIStubs	Any number less than MaxCGIStubs	2	Controls the number of processes that are started by default.
MtaHost	A valid e-mail address	(none)	Specifies the SMTP mail server used by the server's agents. This value must be specified before reports can be sent to a mailing address.
NativePoolMaxThreads	Any number of threads	128	Determines the maximum number of threads in the native (kernel) thread pool.

Table 2-10 magnus.conf directives

Directive	Allowed Values	Default Value	Description
NativePoolMinThreads	Any number of threads	1	Determines the minimum number of threads in the native (kernel) thread pool.
NativePoolQueueSize	Any nonnegative number	0	Determines the number of threads that can wait in the queue for the thread pool.
NativePoolStackSize	Any nonnegative number	0	Determines the stack size of each thread in the native (kernel) thread pool.
NetSiteRoot	A path	(none)	Specifies the absolute pathname to the top-level directory under which server instances can be found. There is no default value; you must specify a value.
PidLog	A valid path to a file	(none)	Specifies a file in which to record the process ID (pid) of the base server process.
PostThreadsEarly	1 (on), 0 (off)	0 (off)	If on, checks whether the minimum number of threads are available at a socket after accepting a connection but before sending the response to the request.
RcvBufSize	Range is platform-specific	0 (uses platform-specific default)	Controls the size of the receive buffer at the server's sockets.
RqThrottle	Any number of requests	512	Specifies the maximum number of simultaneous request processing threads that the server can handle simultaneously per socket.
RqThrottleMin	Any number less than RqThrottle	48	Specifies the number of request processing threads that are created when the server is started. As the load on the server increases, more request processing threads are created (up to a maximum of RqThrottle threads).

Table 2-10 magnus.conf directives

Directive	Allowed Values	Default Value	Description
`Security`	`on`, `off`	`off`	Globally enables or disables SSL by making certificates available to the server instance. Must be `on` for virtual servers to use SSL.
`ServerConfigurationFile`	A file name	`server.xml`	The name of the file that specifies virtual servers.
`ServerID`	A string	(none)	Specifies the server ID, such as `https-boots.mcom.com`.
`#ServerRoot`	A path	(none)	Specifies the server root. This directive is set during installation and is commented out. Unlike other directives, the server expects this directive to start with #. Do not change this directive.
`SndBufSize`	Range is platform-specific	`0` (uses platform-specific default)	Controls the size of the send buffer at the server's sockets.
`SSL3SessionTimeout`	`5 - 86400`	`86400` (24 hours).	The number of seconds until a cached SSL3 session becomes invalid.
`SSLCacheEntries`	A non-negative integer	`10000` (used if `0` is specified)	Specifies the number of SSL sessions that can be cached. There is no upper limit.
`SSLClientAuthDataLimit`	Number of Bytes	`1048576` (1MB)	Specifies the maximum amount of application data that is buffered during the client certificate handshake phase.
`SSLClientAuthTimeout`	Any number of seconds	`60`	Specifies the number of seconds after which the client certificate handshake phase times out.
`SSLSessionTimeout`	`5 - 100`	`100`	Specifies the number of seconds until a cached SSL2 session becomes invalid.

Table 2-10 magnus.conf directives

Directive	Allowed Values	Default Value	Description
`StackSize`	Number of Bytes	The most favorable machine-specific stack size.	Determines the maximum stack size for each request handling thread.
`StrictHttpHeaders`	`on`, `off`	`off`	If `on`, rejects connections that include inappropriately duplicated headers.
`TempDir`	A path	`/tmp` (Unix) `TEMP` (environment variable for Windows NT)	Specifies the directory the server uses for its temporary files. On Unix, this directory should be owned by, and writable by, the user the server runs as.
`TempDirSecurity`	`on`, `off`	`on`	Determines whether the server checks if the `TempDir` directory is secure. On Unix, specifying `TempDirSecurity off` allows the server to use `/tmp` as a temporary directory.
`TerminateTimeout`	Any number of seconds	`30`	Specifies the time in seconds that the server waits for all existing connections to terminate before it shuts down.
`ThreadIncrement`	Any number of threads	`10`	The number of additional or new request processing threads created to handle an increase in the load on the server.
`Umask`	A standard UNIX umask value	(none)	Unix only: Specifies the umask value used by the NSAPI functions `System_fopenWA()` and `System_fopenRW()` to open files in different modes.
`UseNativePoll`	`1` (on), `0` (off)	`1` (on)	Uses a platform-specific poll interface when set to `1` (on). Uses the NSPR poll interface in the KeepAlive subsystem when set to `0` (off).

Table 2-10 magnus.conf directives

Directive	Allowed Values	Default Value	Description
`UseOutputStreamSize`	Any number of bytes	`8192` (8 KB)	Determines the default output stream buffer size for the `net_read` and `netbuf_grab` NSAPI functions.
`User`	A login name, 8 characters or less	(none)	(Windows NT) specifies the user account the server runs with, allowing you to restrict or enable system features for the server. (Unix) if the server is started by the `superuser` or `root` user, the server binds to the Port you specify and then switches its user ID to the user account specified with the User directive. This directive is ignored if the server isn't started as `root`.
`WincgiTimeout`	Any number of seconds	`60`	WinCGI processes that take longer than this value are terminated when this timeout expires.

mime.types

Purpose

Maps standard MIME types to file extensions. Each virtual server can have its own `mime.types` file.

Location

server_root`/https-admserv/config`

server_root`/https-admserv/conf_bk`

server_root`/https-`*server_id*`/config`

server_root`/https-`*server_id*`/conf_bk`

server_root`/bin/https/install/misc`

Syntax

`type=`*type/subtype* `exts=`*ext1*`,`*ext2*`,...`
`...`
`enc=`*subtype* `exts=`*ext1*`,`*ext2*`,...`
`...`

See Also

NSAPI Programmer's Guide for iPlanet Web Server

Table 2-11 mime.types

Directive	Allowed Values	Description
type	`application`, `image`, `text`, `video`, `audio`, `perf`, `x-world`, `x-conference`, `magnus-internal`	The basic type of content.
subtype		The more specific type of content. For example, in `text/html`, the subtype is `html`.
ext1, *ext2*, ...	A comma separated list of file extensions	The file extension(s) for the type. For example, for `text/html`, the file extensions are `htm`, `html`.

ns-cron.conf

Purpose

Activates or deactivates the `cron.conf` file.

Location

server_root`/https-admserv/config`

Contents

```
ConfFile server_root/https-admserv/config/cron.conf
Dir /tmp
Status on
```

Table 2-12 ns-cron.conf

Directive	Allowed Values	Default Value	Description
`ConfFile`	A path		The location of the `cron.conf` file.
`Dir`	A path		The location of a temporary directory.
`Status`	`on`, `off`	`on`	The status of the `cron.conf` file: `on` is activated, `off` is deactivated.

nsfc.conf

Purpose

Sets file cache parameters. This file is present only if file cache parameters have been changed from their defaults.

Location

server_root/`https-admserv/config`

Syntax

parameter=*value*

See Also

Performance Tuning, Sizing, and Scaling Guide for iPlanet Web Server

Table 2-13 nsfc.conf

Parameter	Allowed Values	Default Value	Description
`FileCacheEnable`	on, `off`	on	Enables the file cache.
`CacheFileContent`	on, `off`	on	Enables caching of file contents as well as file information for files smaller than `MediumFileSizeLimit` (smaller than `SmallFileSizeLimit` if `TransmitFiles` is on).
`MaxAge`	Number of seconds	`30`	The maximum age of a valid cache entry. This setting controls how long cached information is used once a file has been cached. An entry older than `MaxAge` is replaced by a new entry for the same file.
`MediumFileSizeLimit`	Limited by available memory	`537600` (525K)	(Unix only) Maximum size of a file that can be cached as a memory-mapped file (if `TransmitFiles` is `off`).
`MediumFileSpace`	Limited by available memory	`10485760` (10 M)	Total size of all files that are cached as memory-mapped files (if `TransmitFiles` is `off`).
`SmallFileSizeLimit`	Limited by available memory	`2048` (2K)	(Unix only) Maximum size of a file that can be read into memory.

Table 2-13 nsfc.conf

Parameter	Allowed Values	Default Value	Description
SmallFileSpace	Limited by available memory	1048576 (Unix, 1 M), 0 (NT)	Total size of all files that are read into memory.
TransmitFiles	on, off	on (NT), off (Unix)	Enables use of the TransmitFile system call. Not supported on IRIX, Compaq, Solaris, or Linux.
MaxFiles		1024	Maximum number of files in the file cache.
HashInitSize	Limited by available memory	0	Initial number of hash buckets. If 0, the number of hash buckets is dynamically determined as 2 * MaxFiles + 1.
CopyFiles	on, off	on	(NT only) Prevents sharing violations by copying files to a temporary directory.
TempDir	A path	*system_temp*/*server_id*	Specifies a temporary directory for the file cache if CopyFiles is on.

obj.conf

Purpose

Determines client responses to requests. Each virtual server can have its own obj.conf file.

Location

server_root/https-admserv/config

server_root/https-admserv/conf_bk

server_root/https-*server_id*/config

server_root/https-*server_id*/conf_bk

Syntax

directive fn=*function param1*="*value1*" ...*paramN*="*valueN*"

An object tag's directives are executed only if a NameTrans directive redirects the flow of control to the tag via a name/name or root/ppath match. An object tag follows this syntax:

```
NameTrans fn=function name="name"|root="path"

<object name="name"|ppath="path">
directive1
directive2
...
</object>
```

NOTE In iPlanet Web Server 6.0, `Init` directives have been moved to the `magnus.conf` file.

In Table 2-15 through Table 2-21, functions are in bold to distinguish them from parameters.

See Also

NSAPI Programmer's Guide for iPlanet Web Server, Chapters 2-3.

For how to create your own functions, see *NSAPI Programmer's Guide for iPlanet Web Server*, Chapters 4-6.

