USER’S GUIDE

Abyss Web Server™

For Windows®

aprelium®

Abyss Web Server For Windows User's Guide

Revision History

Revision 1.0.0	February 20, 2002
Revision 1.0.3	May 10, 2002
Revision 1.0.7	July 1, 2002
Revision 1.1.0	October 3, 2002
Revision 1.1.6	July 7, 2003
Revision 1.2.0 b1	October 18, 2003
Revision 1.2.0 b2	November 10, 2003
Revision 2.0.0 b2	August 28, 2004
Revision 2.0.0 b3	November 3, 2004
Revision 2.0.0	February 18, 2005
Revision 2.0.1	April 20, 2005
Revision 2.0.6	June 10, 2005
Revision 2.3.0 b1	January 12, 2006
Revision 2.3.1	May 08, 2006
Revision 2.3.2	May 21, 2006
Revision 2.4.0 b1	November 8, 2006
Revision 2.4.0 b2	November 30, 2006
Revision 2.4.0.3	January 1, 2007
Revision 2.4.9	June 28, 2007
Revision 2.4.9.8	July 17, 2007
Revision 2.5	August 22, 2007
Revision 2.6	January 6, 2009
Revision 2.7	April 12, 2011
Revision 2.8	April 24, 2012
Revision 2.9	May 24, 2013
Revision 2.9.3	March 27, 2014

Revision 2.9.3.6	October 28, 2014
Revision 2.10	May 20, 2015
Revision 2.11	March 15, 2016
Revision 2.11.8	December 11, 2017
Revision 2.12	June 03, 2018
Revision 2.14	March 27, 2020
Revision 2.14.1	February 15, 2021
Revision 2.15	July 07, 2021
Revision 2.16.4	July 08, 2022
Revision 2.16.8	March 06, 2023
Revision 2.16.10	November 15, 2024

CONTACT INFORMATION

Web Site	`https://aprelium.com`
Technical support	`https://aprelium.com/support`
General inquiries	`https://aprelium.com/contact.html`

Table of Contents

Before You Begin ix

System requirements ix

Alternative formats ix

What you should know ix

Chapter quick reference ix

Conventions used in this guide x

Typographical conventions x

Terminology x

1. Introduction 1

What is Abyss Web Server? 1

How do web servers work? 1

2. Getting Started 3

Getting the software 3

Installing/Upgrading the software 3

First contact 3

Setting up a web site 6

Shutting down the server 6

Asking for support 7

3. Using The Console 9

What is the console? 9

Accessing the console 9

Console interface basics 10

Help buttons 10

Tables 10

Applying configuration changes 10

Console Configuration 11

Changing the console port and protocol 11

Changing the console access credentials 13

Changing the language 14

Controlling the access to the console 15

4. Server Management 17

Overview 17

Server Statistics 17

General server configuration 18

Parameters 19

MIME Types 21

Global Bandwidth Limits 22

Anti-Hacking Protection 23

Logging Parameters 25

SSL/TLS Certificates 25

Overview 26

Generating a private key 27

Importing a private key 27

Generating a CSR ... 27
Adding a signed certificate ... 29
Creating a self-signed certificate ... 29
Getting certificates from an ACME certification authority ... 30

5. Hosts Management ... 33

Overview ... 33
Declaring a new host ... 33
Configuring a host ... 35
General ... 35
Index Files ... 40
Directory Listing ... 40
Aliases ... 42
XSSI Parameters ... 43
Users and Groups ... 44
Custom Error Pages ... 45
Scripting Parameters ... 46
ASP.NET Parameters ... 51
Access Control ... 54
IP Address Control ... 55
URL Rewriting ... 57
Compression ... 61
Bandwidth Limits ... 63
Logging ... 63
Reverse Proxy ... 65
Restricted Downloads Areas ... 67
Anti-Leeching ... 68
Statistics ... 70

6. CGI, FastCGI, and ISAPI ... 73

Setting up an interpreter ... 73
How are they run? ... 73
CGI environment variables ... 74
FastCGI ... 79
ISAPI Extensions ... 79
X-Sendfile Support ... 80

7. eXtended Server Side Includes ... 81

What are XSSI? ... 81
XSSI Syntax ... 81
Attribute Values ... 81
Character Escaping ... 81
Variables Expansion ... 82
Directives ... 82
#config ... 82
#echo ... 83
#exec ... 84
#flastmod ... 84

#fsize 85
#include 85
#printenv 85
#set 86
Conditional blocks 86
Syntax 86
Expression evaluation 86
Encoding 88
8. Custom Directory Listings 89
Defining a template 89
Scripts 91
Glossary 95
A. Command Line Parameters 99
B. Startup Configuration 101
C. IP Addresses and Ranges Format 103
IPv4 Addresses and Ranges 103
IPv6 Addresses and Ranges 103
D. Patterns Format 105
E. Regular Expressions 107
F. Uninstalling The Software 109
G. Solving Ports Listening Problems 111
H. Troubleshooting Guide 113

Before You Begin

System requirements

Abyss Web Server is a multiplatform software designed for all editions of Windows XP and later (including Server 2003, Vista, 7, Server 2008, 8, 8.1, Server 2012, Server 2016, Server 2019, 10, Server 2022 and 11); macOS 10.13, 10.14, 10.15, macOS 11 (Big Sur), 12 (Monterey), 13 (Ventura), 14 (Sonoma) and 15 (Sequoia) and higher; and Linux systems with kernel version 2.4 and higher. It is compatible with both 32 and 64-bit architectures and operating systems.