Appendix A, "Configuration Changes Between iWS 4.x and 6.0"

Table 2-14 obj.conf

Directive	Description
`AuthTrans`	Verifies any authorization information (normally sent in the `Authorization` header) provided in the HTTP request and translates it into a user or a group.
`NameTrans`	Translates the URL specified in the request from a logical URL to a physical file system path for the requested resource. This may also result in redirection to another site.
`PathCheck`	Performs tests on the physical path determined by the `NameTrans` step. In general, these tests determine whether the path is valid and whether the client is allowed to access the requested resource.
`ObjectType`	Determines the MIME (Multi-purpose Internet Mail Encoding) type of the requested resource.
`Service`	Generates and sends the response to the client. This involves setting the HTTP result status, setting up response headers (such as `content-type` and `content-length`), and generating and sending the response data.
`AddLog`	Adds an entry to a log file to record information about the transaction.
`Error`	Handles an HTTP error resulting from execution of the previous directive. Typically the server handles an error by sending a custom HTML document to the user describing the problem and possible solutions.

The bucket Parameter

The following performance buckets are predefined in iPlanet Web Server:

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

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

You can measure the performance of any SAF in `obj.conf` by adding a `bucket=`*bucket-name* parameter to the function, for example `bucket=cache-bucket`. Because `bucket` is a parameter of every `obj.conf` function, for brevity it is not listed in the following tables.

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

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

For more information about performance buckets, see the *Performance Tuning, Sizing, and Scaling Guide for iPlanet Web Server*.

AuthTrans Functions

Table 2-15 obj.conf AuthTrans functions

Function/Parameter	Allowed Values	Default Value	Description
`basic-auth`			Calls a custom function to verify user name and password. Optionally determines the user's group.
`auth-type`	`basic`	`basic`	Specifies the type of authorization to be used.
`userdb`			(optional) specifies the full path and file name of the user database to be used for user verification. This parameter will be passed to the user function.
`userfn`			The name of the user custom function to verify authorization. This function must have been previously loaded with `load-modules`.
`groupdb`			(optional) specifies the full path and file name of the user database. This parameter will be passed to the group function.

Table 2-15 obj.conf AuthTrans functions

Function/Parameter	Allowed Values	Default Value	Description
groupfn			(optional) is the name of the group custom function that must have been previously loaded with `load-modules`.
basic-ncsa			Verifies user name and password against an NCSA-style or system DBM database. Optionally determines the user's group.
auth-type	basic	basic	Specifies the type of authorization to be used.
dbm			(optional) specifies the full path and base file name of the user database in the server's native format. If you use this parameter, don't use the `userfile` parameter as well.
userfile			(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 *name:password*, where *password* is encrypted. If you use this parameter, don't use `dbm`.
grpfile			(optional) specifies the NCSA-style HTTPD group file to be used. Each line of a group file consists of *group*`:`*user1 user2 ... userN* where each user is separated by spaces.
get-sslid			Retrieves a string that is unique to the current SSL session and stores it as the `ssl-id` variable in the `Session->client` parameter block.
qos-handler			Examines the current quality of service statistics for the virtual server, virtual server class, and global server, logs the statistics, and enforces the QOS parameters by returning an error. This must be the first `AuthTrans` function configured in the `default` object in order to work properly.

NameTrans Functions

Table 2-16 obj.conf NameTrans functions

Function/Parameter	Allowed Values	Default Value	Description
assign-name			Tells the server to process directives in a named object.
from			A wildcard pattern that specifies the path to be affected.

Table 2-16 obj.conf NameTrans functions

Function/Parameter	Allowed Values	Default Value	Description
name			Specifies an additional named object in `obj.conf` whose directives will be applied to this request.
find-pathinfo-forward	The value is ignored		(optional) makes the server look for the PATHINFO forward in the path right after the ntrans-base instead of backward from the end of path as the server function `assign-name` does by default.
nostat			(optional) prevents the server from performing a stat on a specified URL whenever possible. Use `nostat` only when the path of the *virtual-path* does not exist on the system, for example, for NSAPI plugin URLs, to improve performance by avoiding unnecessary stats on those URLs.
document-root			Translates a URL into a file system path by replacing the `http://`*server-name*`/` part of the requested resource with the document root directory.
root		*server_root*`/docs`	The file system path to the server's root document directory.
home-page			Translates a request for the server's root home page (`/`) to a specific file.
path			The path and name of the home page file. If `path` starts with a slash (`/`), it is assumed to be a full path to a file.
pfx2dir			Translates any URL beginning with a given prefix to a file system directory and optionally enables directives in an additional named object.
from			The URI prefix to convert. It should not have a trailing slash (/).
dir			The local file system directory path that the prefix is converted to. It should not have a trailing slash (/).
name			(optional) specifies an additional named object in `obj.conf` whose directives will be applied to this request.
find-pathinfo-forward	The value is ignored		(optional) makes the server look for the PATHINFO forward in the path right after the ntrans-base instead of backward from the end of path as the server function `pfx2dir` does by default.

Table 2-16 obj.conf NameTrans functions

Function/Parameter	Allowed Values	Default Value	Description
`redirect`			Redirects the client to a different URL.
`from`			Specifies the prefix of the requested URI to match.
`url`			(maybe optional) specifies a complete URL to return to the client. If you use this parameter, don't use `url-prefix` (and vice-versa).
`url-prefix`			(maybe optional) the new URL prefix to return to the client. The `from` prefix is simply replaced by this URL prefix. If you use this parameter, don't use `url` (and vice-versa).
`escape`	`yes`, `no`	`yes`	(optional) is a flag which tells the server to `util_uri_escape` the URL before sending it.
`strip-params`			Removes embedded semicolon-delimited parameters from the path. For example, a URI of `/dir1;param1/dir2` becomes `/dir1/dir2`. If used, should be the first `NameTrans` directive listed.
`unix-home`			Translates a URL to a specified directory within a user's home directory.
`from`			The URL prefix to translate, usually "`/~`".
`subdir`			The subdirectory within the user's home directory that contains their web documents.
`pwfile`			(optional) the 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.

PathCheck Functions

Table 2-17 obj.conf PathCheck functions

Function/Parameter	Allowed Values	Default Value	Description
`cert2user`			Determines the authorized user from the client certificate.

Table 2-17 obj.conf PathCheck functions

Function/Parameter	Allowed Values	Default Value	Description
`userdb`			Names the user database from which to obtain the certificate.
`makefrombasic`			Tells the function to establish a certificate-to-user mapping.
`require`	`0` or `1`	`1`	Governs the return value if the certificate cannot be mapped to a user name. If `require=0`, the function returns `REQ_NOACTION`, allowing the processing of the request to continue. If `require` is not `0`, the function returns `REQ_ABORTED` and sets the protocol status to `403 FORBIDDEN`.
`method`			Specifies a wildcard pattern for the HTTP methods for which this function will be applied. If `method` is absent, the function is applied for any method.
`check-acl`			Checks an access control list for authorization.
`acl`			The name of an Access Control List.
`shexp`			(optional) a wildcard pattern that specifies the path for which to apply the ACL.
`bong-file`			(optional) the path name for a file that will be sent if this ACL denies access.
`deny-existence`			Indicates that a resource was not found.
`path`			(optional) a 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 “not found” message. It is a full file-system path.
`find-index`			Locates a default file when a directory is requested.
`index-names`	A comma separated list		A 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.

Table 2-17 obj.conf PathCheck functions

Function/Parameter	Allowed Values	Default Value	Description
`find-links`			Denies access to directories with certain file system links
`disable`	`h`, `s`, `o`		A character string of links to disable: • `h` is hard links • `s` is soft links • `o` allows symbolic links from user home directories only if the user owns the target of the link.
`dir`			The 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.
`find-pathinfo`			Locates extra path info beyond the file name for the PATH_INFO CGI environment variable.
`find-pathinfo-forward`	The value is ignored		(optional) makes the server look for the PATHINFO forward in the path right after the ntrans-base instead of backward from the end of path as the server function `find-pathinfo` does by default.
`get-client-cert`			Gets the authenticated client certificate from the SSL3 session.
`dorequest`	`0` or `1`	`0` if `dorequest` is absent	Controls whether to actually try to get the certificate, or just test for its presence. • 1 tells the function to redo the SSL3 handshake to get a client certificate, if the server does not already have the client certificate. This typically causes the client to present a dialog box to the user to select a client certificate. • 0 tells the function not to redo the SSL3 handshake if the server does not already have the client certificate.

Table 2-17 obj.conf PathCheck functions

Function/Parameter	Allowed Values	Default Value	Description
`require`	`0` or `1`	`1` if `require` is absent	Controls whether failure to get a client certificate will abort the HTTP request. • 1 tells the function to abort the HTTP request if the client certificate is not present after `dorequest` is handled. In this case, the HTTP status is set to `PROTOCOL_FORBIDDEN`, and the function returns `REQ_ABORTED`. • 0 tells the function to return `REQ_NOACTION` if the client certificate is not present after dorequest is handled.
`method`			(optional) specifies a wildcard pattern for the HTTP methods for which the function will be applied. If `method` is absent, the function is applied to all requests.
`load-config`			Finds and loads extra configuration information from a file in the requested path
`file`		`.nsconfig`	(optional) the name of the dynamic configuration file containing the access rules to be applied to the requested resource.
`disable-types`			(optional) specifies a wildcard pattern of types to disable for the base directory, such as `magnus-internal/cgi`. Requests for resources matching these types are aborted.
`descend`			(optional) if present, specifies that the server should search in subdirectories of this directory for dynamic configuration files.
`basedir`	A path	The result of translating the requested resource's URL to a physical pathname	(optional) specifies base directory. This is the highest-level directory for which requests will invoke the `load-config` function and is also the directory where the server starts searching for configuration files.
`nt-uri-clean`			Denies access to requests with unsafe path names by indicating not found.
`ntcgicheck`			Looks for a CGI file with a specified extension.

Table 2-17 obj.conf PathCheck functions

Function/Parameter	Allowed Values	Default Value	Description
extension			The replacement file extension.
require-auth			Denies access to unauthorized users or groups.
path			(optional) a wildcard local file system path on which this function should operate. If no path is provided, the function applies to all paths.
auth-type	basic	basic	The type of HTTP authorization used and must match the auth-type from the previous authorization function in AuthTrans.
realm			A string sent to the browser indicating the secure area (or realm) for which a 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, then any user authorized by the authorization function is allowed access.
auth-group			(optional) specifies a wildcard list of groups that are allowed access.
set-virtual-index			Specifies a virtual index for a directory.
virtual-index			The URI of the content generator that acts as an index for the URI the user enters.
from	A comma separated list		(optional) a list of URIs for which this `virtual-index` is applicable. If `from` is not specified, the `virtual-index` always applies.
ssl-check			Checks the secret keysize.
secret-keysize			(optional) the minimum number of bits required in the secret key.
bong-file			(optional) the name of a file (not a URI) to be served if the restriction is not met.
ssl-logout			Invalidates the current SSL session in the server's SSL session cache.
unix-uri-clean			Denies access to requests with unsafe path names by indicating not found.

ObjectType Functions

Table 2-18 obj.conf ObjectType functions

Function/Parameter	Allowed Values	Default Value	Description
`force-type`			Sets the content-type header for the response to a specific type.
`type`			(optional) the type assigned to a matching request (the `content-type` header).
`enc`			(optional) the encoding assigned to a matching request (the `content-encoding` header).
`lang`			(optional) the language assigned to a matching request (the `content-language` header).
`charset`			(optional) the character set for the `magnus-charset` parameter in `rq->srvhdrs`. If the browser sent the `Accept-charset` header or its `User-agent` is mozilla/1.1 or newer, then append "`; charset=`*charset*" to content-type, where *charset* is the value of the `magnus-charset` parameter in `rq->srvhdrs`.
`set-default-type`			Allows you to define a default `charset`, `content-encoding`, and `content-language` for the response being sent back to the client.
`enc`			(optional) the encoding assigned to a matching request (the `content-encoding` header).
`lang`			(optional) the language assigned to a matching request (the `content-language` header).
`charset`			(optional) the character set for the `magnus-charset` parameter in `rq->srvhdrs`. If the browser sent the `Accept-charset` header or its `User-agent` is mozilla/1.1 or newer, then append "`; charset=`*charset*" to content-type, where *charset* is the value of the `magnus-charset` parameter in `rq->srvhdrs`.
`shtml-hacktype`			Requests that `.htm` and `.html` files are parsed for server-parsed html commands.
`exec-hack`	The value is ignored		(Unix only, optional) if present, tells the function to change the content-type only if the execute bit is enabled.

Table 2-18 obj.conf ObjectType functions

Function/Parameter	Allowed Values	Default Value	Description
`type-by-exp`			Sets the content-type header for the response based on the requested path.
`exp`			The wildcard pattern of paths for which this function is applied.
`type`			(optional) the type assigned to a matching request (the `content-type` header).
`enc`			(optional) the encoding assigned to a matching request (the `content-encoding` header).
`lang`			(optional) the language assigned to a matching request (the `content-language` header).
`charset`			(optional) the character set for the `magnus-charset` parameter in `rq->srvhdrs`. If the browser sent the `Accept-charset` header or its `User-agent` is mozilla/1.1 or newer, then append "`; charset=`*charset*" to content-type, where *charset* is the value of the `magnus-charset` parameter in `rq->srvhdrs`.
`type-by-extension`			Sets the content-type header for the response based on the files extension and the MIME types database.

Service Functions

Table 2-19 obj.conf Service functions

Function/Parameter	Allowed Values	Default Value	Description
Common `Service` parameters			The first seven parameters listed are common to all `Service` functions. For brevity, they are listed once.
`type`			(optional) specifies a wildcard pattern of MIME types for which this function will be executed. The `magnus-internal/*` MIME types are used only to select a `Service` function to execute.

Table 2-19 obj.conf Service functions

Function/Parameter	Allowed Values	Default Value	Description
`method`			(optional) specifies a wildcard pattern of HTTP methods for which this function will be executed. Common HTTP methods are `GET`, `HEAD`, and `POST`.
`query`			(optional) specifies a wildcard pattern of query strings for which this function will be executed.
`UseOutputStreamSize`	Number of bytes	`8192`	(optional) determines the default output stream buffer size for the `net_read` and `netbuf_grab` NSAPI functions.
`flushTimer`	Number of milli-seconds	`3000`	(optional) determines the maximum time between write operations in which buffering is enabled. If the interval between subsequent write operations is greater than the `flushTimer` value for an application, further buffering is disabled.
`ChunkedRequestBufferSize`	Number of bytes	`8192`	(optional) determines the default buffer size for "un-chunking" request data.
`ChunkedRequestTimeout`	Number of seconds	`60`	(optional) determines the default timeout, in seconds, for "un-chunking" request data.
`add-footer`			Appends a footer specified by a filename or URL to a an HTML file.
`file`			(optional) The pathname to the file containing the footer. Specify either `file` or `uri`.
`uri`			(optional) A URI pointing to the resource containing the footer. Specify either `file` or `uri`.
`NSIntAbsFilePath`	`yes` or `no`	`no`	(optional) if the `file` parameter is specified, the `NSIntAbsFilePath` parameter determines whether the file name is absolute (`yes`) or relative (`no`).
`add-header`			Prepends a header specified by a filename or URL to an HTML file.

Table 2-19 obj.conf Service functions

Function/Parameter	Allowed Values	Default Value	Description
`file`			(optional) The pathname to the file containing the header. Specify either `file` or `uri`.
`uri`			(optional) A URI pointing to the resource containing the header. Specify either `file` or `uri`.
`NSIntAbsFilePath`	`yes` or `no`	`no`	(optional) if the `file` parameter is specified, the `NSIntAbsFilePath` parameter determines whether the file name is absolute (`yes`) or relative (`no`).
`append-trailer`			Appends text to the end of an HTML file.
`trailer`			The text to append to HTML documents. The string is unescaped with `util_uri_unescape` before being sent. The text can contain HTML tags and can be up to 512 characters long after unescaping and inserting the date. If you use the string `:LASTMOD:`, which is replaced by the date the file was last modified; you must also specify a time format with `timefmt`.
`timefmt`			(optional) a time format string for `:LASTMOD:`. If `timefmt` is not provided, `:LASTMOD:` is not replaced with the time. For details about time formats, see the *NSAPI Programmer's Guide for iPlanet Web Server*.
`imagemap`			Handles server-side image maps.
`index-common`			Generates a fancy list of the files and directories in a requested directory.
`header`			(optional) the path (relative to the directory being indexed) and name of a file (HTML or plain text) which is included at the beginning of the directory listing to introduce the contents of the directory.
`readme`			(optional) the path (relative to the directory being indexed) and name of a file (HTML or plain text) to append to the directory listing.

Table 2-19 obj.conf Service functions

Function/Parameter	Allowed Values	Default Value	Description
index-simple			Generates a simple list of files and directories in a requested directory.
key-toosmall			Indicates to the client that the provided certificate key size is too small to accept.
list-dir			Lists the contents of a directory. The request `method` must be `INDEX`.
make-dir			Creates a directory. The request `method` must be `MKDIR`.
query-handler			Handles the HTML ISINDEX tag.
`path`			The full path and file name of the CGI program to run.
remove-dir			Deletes an empty directory. The request `method` must be `RMDIR`.
remove-file			Deletes a file. The request `method` must be `DELETE`.
rename-file			Renames a file. The request `method` must be `MOVE`.
send-cgi			Sets up environment variables, launches a CGI program, and sends the response to the client.
`user`			(Unix only) The name of the user to execute CGI programs as.
`group`			(Unix only) The name of the group to execute CGI programs as.
`chroot`			(Unix only) The directory to chroot to before execution begins. This is relative to the `Chroot` defined in `magnus.conf`.
`dir`			(Unix only) The directory to chdir to after chroot but before execution begins.
`rlimit_as`			(Unix only) The maximum CGI program address space in bytes. You can supply both current (soft) and maximum (hard) limits, separated by a comma. The soft limit must be listed first. If only one limit is specified, both limits are set to this value.

Table 2-19 obj.conf Service functions

Function/Parameter	Allowed Values	Default Value	Description
`rlimit_core`			(Unix only) The maximum CGI program core file size. A value of 0 disables writing cores. You can supply both current (soft) and maximum (hard) limits, separated by a comma. The soft limit must be listed first. If only one limit is specified, both limits are set to this value.
`rlimit_nofile`			(Unix only) The maximum number of file descriptors for the CGI program. You can supply both current (soft) and maximum (hard) limits, separated by a comma. The soft limit must be listed first. If only one limit is specified, both limits are set to this value.
`nice`			(Unix only) Accepts an increment that determines the CGI program's priority relative to the server. Typically, the server is run with a nice value of 0 and the nice increment would be between 0 (the CGI program runs at same priority as server) and 19 (the CGI program runs at much lower priority than server).
`send-file`			Sends a local file to the client. This directive is invoked if the method of the request is `GET`, `HEAD`, or `POST`, and the type does *not* start with `magnus-internal/`.
`nocache`	The value is ignored		(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.
`send-range`			Sends a range of bytes of a file to the client.
`send-shellcgi`			(Windows NT only) Sets up environment variables, launches a shell CGI program, and sends the response to the client.
`send-wincgi`			(Windows NT only) Sets up environment variables, launches a WinCGI program, and sends the response to the client.

Table 2-19 obj.conf Service functions

Function/Parameter	Allowed Values	Default Value	Description
service-dump			Creates a performance report based on collected performance bucket data (see "The bucket Parameter," on page 72). The `mime.types` file must contain the following line: `type=perf exts=perf`. To read the report, point the browser here: `http://`*server_id*`:`*port*`/.perf`.
`type`	`perf`		Specifies the MIME type of the report.
shtml_send			Parses an HTML document, scanning for embedded commands. These commands may provide information from the server, include the contents of other files, or execute a CGI program. The `shtml_send` function is only available when the Shtml plugin (`libShtml.so` on Unix, `libShtml.dll` on Windows NT) is loaded.
`ShtmlMaxDepth`		`10`	Maximum depth of include nesting allowed.
`addCgiInitVars`	`yes`, `no`	`no`	(Unix only) If present and equal to `yes`, adds the environment variables defined in the `init-cgi` SAF to the environment of any command executed through the SHTML `exec` tag.
stats-xml			Creates a performance report in XML format. You must initialize this function using the `stats-init` function in `magnus.conf`, then use a NameTrans function to direct requests to the `stats-xml` function. The report is generated here: `http://`*server_id*`:`*port*`/stats-xml/iwsstats.xml`. The associated DTD file is here: `http://`*server_id*`:`*port*`/stats-xml/iwsstats.dtd`.
upload-file			Uploads and saves a file. The request method must be `PUT`.