This guide documents the Windows version. Abyss Web Server for Windows minimum requirements are:

- A Pentium-class processor.
- 8 Mb of free RAM.
- A modern web browser with HTML 5.0 support. Legacy web browsers (graphical or text-based) can still be used but the user experience will be partially degraded.
- TCP/IP protocol stack installed.

Although Abyss Web Server can run on a standalone computer, it is recommended to be connected to a LAN (Local Area Network) or to the Internet.

Alternative formats

This guide exists in several formats: Online HTML, Offline HTML, and PDF. For more information, browse the documentation section at **`https://aprelium.com/abyssws`**.

What you should know

Because it is impossible to provide highly detailed information for every given instruction in this guide, you are encouraged to read the documentation accompanying your operating system if you are not already familiar with it. This guide assumes also that you master the basics of web browsing.

Chapter quick reference

This guide documents the features and capabilities of Abyss Web Server. It can guide you to quickly master the basics and move on to use the advanced features.

If you are new to web servers, go to Chapter 1, "Introduction," to get a quick overview.

If you want to get started right away, go to Chapter 2, "Getting Started," for step by step instruction on installing and running the software.

Chapter 3, "Using the Console," describes the web configuration interface of Abyss Web Server.

Use Chapter 4, "Server Management," to understand how to configure your web server and manage SSL/TLS certificates.

Chapter 5, "Hosts Management," focuses on all the parameters related to the configuration of hosts served by Abyss Web Server.

See Chapter 6, "CGI, FastCGI, and ISAPI," for information on using CGI, FastCGI, and ISAPI interpreters or applications with Abyss Web Server and making scripts work.

Read Chapter 7, "eXtended Server Side Includes," for the complete reference on XSSI.

Chapter 8, "Directory Listings," provides all the necessary information on directory listings templates and scripts.

Refer to the "Glossary" for definitions of key terms and concepts used in this guide.

Conventions used in this guide

Typographical conventions

UI Object

Refers to user interface objects such as buttons, icons and text fields.

`Computer I/O`

Used for computer input and output such as URLs, filenames and commands.

`<Variable>`

Indicates a variable. You have to substitute it with its real value.

Terminology

Your computer

The computer on which Abyss Web Server is installed.

The server

The instance of Abyss Web Server under execution.

The browser

The web browser you use.

The console

The web configuration interface of Abyss Web Server.

The network

The network to which your computer is connected. It can be:

- A single computer network if you are not physically connected to any network and your computer is alone.
- A LAN (Local Area Network).
- The Internet

Chapter 1. Introduction

What is Abyss Web Server?

Abyss Web Server turns your computer in a full-featured web server. People all over the Internet can view documents, download graphics, listen to music and view movies hosted on it. Abyss Web Server was designed with both novice and experimented users in mind. That's why it is easy to use and incredibly powerful.

Abyss Web Server exists in two editions:

- Abyss Web Server X1: Free personal edition. It is a fully functional software with no limitations, no nag screens, no spyware, and no advertisements.
- Abyss Web Server X2: Professional edition. It is suitable for demanding users and medium to high loaded web sites. It supports virtual hosting and advanced features. For more information, visit **`https://aprelium.com/abyssws/x2`**

For additional information about both editions and a comparison between their feature set, browse **`https://aprelium.com/abyssws`**.

How do web servers work?

What happens when you enter in the address field of your browser the URL **`http://www.aprelium.com/doc/sample.html`**?

First, the browser slices the URL in 3 parts:

- **`http://`**: This part indicates that the document you want to access can be retrieved from web server, which understands the HTTP protocol. The HTTP protocol is a standardized language of communication between browsers and web servers.
- **`www.aprelium.com`**: This is the host name of the computer from which the document can be downloaded.
- **`/doc/sample.html`**: This is the virtual path of the document in the **`www.aprelium.com`**'s web server.

Then, the browser contacts a DNS (Domain Name System) server to know the IP address of the computer which full qualified domain name is **`www.aprelium.com`**. The domain name server is usually run by your ISP or by your company.

The browser establishes a connection channel with the web server on the computer which IP address was given by the DNS server and requests the document on the host which name is **`www.aprelium.com`** and which virtual path is **`doc/sample.html`**. The browser has to specify in the request the host name because modern web servers (including Abyss Web Server) have the ability to serve more than a one host from a single computer with a single IP address only. This is called *virtual hosting*. In such a case, the IP address of this computer is associated with more than one domain name.

The server decodes the request and maps the virtual path to a real one, which should match an existing file. The server sends the file to the browser with some useful information such as its last modification time and its MIME type. The MIME type helps the browser decide how to display the received document. In our example, it is a HTML file. So the server sets its MIME type to **`text/html`** and the browser understands that it must render it as text.

Sometimes you enter a URL without an explicit filename such as **`http://www.aprelium.com/doc`**. The browser sends the request to the web server as in the previous example. The server detects that the virtual path maps to a directory and not to a file. It searches then in this directory an index file. Index files are usually named **`index.html`** or **`index.htm`**. If it finds for example **`index.html`**, it acts as if the requested URL was **`http://www.aprelium.com/doc/index.html`**. If no index file is found, the web server generates a listing of the directory contents and sends it to the browser or reports an error.