AddLog Functions

Table 2-20 obj.conf AddLog functions

Function/Parameter	Allowed Values	Default Value	Description
`common-log`			Records information about the request in the common log format.
`name`			(optional) gives the name of a log file, which must have been given as a parameter to the `init-clf` Init function. If no name is given, the entry is recorded in the global log file.
`iponly`	The value is ignored		(optional) instructs the server to log the IP address of the remote client rather than looking up and logging the DNS name. This will improve performance if DNS is off in the `magnus.conf` file.
`flex-log`			Records information about the request in a flexible, configurable format.
`name`			(optional) gives the name of a log file, which must have been given as a parameter to the `init-clf` Init function. If no name is given, the entry is recorded in the global log file.
`iponly`	The value is ignored		(optional) instructs the server to log the IP address of the remote client rather than looking up and logging the DNS name. This will improve performance if DNS is off in the `magnus.conf` file.
`record-useragent`			Records the client's IP address and user-agent header.
`name`			(optional) gives the name of a log file, which must have been given as a parameter to the `init-clf` Init function. If no name is given, the entry is recorded in the global log file.

Error Functions

Table 2-21 obj.conf Error functions

Function/Parameter	Description
`send-error`	Sends an HTML file to the client in place of a specific HTTP response status.
`path`	Specifies the full file system path of an HTML file to send to the client. The file is sent as `text/html` regardless of its name or actual type. If the file does not exist, the server sends a simple default error page.

Table 2-21 obj.conf Error functions

Function/Parameter	Description
`reason`	(optional) the text of one of the reason strings (such as “Unauthorized” or “Forbidden”). The string is not case sensitive.
`code`	(optional) a three-digit number representing the HTTP response status code, such as 401 or 407. This can be any HTTP response status code or reason phrase according to the HTTP specification.
`qos-error`	Returns an error page stating which quality of service limits caused the error and what the value of the QOS statistic was.
`code`	(optional) a three-digit number representing the HTTP response status code, such as 401 or 407. This can be any HTTP response status code or reason phrase according to the HTTP specification. The recommended value is 503.

password.conf

Purpose

By default, the web server prompts the administrator for the key database password before starting up. If you want the web server to be able to restart unattended, you need to save the password in a `password.conf` file. Be sure that your system is adequately protected so that this file and the key databases are not compromised.

Location

server_root`/https-admserv/config`

server_root`/https-`*server_id*`/config`

This file is not present by default. You must create it if you need it.

Syntax

PKCS#11_module_name`:`*password*

If you are using the internal PKCS#11 software encryption module that comes with the server, type the following:

Communicator_Cert_DB`:`*password*

If you are using a different PKCS#11 module, for example for hardware encryption or hardware accelerators, you will need to specify the name of the PKCS#11 module, followed by the password, for example:

`internal:`*password*

See Also

iPlanet Web Server Administrator's Guide

rules.properties

Purpose

Provided for backward compatibility with iPlanet Web Server 4.*x*. Using `web.xml` instead to configure servlets is recommended.

Defines servlet virtual path translations. In iPlanet Web Server 6.0, supported for the default virtual server only.

Location

server_root`/https-admserv/config`

server_root`/https-admserv/conf_bk`

server_root`/https-`*server_id*`/config`

server_root`/https-`*server_id*`/conf_bk`

Syntax

virtual_path=servlet_name

The URL `http://`*server_id*`/`*virtual_path* invokes the servlet that is given the name *servlet_name* in the `servlets.properties` file.

The *virtual_path* can be a regular expression. For example, the following expression tells the server to run the `wasp` servlet whenever there is a request for a URL such as `/my/`*xxx*`.foo`:

```
@.*[.]foo$=wasp
```

See Also

Programmer's Guide to Servlets for iPlanet Web Server

The `web.xml` file

Appendix A, "Configuration Changes Between iWS 4.x and 6.0"

The Servlet 2.2 API specification at:

`http://java.sun.com/products/servlet/index.html`

server.xml

Purpose

Defines listen sockets and virtual servers.

Location

server_root/https-admserv/config

server_root/https-admserv/conf_bk

server_root/https-*server_id*/config

server_root/https-*server_id*/conf_bk

Syntax

The file has the following basic XML syntax, with nested elements:

```
<ELEMENT attribute="value" attribute="value" ... >
    <SUBELEMENT attribute="value" attribute="value" ... />
</ELEMENT>
```

In Table 2-22, elements are in bold to distinguish them from attributes, and default values are assumed if the specified attributes are not present.

See Also

NSAPI Programmer's Guide for iPlanet Web Server, Chapter 8

Table 2-22 server.xml

Element/Attribute	Allowed Subelements or Values	Default Value	Description
SERVER	**VARS**, **LS**, **MIME**, **ACLFILE**, **VSCLASS**, **QOSPARAMS**		Defines a server. Subelements must be defined in the order shown.
qosactive	yes, no, on, off, 1, 0	no	Enables quality of service features, which let you set limits on server entities or view server statistics for bandwidth and connections.
qosmetricsinterval	Number of seconds	30	(optional) The interval during which the traffic is measured.
qosrecomputeinterval	Number of milliseconds	100	(optional) The period in which the bandwidth gets recomputed for all server entities.

Table 2-22 server.xml

Element/Attribute	Allowed Subelements or Values	Default Value	Description
legacyls			The id attribute of the listen socket for legacy (4.*x*) applications. This LS should contain only one CONNECTIONGROUP, which should be configured to only one VS, its defaultvs. All legacy applications must run on this virtual server.
VARS	(no subelements; commonly defined variables are docroot, adminusers, webapps_file, webapps_enable, accesslog, user, group, chroot, dir, and nice)		Defines variables that can be given values in server.xml and referenced in obj.conf. No variables are present by default, but the most commonly defined variable is docroot, used in the document-root function in obj.conf. For more information, see Chapter 8 of the *NSAPI Programmer's Guide for iPlanet Web Server*.
LS	(none)		Defines a listen socket.
id			Internal name for the listen socket. Used in VS elements to define the listen socket(s) a virtual server is bound to.
ip	An IP address in dotted-pair or IPv6 notation. Can also be 0.0.0.0 for INADDR_ANY.		IP address of the listen socket. Configuring a listen socket to listen on 0.0.0.0 is required if more than one CONNECTIONGROUP is configured to it.
port	1 - 65535		Port number to create the listen socket on. On Unix, creating sockets that listen on ports 1 - 1024 requires superuser privileges. Configuring an SSL listen socket to listen on port 443 is recommended. Two different IP addresses can't use the same port.
security	on, off, yes, no, 1, 0	no	(optional) Determines whether the listen socket runs SSL. You can turn SSL2 or SSL3 on or off and set ciphers using an SSLPARAMS object in a CONNECTIONGROUP object.

Table 2-22 server.xml

Element/Attribute	Allowed Subelements or Values	Default Value	Description
acceptorthreads	1 - 1024	1	(optional) Number of acceptor threads for the listen socket.
family	inet, inet6, nca	inet	(optional) The socket family type. Use the value inet6 for IPv6 listen sockets. When using the value of inet6, IPv4 addresses are prefixed with ::ffff: in the log file. Specify nca to make use of the Solaris Network Cache and Accelerator.
blocking	on, off, yes, no, 1, 0	no	(optional) Determines whether the listen socket and the accepted socket are put in to blocking mode. Use of blocking mode may improve benchmark scores. Should be no for production environments.
CONNECTIONGROUP	**SSLPARAMS**		Defines MIME types.
id			Internal name for the connection group. Used in a VS element to define the connections used by the virtual server.
matchingip	An IP address in dotted-pair or IPv6 notation or the value default. Cannot be 0.0.0.0 for INADDR_ANY.		IP address that the associated virtual servers use. Must be default if the containing LS does not have ip=0.0.0.0. If the containing LS has ip=0.0.0.0, can be a specific IP address or default. In this case, default means any IP addresses not specified in other LS or CONNECTIONGROUP elements.
defaultvs			The id attribute of the default virtual server for this particular connection group.
servername			Tells the server what to put in the host name section of any URLs it sends to the client. If you append a colon and port number, that port will be used in URLs the server sends to the client.

Table 2-22 server.xml

Element/Attribute	Allowed Subelements or Values	Default Value	Description
SSLPARAMS	(none)		Defines SSL parameters of a connection group. An SSLPARAMS element is required inside, and only allowed inside, a CONNECTIONGROUP element contained by a listen socket that has its security attribute set to on.
servercertnickname			The nickname of the server certificate in the certificate database or the PKCS#11 token. In the certificate, the name format is *tokenname*:*nickname*. Including the *tokenname*: part of the name in this attribute is optional.
ssl2	on, off, yes, no, 1, 0	no	(optional) Determines whether SSL2 is enabled.
ssl2ciphers	rc4, rc4export, rc2, rc2export, idea, des, desede3	none	(optional) A space-separated list of the SSL2 ciphers used, with the prefix + to enable or - to disable, for example +rc4.
ssl3	on, off, yes, no, 1, 0	yes	(optional) Determines whether SSL3 is enabled.
ssl3tlsciphers	rsa_rc4_128_md5, rsa3des_sha, rsa_des_sha, rsa_rc4_40_md5, rsa_rc2_40_md5, rsa_null_md5, rsa_des_56_sha, rsa_rc4_56_sha	none	(optional) A space-separated list of the SSL3 and TLS ciphers used, with the prefix + to enable or - to disable, for example +rsa_des_sha.
tls	on, off, yes, no, 1, 0	no	(optional) Determines whether TLS is enabled.
tlsrollback	on, off, yes, no, 1, 0	on	(optional) Determines whether TLS rollback is enabled.
clientauth	on, off, yes, no, 1, 0	no	(optional) Determines whether SSL3 client authentication is performed on every request, independent of ACL-based access control.
MIME	(none)		Defines MIME types.

Table 2-22 server.xml

Element/Attribute	Allowed Subelements or Values	Default Value	Description
`id`			Internal name for the MIME types listing. Used in a VS element to define the MIME types used by the virtual server.
`file`			The name of a MIME types file. For information about the format of this file, see Appendix B of the *NSAPI Programmer's Guide for iPlanet Web Server*.
`ACLFILE`	(none)		References one or more ACL files.
`id`			Internal name for the ACL file listing. Used in a VS element to define the ACL file used by the virtual server.
`file`			A space-separated list of ACL files. Each ACL file must have a unique name. For information about the format of an ACL file, see the *Administrator's Guide for iPlanet Web Server*. The name of the default ACL file is `generated.https-`*server_id*`.acl`, and the file resides in the *server_root*`/`*server_id*`/httpacl` directory. To use this file, you must reference it in `server.xml`.
`VSCLASS`	**`VARS`**, **`VS`**, **`QOSPARAMS`**		Defines a virtual server class. Subelements must be defined in the order shown.
`id`			Virtual server class ID. This is a unique ID that allows lookup of a specific virtual server class.
`objectfile`			The file name of the `obj.conf` file for this class of virtual servers. Cannot be overridden in a VS element.

Table 2-22 server.xml

Element/Attribute	Allowed Subelements or Values	Default Value	Description
rootobject		default	(optional) Tells the server which object loaded from an obj.conf file is the default. The default object is expected to have all the name translation (NameTrans) directives for the virtual server. The Server Manager assumes the default to be the object named default.
acceptlanguage	on, off	off	(optional) If on, the server parses the Accept-Language header and sends an appropriate language version based on which language the client can accept. You should set this value to on only if the server supports multiple languages. Can be overridden in a VS element.
VS	**VARS**, **QOSPARAMS**, **USERDB**		Defines a virtual server. Subelements must be defined in the order shown.
id	Must begin with a letter		Virtual server ID. This is a unique ID that allows lookup of a specific virtual server. Can also be referred to as the variable $id in an obj.conf file.
connections			(optional) A space-separated list of CONNECTIONGROUP ids that specify the connection(s) the virtual server uses. Required only for a VS that is not the defaultvs of a CONNECTIONGROUP.
urlhosts			A space-separated list of values allowed in the URLHost request header to select the current virtual server. Each VS that is configured to the same CONNECTIONGROUP must have a unique urlhosts value for that group.
mime			The id of the MIME element used by the virtual server.

Table 2-22 server.xml

Element/Attribute	Allowed Subelements or Values	Default Value	Description
state	on, off, disable	on	(optional) Determines whether a VS is active (on) or inactive (off, disable). When inactive, a VS does not service requests. If a VS is disable, only the global server administrator can turn it on.
aclids			(optional) One or more id attributes of ACLFILE elements, separated by spaces. Specifies the ACL file(s) used by the virtual server.
errorlog			(optional) Specifies a log file for virtual-server-specific error messages.
acceptlanguage	on, off	off	(optional) If on, the server parses the Accept-Language header and sends an appropriate language version based on which language the client can accept. You should set this value to on only if the server supports multiple languages.
QOSPARAMS	(none)		Defines quality of service parameters of a SERVER, VSCLASS, or VS.
maxbps	Bytes per second		(optional) The maximum bandwidth limit for the SERVER, VSCLASS, or VS.
enforcebandwidth	yes, no, on, off, 1, 0	no	Specifies whether the bandwidth limit should be enforced or not.
maxconn	Number of connections		(optional) The maximum number of concurrent connections for the SERVER, VSCLASS, or VS.
enforceconnections	yes, no, on, off, 1, 0	no	Specifies whether the connection limit should be enforced or not.
USERDB	(none)		Defines the user database used by the virtual server.
id			The user database name in the virtual server's ACL file.
database			The user database name in the dbswitch.conf file.

Table 2-22 server.xml

Element/Attribute	Allowed Subelements or Values	Default Value	Description
`basedn`			(optional) Overrides the base DN lookup in the `dbswitch.conf` file.
`certmaps`			(optional) Specifies which certificate to LDAP entry mappings (defined in `certmap.conf`) to use. If not present, all mappings are used. All lookups based on mappings in `certmap.conf` are relative to the final base DN of the VS.

servers.lst

Purpose

Lists the iPlanet servers managed by the Administration Server. Do not modify this file.

Location

server_root`/https-admserv/config`

Syntax

protocol`:`*server*

Table 2-23 servers.lst

Directive	Allowed Values	Default Value	Description
protocol	`http`, `https`	`https`	A communication protocol.
server	An iPlanet server name	`Web Server, Enterprise Edition`	An iPlanet server that is managed by the Administration Server.

servlets.properties

Purpose

Provided for backward compatibility with iPlanet Web Server 4.*x*. Using `web-apps.xml` instead to configure servlets is recommended.

Defines global servlet settings and the list of servlets in the system. In iPlanet Web Server 6.0, supported for the default virtual server only.

Location

server_root`/https-admserv/config`

server_root`/https-admserv/conf_bk`

server_root`/https-`*server_id*`/config`

server_root`/https-`*server_id*`/conf_bk`

Syntax

`servlets.`*property=value*

`servlet.`*servlet_name*`.`*property=value*

The `servlet` properties, described in Table 2-24, apply only to the named servlet. The `servlets` properties, described in Table 2-25, apply to all servlets.

See Also

Programmer's Guide to Servlets for iPlanet Web Server

The `server.xml` and `web-apps.xml` files

Appendix A, "Configuration Changes Between iWS 4.x and 6.0"

The Servlet 2.2 API specification at:

`http://java.sun.com/products/servlet/index.html`

Table 2-24 servlets.properties individual (servlet) properties

Property	Allowed Values	Default Value	Description
`code`	Class or class file name		The name of the class or class file for the servlet. The `.class` extension is optional.
`context`	Context name		The context to which the servlet belongs. You change context settings using the `contexts.properties` file.

Table 2-24 servlets.properties individual (servlet) properties

Property	Allowed Values	Default Value	Description
`classpath`	URLs or paths with forward slashes only		The URL or path to the directory where classes are located or the list of directories (but not URLs) or jar files, like the CLASSPATH environment variable.
`initArgs`	Comma separated *name*=*value* pairs		List of *name*=*value* pairs which can be accessed by the servlet using the servlet API calls.
`startup`	`true`, `false`	`true`	Determines whether the servlet is started up automatically when the web server is started up.

Table 2-25 servlets.properties general (servlets) properties

Property	Allowed Values	Default Value	Description
`config.docRoot`	A path with forward slashes	Web server's document root	The document root for all servlets. If `docRoot` is not specified, the web server's document root is used.
`config.realPathFromRequest`	`true`, `false`	`false`	If `true`, calculates `getRealPath` based on the `docRoot` of the servlets. If `false`, tries to go through normal NSAPI steps.
`config.respondCookieVersion`	A cookie version number	`0`	Tells the server whether to respond with a specific cookie version.
`config.sessionExpireOnClose`	`true`, `false`	`false`	Tells the server to mark session cookies as directed to expire when the user quits the browser.
`sessionmgr`	A session manager object	`com.iplanet.server.http.session.IWSSessionMan-ager` (all on one line, no dash)	The name of the session manager for the servlet. Some session managers, such as `MMapSessionManager`, can only be instantiated once within the server.

Table 2-25 servlets.properties general (servlets) properties

Property	Allowed Values	Default Value	Description
`config.reloadInterval`	Number of seconds	`5`	The time interval within which the server checks for JSP and servlet files being modified.
`config.bufferSize`	Number of bytes	`4096`	The initial HTTP output stream buffer size.
`startup`	`true`, `false`	`true`	Determines whether all servlets are started up automatically when the web server is started up. Using the `servlet.startup` property instead is recommended.

web.xml

Purpose

Defines a web application, including its servlets, URL mappings, security constraints, and so on. Each web application has its own `web.xml` file.

Location

The location is application-specific and user-defined.

Syntax

Refer to the .DTD file at:

`http://java.sun.com/j2ee/dtds/web-app_2_2.dtd`

or at:

server_root`/bin/https/dtds/web-app_2_2.dtd`

See Also

iPlanet Web Server Programmer's Guide to Servlets

The Servlet 2.2 API specification at:

`http://java.sun.com/products/servlet/index.html`

The JSP 1.1 specification at:

`http://java.sun.com/products/jsp/download.html`

There is no listing of this file's elements and attributes because the Servlet 2.2 API specification describes them.

web-apps.xml

Purpose

Defines a set of web applications hosted by a virtual server. Each virtual server can have its own `web-apps.xml` file.

Location

This is the location for the default file only.

server_root`/https-`*server_id*`/config`

Syntax

Most of the file has the following basic XML syntax, with nested elements:

```
<ELEMENT attribute="value" attribute="value" ... >
    <SUBELEMENT attribute="value" attribute="value" ... />
</ELEMENT>
```

In Table 2-26, elements are in bold to distinguish them from attributes, and default values are assumed if the specified attributes are not present.

See Also

iPlanet Web Server Programmer's Guide to Servlets

The Servlet 2.2 API specification at:

`http://java.sun.com/products/servlet/index.html`

The JSP 1.1 specification at:

`http://java.sun.com/products/jsp/download.html`

Table 2-26 web-apps.xml

Element/Attribute	Allowed Subelements or Values	Default Value	Description
`auth-native`	(none)		(optional) Configures a specific native user/group database for authentication and role mapping. If this element is not specified, authentication is enabled using the native default authentication database.
`authdb`			The native authentication database.
`class-loader`	(none)		The class loader for the web application.
`classpath`			The classpath used by the class loader.
`delegate`	`true`, `false`	`false`	Specifies that the class loader for the virtual server or system is called first to load a class.
`reload-interval`	Number of seconds	`30`	The time interval within which the server checks for web applications being modified.
`description`	(none)		A description of a parameter. Used within an `init-param` element. iPlanet Web Server ignores this element.

Table 2-26 web-apps.xml

Element/Attribute	Allowed Subelements or Values	Default Value	Description
`filter`, **`filter-mapping`**			Implements the Filter API from the Servlet 2.3 specification. Used within a `web-app` element. Although iPlanet Web Server 6.0 supports only the Servlet 2.2 API in the `web.xml` file, this feature is available in the `web-apps.xml` file. Except for their file location, `filter` and `filter-mapping` are as described in the Servlet 2.3 specification. For more information, see: `http://java.sun.com/products/servlet/index.html`
`form-login-session`	**`session-manager`**		Configures form-based authentication for single sign-on across all web applications in a virtual server. If not present, the default virtual server level session manager is used.
`cookie-name`	A cookie name	`iwsformloginid`	The name of the cookie that tracks the session ID.
`timeOut`	Number of seconds	`600` (10 minutes)	The session timeout.