Chapter 2. Getting Started

Getting the software

If you already have the Abyss Web Server installation package, skip this section.

To download Abyss Web Server, browse `https://aprelium.com/downloads`. Follow the provided instructions and choose the correct version of the software suitable to your computer and to the operating system you are using. Once done, the browser displays a dialog box asking where to save the downloaded file. Choose a directory and validate.

Installing/Upgrading the software

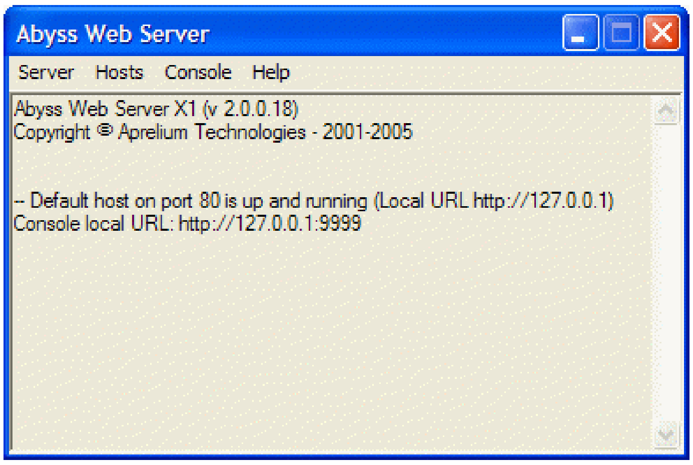
To setup Abyss Web Server:

- Open the directory where you have saved the software package.
- Double-click on the software package icon.
- If an old version of Abyss Web Server is already installed on your computer, you may be asked to uninstall it. All you have to do is to follow the on-screen instructions. *IMPORTANT NOTICE: When the uninstaller asks you to confirm the deletion of the installation directory, select* **No**.
- Read carefully the license agreement. If you agree with it, press **I Agree**. If you do not, press **Cancel** and delete Abyss Web Server package from your computer.
- Deselect components you do not want to install. **Auto Start** enables Abyss Web Server auto starting when a Windows session starts. **Start Menu Shortcuts** enables adding Abyss Web Server shortcuts in the **Start Menu**. **Documentation** installs help files.
- Press **Next**.
- Choose a directory where you want to install Abyss Web Server files. From now on, **`<Abyss Web Server directory>`** will refer to this directory.
- Press **Install**.

After installing, you will be asked if you want to launch Abyss Web Server. If you press **No**, you should press **Close** to close the setup program.

First contact

If Abyss Web Server is not running, open the **Start** menu, choose **Programs**, then **Abyss Web Server** and select **Abyss Web Server**. You can also open the directory where you have installed it and double-click on the **abyssws.exe** or **abyssws** icon.



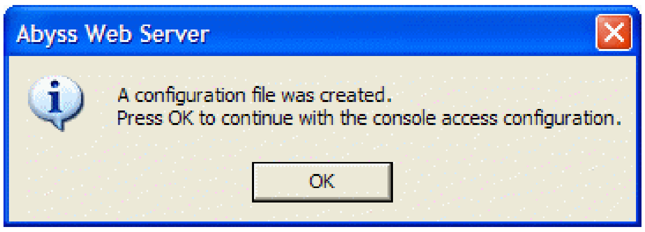
```
Abyss Web Server X1 (v 2.0.0.18)
Copyright © Aprelium Technologies - 2001-2005

-- Default host on port 80 is up and running (Local URL http://127.0.0.1)
Console local URL: http://127.0.0.1:9999
```

Figure 2-1. Abyss Web Server main window

Abyss Web Server inserts a small icon in the system tray. The icon represents a globe that spins when the server is accessed.

Figure 2-2. Abyss Web Server icon in the system tray

Abyss Web Server creates automatically a configuration file where it stores the web server's parameters. Then, it opens the browser and displays the console.

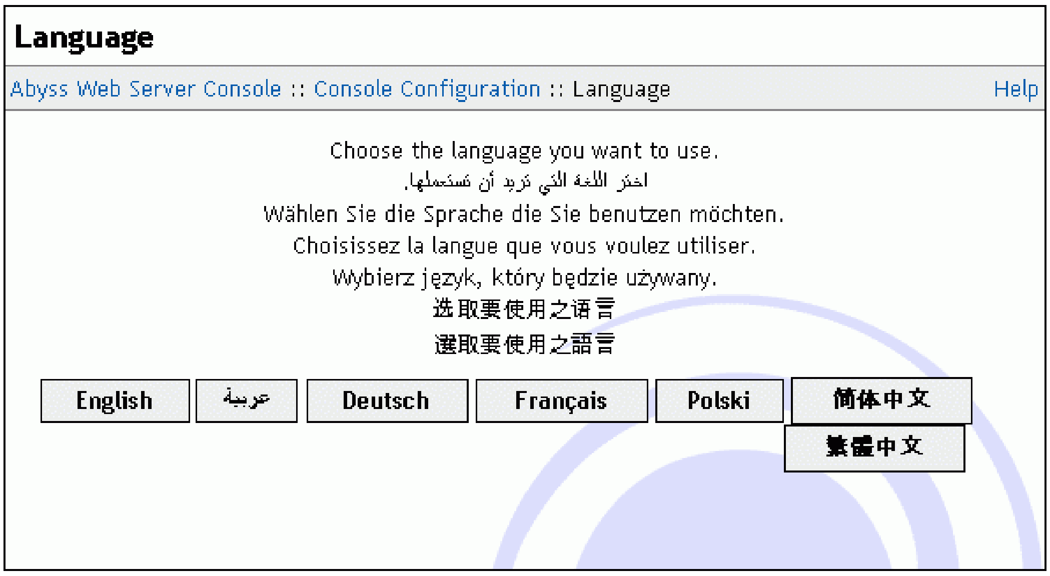


A configuration file was created.
Press OK to continue with the console access configuration.

Figure 2-3. Configuration file creation notification

Note: *Refer to the "Troubleshooting Guide" appendix to solve the most common problems that can occur with Abyss Web Server startup.*

In the displayed browser window, press a button corresponding to the language you want to use in the console.

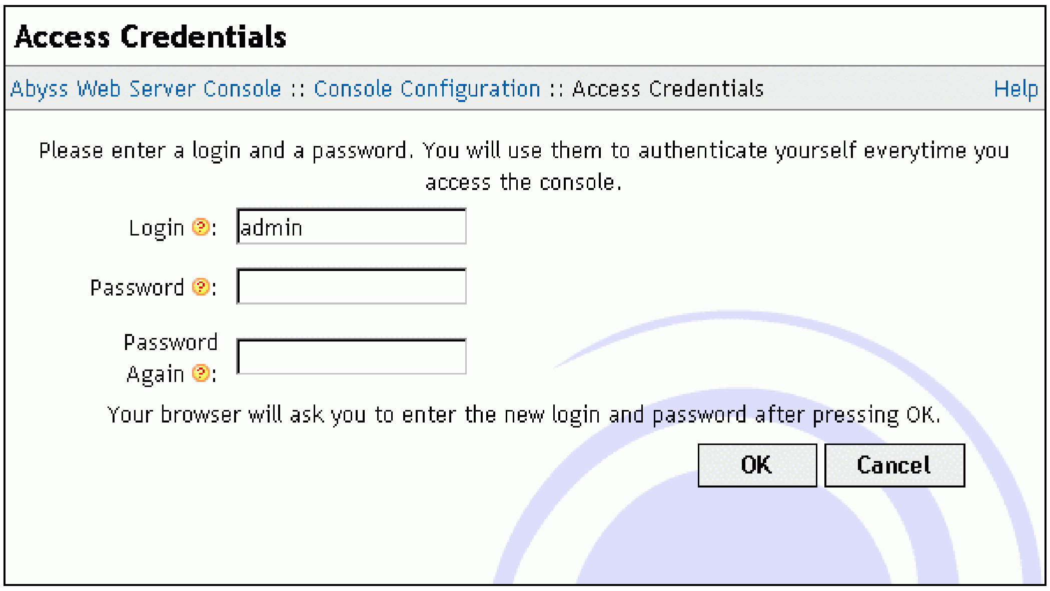

Language

Abyss Web Server Console :: Console Configuration :: Language Help

Choose the language you want to use.
اختر اللغة التي تريد أن تستعملها.
Wählen Sie die Sprache die Sie benutzen möchten.
Choisissez la langue que vous voulez utiliser.
Wybierz język, który będzie używany.
选取要使用之语言
選取要使用之語言

Figure 2-4. Console Language setup

Next, choose a login and a password and enter them in the form displayed in the browser. Press **OK** to store them and proceed to the next step.


Access Credentials

Abyss Web Server Console :: Console Configuration :: Access Credentials Help

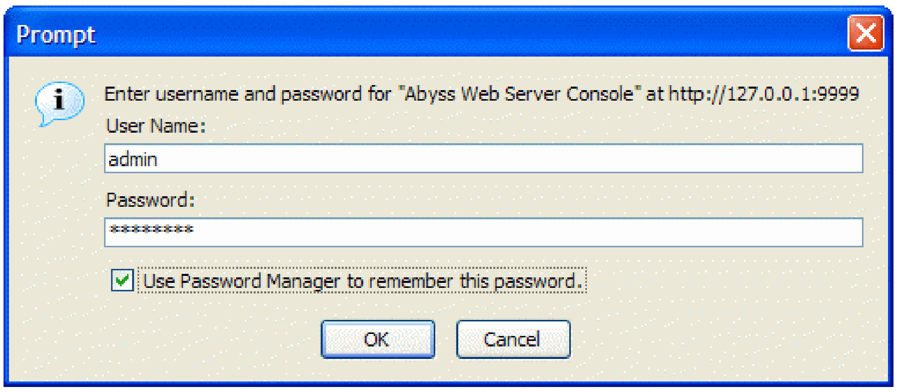
Please enter a login and a password. You will use them to authenticate yourself everytime you access the console.

Login:	admin
Password:	
Password Again:	

Your browser will ask you to enter the new login and password after pressing OK.

Figure 2-5. Console Access Credentials setup

The browser asks you for your credentials. Enter the login and the password you have already chosen and validate.



Enter username and password for "Abyss Web Server Console" at http://127.0.0.1:9999

User Name: admin

Password: *********

[✓] Use Password Manager to remember this password.