Table 2-26 web-apps.xml

Element/Attribute	Allowed Subelements or Values	Default Value	Description
`init-param`	**`param-name`**, **`param-value`**, **`description`**		Specifies an initialization parameter for the containing element. The attributes of `init-param` depend on the object referenced by the containing element. For example, if the containing element is `session-manager` and the session manager is `IWSSessionManager`, the attributes of `init-param` are the initialization parameters of `IWSSessionManager`.
`jsp-servlet`	**`init-param`**		Configures JSP compilation behavior. Set the initialization parameter `use-precompiled` to `true` to tell iPlanet Web Server that all JSPs in a virtual server are precompiled. See the *iPlanet Web Server Programmer's Guide to Servlets* for more information about `jsp-servlet` initialization parameters.
`enable`	`true`, `false`	`true`	Enables JSP.
`interval`	Number of seconds	`0`	The interval between flushes.
`param-name`	(none)		The name of a parameter. Used within an `init-param` element.
`param-value`	(none)		The value of a parameter. Used within an `init-param` element.
`parameter-encoding`	(none)		Advises the web server on how to decode parameters from forms.

Table 2-26 web-apps.xml

Element/Attribute	Allowed Subelements or Values	Default Value	Description
enc	none, auto, or a specific *encoding* such as utf8 or Shift_JIS	auto	• *encoding*: uses the specified encoding. • none: uses the system default encoding. • auto: tries to figure out the encoding from, in order, 1) the charset , 2) the parameter Encoding attribute, then 3) a hidden form field defined in form-hint-field. Otherwise same as none.
form-hint-field		j_encoding	The name of the hidden field in the form that specifies the encoding.
response-buffer	(none)		Configures the initial and default size of the HTTP servlet's response buffer.
flush-timeout	Number of seconds	0	Forces the stream to flush the data if the specified number of seconds has elapsed since the last flush. If set to 0 (the default) or a negative number, the output stream doesn't force a flush unless the buffer is full.
size	Number of bytes	8192	The buffer size.
response-cookie	(none)		Tells the server to respond with a specific cookie version.
version	A cookie version number	0	The cookie version.
role-mapping	(none)		Maps role-name values from web.xml to LDAP users or groups.

Table 2-26 web-apps.xml

Element/Attribute	Allowed Subelements or Values	Default Value	Description
map-to	user, group	group	Specifies whether to map role-name values from web.xml to LDAP users or groups.
session-cookie	(none)		Sets parameters for the session cookie.
domain	A domain name	(none)	If this attribute is present, its value is tagged onto the cookie. There is no default value.
is-secure	true, false	false	If set to true, the server sends the secure attribute in the session cookie if the request came in a secure connection. The default is false.
session-manager	**init-param**		The session manager for the web application. See the *Programmer's Guide to Servlets for iPlanet Web Server* for the initialization parameters for each session manager.
class			The class for the session manager.
session-tracking	(none)		Determines the method of session tracking.
use-cookies	true, false	true	Uses cookies for session tracking if true.
use-url-rewriting	true, false	true	Uses URL rewriting for session tracking if true.
tempdir	(none)		A temporary directory used by the web application.
dir			The temporary directory.

Table 2-26 web-apps.xml

Element/Attribute	Allowed Subelements or Values	Default Value	Description
vs	**auth-native**, **class-loader**, **form-login-session**, **jsp-servlet**, **parameter-encoding**, **response-buffer**, **response-cookie**, **role-mapping**, **session-manager**, **session-tracking**, **session-cookie**, **tempdir**, **web-app**		The top-level element in the web-apps.xml file. Subelements other than web-app set defaults for all web applications.
web-app	**auth-native**, **class-loader**, **filter**, **filter-mapping**, **jsp-servlet**, **parameter-encoding**, **response-buffer**, **response-cookie**, **role-mapping**, **session-manager**, **session-tracking**, **session-cookie**, **tempdir**		The web application. A web application is packaged in a WAR file and can contain servlets, JSPs, HTML pages, class files, and other resources of an application. The subelements of a web-app element override the equivalent subelements of the containing vs element for that web application.
dir			The directory where the web application contents are located.
uri			The URI that clients use to access the web application. This URI can be a regular expression.

Chapter 3

Server-Parsed HTML Tags

HTML files can contain tags that are executed on the server. In addition to supporting the standard server-side tags, iPlanet Web Server 6.0 allows you to embed servlets and define your own server-side tags.

This chapter has the following sections:

- Using Server-Side HTML Commands
- Embedding Servlets
- Defining Customized Server-Parsed HTML Tags

NOTE The server parses server-side tags only if server-side parsing has been activated. Use the Parse HTML page in the Content Management tab of the Class Manager interface to enable or disable the parsing of server-side tags. (To display the Class Manager, select the Manage Classes page on the Virtual Server Class tab in the Server Manager, select a class from the list, then select the Manage button.)

When you activate parsing, you need to be sure that the following directives are added to your `magnus.conf` file (note that native threads are turned off):

```
Init funcs="shtml_init,shtml_send"
shlib="install_dir/bin/https/bin/Shtml.dll" NativeThreads="no"
fn="load-modules"
```

Note that you must set `NativeThread="no"` for 6.0 iPlanet Web Servers. In addition, these functions now originate from `Shtml.dll` (or `libShtml.so` on Unix), which is located in *install_dir*`/bin/https/bin` for Windows NT (and *install_dir*`/bin/https/lib` for Unix).

In addition, be sure that the following directive is added to your `obj.conf` file:

```
<Object name="default">
...
...
Service fn="shtml_send" type="magnus-internal/parsed-html"
method="(GET|HEAD)"
...
</Object>
```

Using Server-Side HTML Commands

This section describes the HTML commands for including server-parsed tags in HTML files. These commands are embedded into HTML files, which are processed by the built-in SAF `parse-html`.

The server replaces each command with data determined by the command and its attributes.

The format for a command is:

<!--*#command attribute1 attribute2* <Body>... -->

The format for each `attribute` is a name-value pair such as:

name="*value*"

Commands and attribute names should be in lower case.

The commands are “hidden” within HTML comments so they are ignored if not parsed by the server. The standard server-side commands are:

- `config`
- `include`
- `echo`
- `fsize`
- `flastmod`

- exec

config

The config command initializes the format for other commands.

- The errmsg attribute defines a message sent to the client when an error occurs while parsing the file. This error is also logged in the error log file.
- The timefmt attribute determines the format of the date for the flastmod command. It uses the same format characters as the util_strftime function. The default time format is: "%A, %d-%b-%y %T".

  Refer to the Time Formats appendix in the *NSAPI Programmer's Guide for iPlanet Web Server* for details about time formats.
- The sizefmt attribute determines the format of the file size for the fsize command. It can have one of these values:
  - bytes to report file size as a whole number in the format 12,345,678.
  - abbrev (the default) to report file size as a number of KB or MB.

Example:

```
<!--#config timefmt="%r %a %b %e, %Y" sizefmt="abbrev"-->
```

This sets the date format to a value such as 08:23:15 AM Wed Apr 15, 1996, and the file size format to the number of KB or MB of characters used by the file.

include

The include command inserts a file into the parsed file. You can nest files by including another parsed file, which then includes another file, and so on. The client requesting the parsed document must also have access to the included file if your server uses access control for the directories where they reside.

In iPlanet Web Server 6.0, you can use the include command with the virtual attribute to include a CGI program file. You must also use an exec command to execute the CGI program.

- The virtual attribute is the URI of a file on the server.
- The file attribute is a relative path name from the current directory. It cannot contain elements such as ../ and it cannot be an absolute path.

Example:

```
<!--#include file="bottle.gif"-->
```

echo

The `echo` command inserts the value of an environment variable. The `var` attribute specifies the environment variable to insert. If the variable is not found, "(none)" is inserted. For a list of environment variables, see the section "Environment Variables in Server-Side HTML Commands," on page 113.

Example:

```
<!--#echo var="DATE_GMT"-->
```

fsize

The `fsize` command sends the size of a file. The attributes are the same as those for the `include` command (`virtual` and `file`). The file size format is determined by the `sizefmt` attribute in the `config` command.

Example:

```
<!--#fsize file="bottle.gif"-->
```

flastmod

The `flastmod` command prints the date a file was last modified. The attributes are the same as those for the `include` command (`virtual` and `file`). The date format is determined by the `timefmt` attribute in the `config` command.

Example:

```
<!--#flastmod file="bottle.gif"-->
```

exec

The `exec` command runs a shell command or CGI program.

- The `cmd` attribute (Unix only) runs a command using `/bin/sh`. You may include any special environment variables in the command.

- The `cgi` attribute runs a CGI program and includes its output in the parsed file.

Example:

```
<!--#exec cgi="workit.pl"-->
```

Environment Variables in Server-Side HTML Commands

In addition to the normal set of environment variables used in CGI, you may include the following variables in your parsed commands:

- `DOCUMENT_NAME`

  is the file name of the parsed file.

- `DOCUMENT_URI`

  is the virtual path to the parsed file (for example, `/shtml/test.shtml`).

- `QUERY_STRING_UNESCAPED`

  is the unescaped version of any search query the client sent with all shell-special characters escaped with the `\` character.

- `DATE_LOCAL`

  is the current date and local time.

- `DATE_GMT`

  is the current date and time expressed in Greenwich Mean Time.

- `LAST_MODIFIED`

  is the date the file was last modified.

Embedding Servlets

iPlanet Web Server 6.0 supports the <SERVLET> tag as introduced by Java Web Server. This tag allows you to embed servlet output in an SHTML file. No configuration changes are necessary to enable this behavior. If SSI and servlets are both enabled, the <SERVLET> tag is enabled.

The <SERVLET> tag syntax is slightly different from that of other SSI commands; it resembles the <APPLET> tag syntax:

```
<servlet name=name code=code codebase=path iParam1=v1  iParam2=v2>
<param name=param1 value=v3>
<param name=param2 value=v4>
.
.
</servlet>
```

If the servlet is part of a web application, the code parameter is required and other parameters are ignored. The code parameter must include:

- The value of the `url-pattern` element defined in the `web.xml` file for the web application. For more information about `web.xml`, see the Servlet 2.2 API specification:

  `http://java.sun.com/products/servlet/index.html`

- The value of the `uri` attribute defined in the `web-apps.xml` file for the web application. For more information about `web-apps.xml`, see the *Programmer's Guide to Servlets in iPlanet Web Server*.