Figure 2-6. Credentials dialog

Note: *If you are using Abyss Web Server X2, you may be asked at this stage to enter your license information in the console. Follow the on-screen instructions to enter the required information. For more details, please refer to the instructions that were provided to you when downloading Abyss Web Server X2 or its updates.*

At this point, Abyss Web Server is ready to serve. To test it, point the browser to the web server URL, which is printed on the server's window . You should see the **Welcome to Abyss Web Server** page.

Note: *You can launch Abyss Web Server manually or configure it to run automatically when your computer boots up or when you start a user session in Windows.. See the "Startup Configuration" appendix for more information.*

Setting up a web site

Using your favorite web pages editor (also known as HTML editor), create a web site and put its files in the directory **`<Abyss Web Server directory>/htdocs`**. The index files must be called **`index.html`** or **`index.htm`**.

Other computers on the network can access your web site if they browse **`http://<your host name>:<host port>`**.

`<your host name>` is the host name of your computer or its IP address. Ask your network's administrator for this information. If you want to connect to the server from the computer it runs on, you can also use **`127.0.0.1`**, **`::1`**, or **`localhost`**.

`<host port>` is the number of the port the server waits for connections on. It is printed on the server's window or terminal. If **`<host port>`** is 80 (which is the default), it can be omitted from the URL and the web server can be accessed with only **`http://<your host name>`**.

Shutting down the server

Select **Exit** from the **Server** menu or close the server's main window. If the server is running, you will be asked to confirm exiting.

Asking for support

If you have questions or need help, please select **Help and Support** in the console and use one of the support links available there. Alternatively, browse the official Abyss Web Server support pages at `https://aprelium.com/support`.

Chapter 3. Using The Console

What is the console?

The console is a remote web based configuration and monitoring system. With the console, you can:

- Configure Abyss Web Server.
- Stop, run and shutdown the web server.
- View the web server's access statistics.
- Read the documentation.

Accessing the console

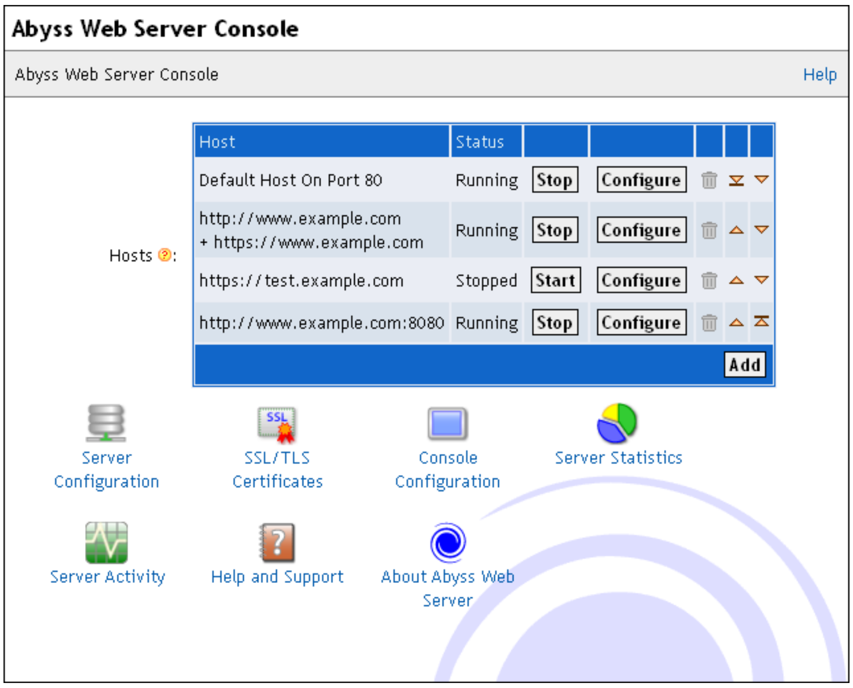
From your computer (locally)

> Browse **`http://127.0.0.1:<console port>`**, **`http://[::1]:<console port>`**, or **`http://localhost:<console port>`**.

From any computer or device connected to the network

> Browse **`http://<your host name>:<console port>`**. Note that by default, only the local access to the console is allowed so that only the computer where Abyss Web Server is installed (which IPv4 address is **`127.0.0.1`** and IPv6 address is **`::1`**) and computers connected to your LAN (which IPv4 addresses range from **`192.168.0.1`** to **`192.168.255.254`**, from **`172.16.0.1`** to **`172.31.255.254`**, and from **`10.0.0.1`** to **`10.255.255.254`**) are able to browse the console. Remote access from other computers is disabled by default. To enable it, please read "Configuring the console IP Address Control".

`<console port>` is printed on the web server's window . It is 9999 by default.


Abyss Web Server Console

Abyss Web Server Console

Host	Status					
Default Host On Port 80	Running	Stop	Configure			
http://www.example.com + https://www.example.com	Running	Stop	Configure			
https://test.example.com	Stopped	Start	Configure			
http://www.example.com:8080	Running	Stop	Configure			

Figure 3-1. Console homepage

Note: *Starting from Abyss Web Server version 2.11, the console interface features a responsive design which allows it to offer a user experience carefully adapted to the device you are using to access it, be it a mobile phone, a tablet, a laptop or a desktop computer.*

Console interface basics

Although the console has an intuitive point and click interface, some of its elements are described more in depth in this section.

Help buttons

A small tip is displayed when you put the mouse cursor on a help button. Pressing it displays the related documentation topic.

Note: *Not all the browsers support displaying tips.*

Tables

Each line of a table contains a pen button and a wastebin button. Press the pen button on a line to display a dialog where you can modify its contents or press the wastebin button to delete it. Pressing **Add** displays a dialog where you can enter the contents of a new line.

Applying configuration changes

After any configuration change, the console displays a request to restart the server. Just press the **Restart** button and wait until the server restarts.

Note: *You can accumulate many configuration changes before restarting.*

Console Configuration

To change the console parameters, open the console and select **Console Configuration**.

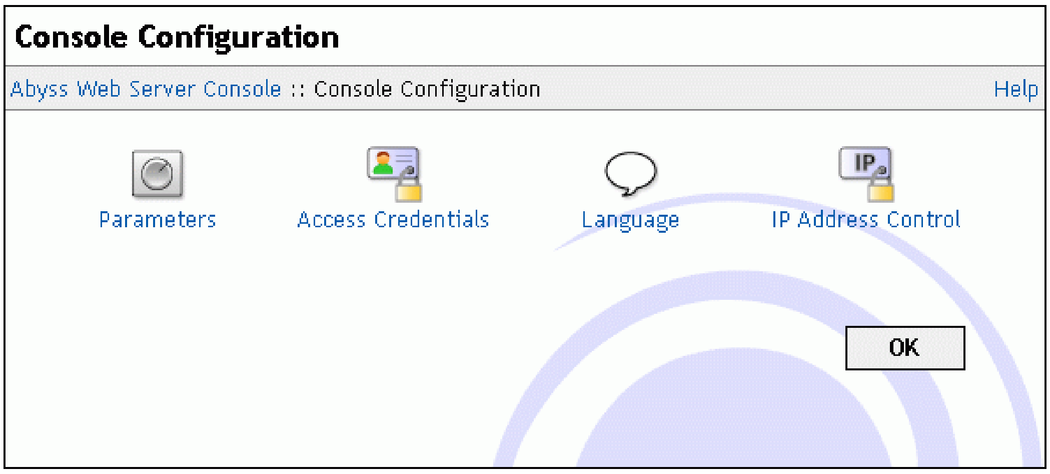
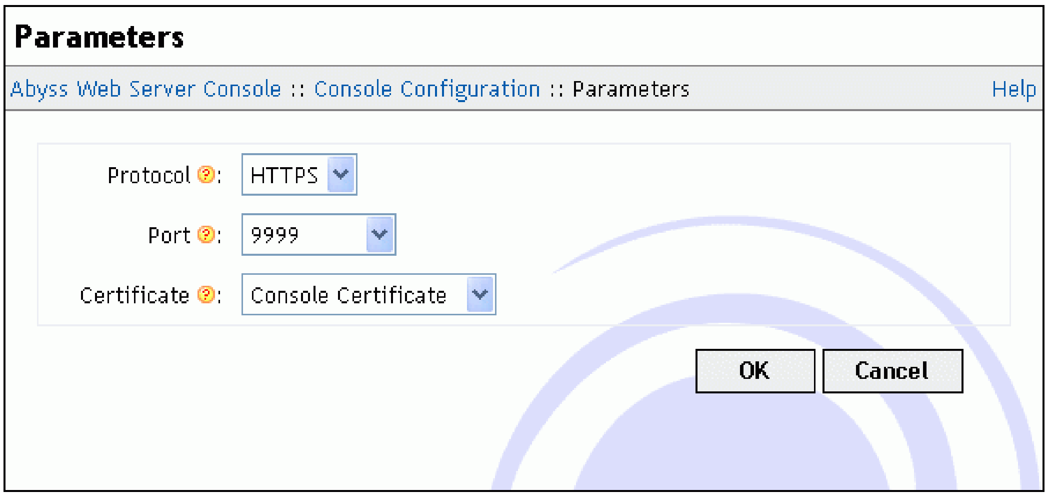


Figure 3-2. Console Configuration dialog

Changing the console port and protocol

In the **Console Configuration** dialog, select **Parameters**. In the displayed dialog, **Protocol** can be set to **HTTP** or **HTTPS**. If **HTTPS** is selected, the console will be accessible through secure connections using SSL/TLS and you will have to set **Certificate** to the name of the certificate you want to use for console accesses. A certificate that will be used for the console can be a self-signed certificate and it is recommended to set its **Host Name (Common Name)** to * to have it match with any host name you may use when accessing the console.

You can also choose a new port number that will be used for the console in the **Port** field. Press **OK** to validate the changes.


Parameters

Abyss Web Server Console :: Console Configuration :: Parameters Help

Protocol: HTTPS

Port: 9999

Certificate: Console Certificate

Figure 3-3. Console Parameters dialog

If the port number is invalid or is already used by another application on your computer, the console displays an error and asks you to enter another value.

Advanced configuration options of the console can be accessed by pressing **Edit...** next to **Advanced Parameters**. The advanced parameters dialog contains the following fields:

- **Bind to IP Address**: The IP address of the network interface that the console should listen on. When empty or set to `*`, the host listens for incoming connections on all available network interfaces (which is the default and the recommended setting.)
- **SSL/TLS Parameters**: Detailed parameters of the SSL/TLS secure layer are accessible by pressing **Edit...**.

  The displayed dialog makes the SSL/TLS profile selection possible. But it is highly recommended to let the profile set to its default and recommended value.

  If the profile is set to **Modern**, only strong ciphers and TLS 1.3 will be accepted. Selecting this profile will restrict access to modern browsers and clients which support TLS 1.3.