For example, if you wanted to include the following in your SHTML file:

```
<servlet name=pparams code="/PrintApp/PrintParams">
</servlet>
```

you would need to include the following in your `web-apps.xml` file:

```
<web-app uri="/PrintApp"
dir="/iws60/https-server.iplanet.com/acme.com/webapps/PrintApp"/>
```

you would also need to include the following in your `web.xml` file:

```
<servlet>
   <servlet-name> pparams </servlet-name>
   <servlet-class> PrintPackage.PrintParams </servlet-class>
</servlet>
<servlet-mapping>
   <servlet-name> pparams </servlet-name>
   <url-pattern> /PrintParams </url-pattern>
</servlet-mapping>
```

You must also include any servlet initialization parameters in the `web.xml` file.

For legacy (iPlanet Web Server 4.*x*) servlets, the `code` parameter specifies the `.class` file for the servlet and is required. The `codebase` parameter is required if the servlet is *not* defined in the `servlets.properties` file and the `.class` file is *not* in the same directory as the HTML file containing the `<SERVLET>` tag. Legacy servlets must be configured in the default virtual server and do not require a `web.xml` file.

For more information about creating servlets, see the *Programmer's Guide to Servlets in iPlanet Web Server*.

Defining Customized Server-Parsed HTML Tags

In iPlanet Web Server 6.0, users can define their own server-side tags. For example, you could define the tag `HELLO` to invoke a function that prints "Hello World!" You could have the following code in your `hello.shtml` file:

```
<html>
<head>
<title>shtml custom tag example</title>
</head>
<body>
<!--#HELLO-->
</body>
</html>
```

When the browser displays this code, each occurrence of the `HELLO` tag calls the function.

The Mechanics

The steps for defining a customized server-parsed tag are:

1. Define the Functions that Implement the Tag.

   You must define the tag execution function. You must also define other functions that are called on tag loading and unloading and on page loading and unloading.

2. Write an Initialization Function to Register the New Tag.

   Write an initialization function that registers the tag using the `shtml_add_tag` function.

3. Load the New Tag into the Server.

Define the Functions that Implement the Tag

Define the functions that implement the tags in C, using NSAPI.

- Include the header `shtml_public.h`, which is in the directory *install_dir*`/plugins/include/shtml`.
- Link against the `shtml` shared library. On Windows NT, `shtml.dll` is in *install_dir*`/bin/https/bin`. On Unix platforms, `libshtml.so` or `.sl` is in *install_dir*`/bin/https/lib`.

`ShtmlTagExecuteFunc` is the actual tag handler. It gets called with the usual NSAPI *pblock*, *Session*, and *Request* variables. In addition, it also gets passed the `TagUserData` created from the result of executing the tag loading and page loading functions (if defined) for that tag.

The signature for the tag execution function is:

```
typedef int (*ShtmlTagExecuteFunc)(pblock*, Session*, Request*,
TagUserData, TagUserData);
```

Write the body of the tag execution function to generate the output to replace the tag in the `.shtml` page. Do this in the usual NSAPI way, using the `net_write` NSAPI function, which writes a specified number of bytes to a specified socket from a specified buffer.

For more information about writing NSAPI plugins, see Chapter 4, “Creating Custom SAFs,” in the *NSAPI Programmer’s Guide for iPlanet Web Server*.

For more information about `net_write` and other NSAPI functions, see Chapter 5, “NSAPI Function Reference,” of the *NSAPI Programmer’s Guide for iPlanet Web Server*.

The tag execution function must return an `int` that indicates whether the server should proceed to the next instruction in `obj.conf` or not, which is one of:

- `REQ_PROCEED` -- the execution was successful.
- `REQ_NOACTION` -- nothing happened.
- `REQ_ABORTED` -- an error occurred.
- `REQ_EXIT` -- the connection was lost.

The other functions you must define for your tag are:

- `ShtmlTagInstanceLoad`

  This is called when a page containing the tag is parsed. It is not called if the page is retrieved from the browser's cache. It basically serves as a constructor, the result of which is cached and is passed into `ShtmlTagExecuteFunc` whenever the execution function is called.

- `ShtmlTagInstanceUnload`

  This is basically a destructor for cleaning up whatever was created in the `ShtmlTagInstanceLoad` function. It gets passed the result that was originally returned from the `ShtmlTagInstanceLoad` function.

- `ShtmlTagPageLoadFunc`

  This is called when a page containing the tag is executed, regardless of whether the page is still in the browser's cache or not. This provides a way to make information persistent between occurrences of the same tag on the same page.

- `ShtmlTagPageUnLoadFn`

  This is called after a page containing the tag has executed. It provides a way to clean up any allocations done in a `ShtmlTagPageLoadFunc` and hence gets passed the result returned from the `ShtmlTagPageLoadFunc`.

The signatures for these functions are:

```
#define TagUserData void*
typedef TagUserData (*ShtmlTagInstanceLoad)(
    const char* tag, pblock*, const char*, size_t);
typedef void (*ShtmlTagInstanceUnload)(TagUserData);
typedef int (*ShtmlTagExecuteFunc)(
    pblock*, Session*, Request*, TagUserData, TagUserData);
typedef TagUserData (*ShtmlTagPageLoadFunc)(
    pblock* pb, Session*, Request*);
typedef void (*ShtmlTagPageUnLoadFunc)(TagUserData);
```

Here is the code that implements the `HELLO` tag:

```
/*
 * mytag.c: NSAPI functions to implement #HELLO SSI calls
 *
 *
 */

#include "nsapi.h"
#include "shtml/shtml_public.h"

/* FUNCTION : mytag_con
 *
 * DESCRIPTION: ShtmlTagInstanceLoad function
 */
#ifdef __cplusplus
extern "C"
#endif
TagUserData
mytag_con(const char* tag, pblock* pb, const char* c1, size_t t1)
{
   return NULL;
}

/* FUNCTION : mytag_des
 *
 * DESCRIPTION: ShtmlTagInstanceUnload
 */
#ifdef __cplusplus
extern "C"
#endif
```

```
void
mytag_des(TagUserData v1)
{

}

/* FUNCTION : mytag_load
 *
 * DESCRIPTION: ShtmlTagPageLoadFunc
 */
#ifdef __cplusplus
extern "C"
#endif
TagUserData
mytag_load(pblock *pb, Session *sn, Request *rq)
{
   return NULL;
}

/* FUNCTION : mytag_unload
 *
 * DESCRIPTION: ShtmlTagPageUnloadFunc
 */
#
#ifdef __cplusplus
extern "C"
#endif
void
mytag_unload(TagUserData v2)
{

}

/* FUNCTION : mytag
 *
 * DESCRIPTION: ShtmlTagExecuteFunc
 */
#ifdef __cplusplus
extern "C"
#endif
int
mytag(pblock* pb, Session* sn, Request* rq, TagUserData t1,
TagUserData t2)
{
   char* buf;
   int length;
   char* client;
```

```
    buf = (char *) MALLOC(100*sizeof(char));
    length = util_sprintf(buf, "<h1>Hello World! </h1>", client);
    if (net_write(sn->csd, buf, length) == IO_ERROR)
    {
        FREE(buf);
        return REQ_ABORTED;
    }
    FREE(buf);
    return REQ_PROCEED;
}

/* FUNCTION : mytag_init
*
* DESCRIPTION: initialization function, calls shtml_add_tag() to
* load new tag
*/
#
#ifdef __cplusplus
extern "C"
#endif
int
mytag_init(pblock* pb, Session* sn, Request* rq)
{
    int retVal = 0;
// NOTE: ALL arguments are required in the shtml_add_tag() function
    retVal = shtml_add_tag("HELLO", mytag_con, mytag_des, mytag,
mytag_load, mytag_unload);

    return retVal;
}
/* end mytag.c */
```

Write an Initialization Function to Register the New Tag

In the initialization function for the shared library that defines the new tag, register the tag using the function `shtml_add_tag`. The signature is:

```
NSAPI_PUBLIC int shtml_add_tag (
    const char* tag,
    ShtmlTagInstanceLoad ctor,
    ShtmlTagInstanceUnload dtor,
    ShtmlTagExecuteFunc execFn,
    ShtmlTagPageLoadFunc pageLoadFn,
    ShtmlTagPageUnLoadFunc pageUnLoadFn);
```

Any of these arguments can return `NULL` except for the `tag` and `execFn`.

Load the New Tag into the Server

After creating the shared library that defines the new tag, you load the library into the iPlanet Web Server in the usual way for NSAPI plugins. That is, add the following directives to the configuration file `magnus.conf`:

1. Add an `Init` directive whose `fn` parameter is `load-modules` and whose `shlib` parameter is the shared library to load. For example, if you compiled your tag into the shared object *install_dir*`/hello.so`, it would be:

   ```
   Init funcs="mytag,mytag_init" shlib="installdir/hello.so"
   fn="load-modules"
   ```

2. Add another `Init` directive whose `fn` parameter is the initialization function in the shared library that uses `shtml_add_tag` to register the tag. For example:

   ```
   Init fn="mytag_init"
   ```

Appendix A

Configuration Changes Between iWS 4.x and 6.0

This chapter summarizes major configuration file changes since the last version of iPlanet Web Server. The following 4.*x* files are described:

- magnus.conf
- obj.conf
- contexts.properties
- rules.properties
- servlets.properties

magnus.conf

The `magnus.conf` file has lost directives to other configuration files, gained directives from other configuration files, and acquired entirely new directives. Table 3-1 summarizes the changes.

Table 3-1 magnus.conf changes

4.x Directive	6.0 Directive	Comments
`AccelFileCache`	(none)	Obsolete because a file cache accelerator is no longer necessary
`AcceptLanguage`	(none)	See the VSCLASS and VS elements in `server.xml`
`AcceptTimeout`	(none)	`IOTimeout` is an approximate equivalent

Table 3-1 magnus.conf changes

4.x Directive	6.0 Directive	Comments
ACLFile	(none)	See the ACLFILE element in server.xml
Address	(none)	See the LS and VS elements in server.xml
AdminLanguage	AdminLanguage	(unchanged)
AsyncDNS	AsyncDNS	(unchanged)
BlockingListenSockets	(none)	Obsolete due to virtual server implementation
CGIExpirationTimeout	CGIExpirationTimeout	(unchanged)
CGIStubIdleTimeout	CGIStubIdleTimeout	(unchanged)
CGIWaitPid	CGIWaitPid	(unchanged)
ChildRestartCallback	ChildRestartCallback	(unchanged)
Chroot	Chroot	(unchanged)
ChunkedRequestBufferSize	ChunkedRequestBufferSize	(unchanged)
ChunkedRequestTimeout	ChunkedRequestTimeout	(unchanged)
Ciphers	(none)	See the SSLPARAMS element in server.xml
ClientLanguage	ClientLanguage	(unchanged)
Concurrency	(none)	Obsolete due to virtual server implementation
(none)	ConnQueueSize	(new)
DaemonStats	(none)	Obsolete due to new performance statistics system
DefaultCharSet	DefaultCharSet	(unchanged)
DefaultLanguage	DefaultLanguage	(unchanged)
DNS	DNS	(unchanged)
ErrorLog	ErrorLog	(unchanged)
ErrorLogDateFormat	ErrorLogDateFormat	(unchanged)
ExtraPath	ExtraPath	(unchanged)
HeaderBufferSize	HeaderBufferSize	(unchanged)
HTTPVersion	HTTPVersion	(unchanged)

Table 3-1 magnus.conf changes

4.x Directive	6.0 Directive	Comments
(none)	`IOTimeout`	(new)
(none)	`Init` functions from `obj.conf`	All functions are present except for `cache-init` and `load-types`, which are obsolete (for `load-types`, see the `MIME` element in the `server.xml` file).
`KeepAliveThreads`	`KeepAliveThreads`	(unchanged)
`KeepAliveTimeout`	`KeepAliveTimeout`	(unchanged)
`KernelThreads`	`KernelThreads`	(unchanged)
`ListenQ`	`ListenQ`	(unchanged)
`LoadObjects`	(none)	See the `VSCLASS` element in `server.xml`
`LogFlushInterval`	`LogFlushInterval`	(unchanged)
`LogVerbose`	`LogVerbose`	(unchanged)
`MaxCGIStubs`	`MaxCGIStubs`	(unchanged)
`MaxKeepAliveConnections`	`MaxKeepAliveConnections`	(unchanged)
`MaxProcs`	`MaxProcs`	(unchanged)
`MaxRqHeaders`	`MaxRqHeaders`	(unchanged)
`MaxThreads`	(none)	Obsolete due to new thread handling system
`MinCGIStubs`	`MinCGIStubs`	(unchanged)
`MinProcs`	(none)	Obsolete due to new thread handling system
`MinThreads`	(none)	Obsolete due to new thread handling system
`MtaHost`	`MtaHost`	(unchanged)
`NativePoolMaxThreads`	`NativePoolMaxThreads`	(unchanged)
`NativePoolMinThreads`	`NativePoolMinThreads`	(unchanged)
`NativePoolQueueSize`	`NativePoolQueueSize`	(unchanged)
`NativePoolStackSize`	`NativePoolStackSize`	(unchanged)
`NetSiteRoot`	`NetSiteRoot`	(unchanged)

Table 3-1 magnus.conf changes

4.x Directive	6.0 Directive	Comments
PidLog	PidLog	(unchanged)
Port	(none)	See the LS element in server.xml
PostThreadsEarly	PostThreadsEarly	(unchanged)
RcvBufSize	RcvBufSize	(unchanged)
RootObject	(none)	See the VSCLASS element in server.xml
RqThrottle	RqThrottle	(unchanged)
RqThrottleMinPerSocket	(none)	See the LS element in server.xml
(none)	RqThrottleMin	(new)
Security	Security	(unchanged)
ServerCert	(none)	See the SSLPARAMS element in server.xml
(none)	ServerConfigurationFile	(new)
ServerID	ServerID	(unchanged)
ServerKey	(none)	See the SSLPARAMS element in server.xml
ServerName	(none)	See the VS element in server.xml
#ServerRoot	#ServerRoot	(unchanged)
SndBufSize	SndBufSize	(unchanged)
SSL2	(none)	See the SSLPARAMS element in server.xml
SSL3	(none)	See the SSLPARAMS element in server.xml
SSL3Ciphers	(none)	See the SSLPARAMS element in server.xml
SSL3SessionTimeout	SSL3SessionTimeout	(unchanged)
SSLCacheEntries	SSLCacheEntries	(unchanged)
SSLClientAuth	(none)	See the SSLPARAMS element in server.xml

Table 3-1 magnus.conf changes

4.x Directive	6.0 Directive	Comments
SSLClientAuthDataLimit	SSLClientAuthDataLimit	(unchanged)
SSLClientAuthTimeout	SSLClientAuthTimeout	(unchanged)
SSLSessionTimeout	SSLSessionTimeout	(unchanged)
StackSize	StackSize	(unchanged)
StrictHttpHeaders	StrictHttpHeaders	(unchanged)
TerminateTimeout	TerminateTimeout	(unchanged)
ThreadIncrement	ThreadIncrement	(unchanged)
Umask	Umask	(unchanged)
UseNativePoll	UseNativePoll	(unchanged)
UseOutputStreamSize	UseOutputStreamSize	(unchanged)
User	User	(unchanged)
VirtualServerFile	(none)	Obsolete due to virtual server implementation
WincgiTimeout	WincgiTimeout	(unchanged)

obj.conf

The `obj.conf` file has lost its Init directives to the `magnus.conf` file and acquired new directives and parameters. Table 3-2 summarizes the changes. Only the new and changed directives are listed.

Table 3-2 obj.conf changes

4.x Directive	6.0 Directive	Comments
`Init` functions	(none)	All functions have moved to `magnus.conf` except for `cache-init` and `load-types`, which are obsolete (for `load-types`, see the `MIME` element in the `server.xml` file).
(none)	`AuthTrans fn=qos-handler`	(new)
`Service fn=parse-html`	`Service fn=shtml_send`	

Table 3-2 obj.conf changes

4.x Directive	6.0 Directive	Comments
`Service fn=send-cgi`	`Service fn=send-cgi`	New parameters have been added.
(none)	`Service fn=stats-xml`	(new)
(none)	`Error fn=qos-error`	(new)

contexts.properties

The `contexts.properties` file is still supported for the default virtual server. For all other virtual servers, most of the same functions are in the `web-apps.xml` file.

A few `contexts.properties` functions are in the `server.xml` file. For more information about the `server.xml` file, see the *NSAPI Programmer's Guide for iPlanet Web Server*.

Table 3-3 lists the equivalent functions in the `servlets.properties` and `web-apps.xml` files.

Table 3-3 contexts.properties to web-apps.xml correspondences

contexts.properties Property	web-apps.xml Element or Attribute	Comments
`sessionmgr`	`session-manager` element	
`sessionmgr.initArgs`	`init-param` subelements of `session-manager` element	
`initArgs`	`init-param` subelements of elements that support them	
`respondCookieVersion`	`version` attribute of `response-cookie` element	
`tempDir`	`tempdir` element	
`reloadInterval`	`reload-interval` attribute of `class-loader` element	
`bufferSize`	`size` attribute of `response-buffer` element	
`docRoot`	(none)	Specified in the `server.xml` file for each virtual server.
`inputStreamLengthCheck`	(none)	Obsolete due to web application support.

Table 3-3 contexts.properties to web-apps.xml correspondences

contexts.properties Property	web-apps.xml Element or Attribute	Comments
outputStreamFlushTimer	flush-timeout attribute of response-buffer element	
uri	uri attribute of web-app element	
authdb	authdb attribute of auth-native element	
classpath	classpath attribute of class-loader element	
singleClassLoader	(none)	Obsolete because by default each web application has a single class loader.
serverName	(none)	Specified in the server.xml file for each virtual server.
contentTypeIgnoreFromSSI	(none)	Obsolete due to web application support.
parameterEncoding	parameter-encoding element	
isModifiedCheckAggressive	(none)	Obsolete due to web application support.

rules.properties

The function of the rules.properties file is now handled by the servlet-mapping element in the web.xml file. For more information, see the Servlet 2.2 API specification at:

http://java.sun.com/products/servlet/index.html

servlets.properties

The servlets.properties file is still supported for the default virtual server. For all other virtual servers, most of the same functions are in the web-apps.xml file.

A few servlets.properties functions are in the server.xml file. For more information about the server.xml file, see the *NSAPI Programmer's Guide for iPlanet Web Server*.

A few `servlets.properties` functions are in the `web.xml` file. For more information, see the Servlet 2.2 API specification at:

`http://java.sun.com/products/servlet/index.html`

Table 3-4 and Table 3-5 list the equivalent functions in the `servlets.properties` and `web-apps.xml` files.

Table 3-4 servlets.properties to web-apps.xml correspondences for individual (servlet) properties

servlets.properties Property	web-apps.xml Element or Attribute	Comments
`code`	(none)	Specified in a `servlet-class` element in the `web.xml` file.
`context`	(none)	Obsolete because web applications are supported.
`classpath`	`classpath` attribute of `class-loader` element	
`initArgs`	(none)	Use the <SERVLET> tag, which takes servlet-specific initialization parameters.
`startup`	(none)	Specified in a `load-on-startup` element in the `web.xml` file.

Table 3-5 servlets.properties to web-apps.xml correspondences for general (servlets) properties

servlets.properties Property	web-apps.xml Element or Attribute	Comments
`config.docRoot`	(none)	Specified in the `server.xml` file for each virtual server.
`config.realPathFromRequest`	(none)	Deprecated in the Servlet 2.2 API.
`config.respondCookieVersion`	`version` attribute of `response-cookie` element	
`config.sessionExpireOnClose`	(none)	Tracking session expiration in this way is no longer necessary.
`sessionmgr`	`session-manager` element	
`config.reloadInterval`	`reload-interval` attribute of `class-loader` element	
`config.bufferSize`	`size` attribute of `response-buffer` element	

Table 3-5 servlets.properties to web-apps.xml correspondences for general (servlets) properties

servlets.properties Property	web-apps.xml Element or Attribute	Comments
`startup`	(none)	Specified in a `load-on-startup` element in the `web.xml` file. There is no global servlet startup function.

Index

A

B

C

D

E

J

K

L

M

N

O

P

Q

R

S

T

U

V

W