  If the profile is set to **Standard**, only modern ciphers with no known issues and both TLS 1.2 and 1.3 will be accepted. The **Standard** profile offers a good balance between security and reach. It will only disable access for legacy browsers and clients including Internet Explorer 10 and its previous versions which are already deprecated.

  If you set the profile to **Custom**, it becomes possible to select the accepted protocols and configure the **Ciphers** parameter to select the ciphers that the server can accept to use when negotiating with a browser establishing a HTTPS connection to the console.

  When the profile is set to **Strong**, no weak cipher (any cipher with a key length strictly less than 128 bits) will be used. But this can restrict access from old browsers which do not support modern and strong ciphers.

For custom ciphers specification, set this parameter to **Custom Specification** and fill the displayed text field with a string conforming to the https://www.openssl.org/docs/manmaster/man1/openssl-ciphers.html#CIPHER-LIST-FORMAT.

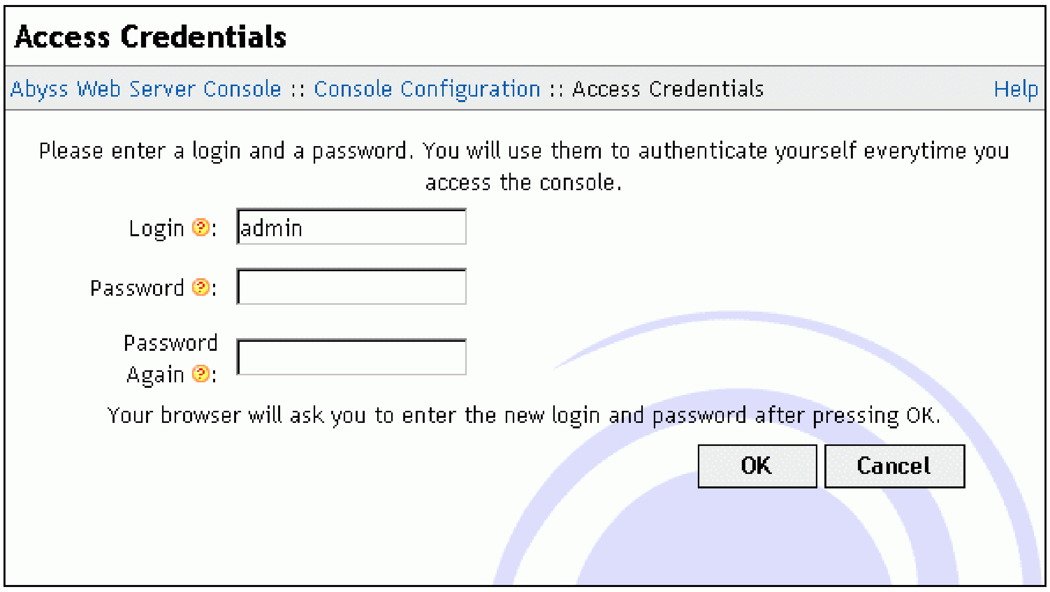
- **HTTP/2 Parameters**: HTTP/2 support in the console can be configured by pressing **Edit...**. The displayed dialog has the following fields:
  - **Disabled**: A switch to disable HTTP/2 support in the console.

Changing the console access credentials

Because the console access needs authorization, it is important to keep your console access credentials secret. It is also highly recommended not to choose them similar to your system login or password or any valuable credentials. Indeed, as the console relies on the browser to perform the authorization and as browsers use a relatively weak authorization scheme, console access information could be intercepted and deciphered by malicious persons on the network.

To change the console access credentials:

- Go to the **Console Configuration** dialog and select **Access Credentials**.
- Enter an administrator login.
- Enter a password. It is recommended to choose a password containing characters, digits, and special symbols. This password should not be a dictionary word.
- Reenter the same password.
- Press **OK**.



Access Credentials

Abyss Web Server Console :: Console Configuration :: Access Credentials

Please enter a login and a password. You will use them to authenticate yourself everytime you access the console.

Login: admin

Password:

Password Again:

Your browser will ask you to enter the new login and password after pressing OK.

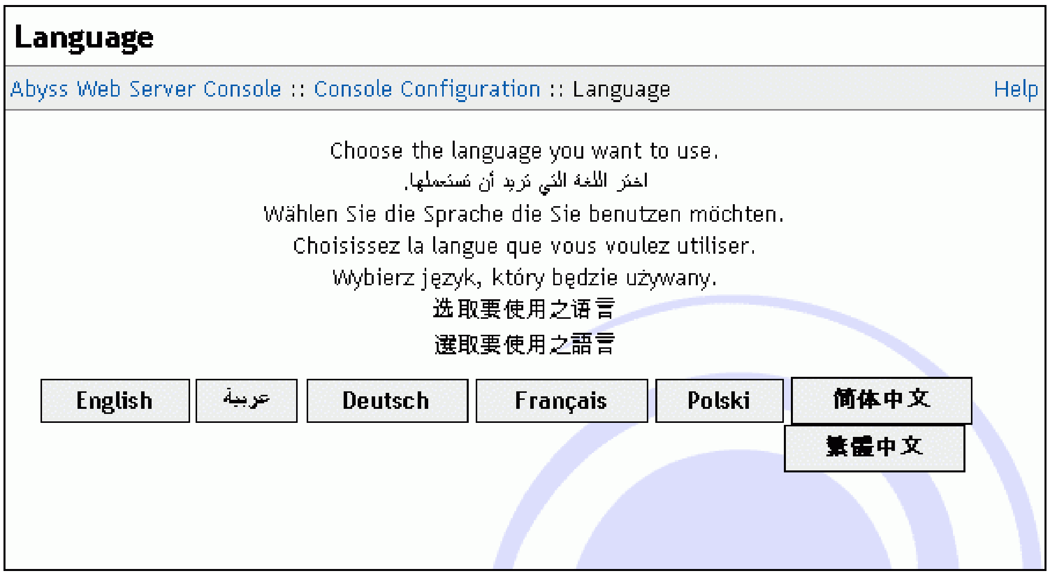
Figure 3-4. Console Access Credentials dialog

Once the change is made, the browser asks you to enter the new login and password.

Changing the language

To change the language used in the console:

- Go to the **Console Configuration** dialog and select **Language**.
- Select a language by pressing one of displayed buttons.



Language

Abyss Web Server Console :: Console Configuration :: Language

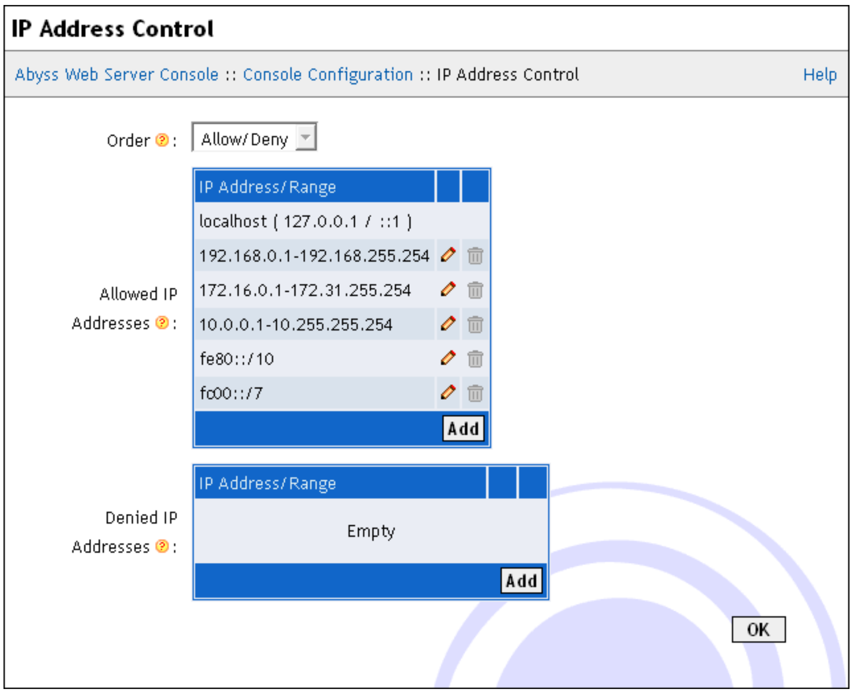
Choose the language you want to use.
اختر اللغة التي تريد أن تستعملها.
Wählen Sie die Sprache die Sie benutzen möchten.
Choisissez la langue que vous voulez utiliser.
Wybierz język, który będzie używany.
选取要使用之语言
選取要使用之語言

Figure 3-5. Console Language dialog

To add support for an additional language, download the corresponding language file from **https://aprelium.com/abyssws/languages** and save it inside **`<Abyss Web Server directory>/lang`**. You may need to restart the server to have it take into account the new language file.

Controlling the access to the console

To restrict or grant access to the console to an IP address or an IP address range, go to the **Console Configuration** dialog and select **IP Address Control**.


IP Address Control

Abyss Web Server Console :: Console Configuration :: IP Address Control

Order: Allow/Deny

Allowed IP Addresses:

IP Address/Range
localhost (127.0.0.1 / ::1)
192.168.0.1-192.168.255.254
172.16.0.1-172.31.255.254
10.0.0.1-10.255.255.254
fe80::/10
fc00::/7

Denied IP Addresses:

IP Address/Range
Empty

Figure 3-6. Console IP Address Control dialog

The displayed dialog contains the following items:

- **Order**: The order that the server follows to check if access is granted to a client based on its IP address. If it is set to **Allow/Deny**, access is denied by default and is allowed only if the IP address is in the **Allow for** list and is not in the **Deny for** list. If it is set to **Deny/Allow**, access is allowed by default and is denied only if the IP address is in the **Deny for** list and is not in the **Allow for** list.
- **Allowed IP Addresses**: The list of IP addresses and IP address ranges for which access is allowed. Refer to "IP Addresses and Ranges Format" appendix for more information about the IP addresses and ranges.
- **Denied IP Addresses**: The list of IP addresses and IP address ranges for which access is denied. Refer to "IP Addresses and Ranges Format" appendix for more information about the IP addresses and ranges.

Note: *The access to the console is always allowed for the computer on which Abyss Web Server runs (which IPv4 address is* **`127.0.0.1`** *and IPv6 address is* **`::1`***.)*

Note: *If* ***Denied IP Addresses*** *list is empty and the order is* ***Deny/Allow****, access is granted to any IP address.*

Chapter 4. Server Management

Overview

All the server management tasks can be accomplished using the console. In this section, you will learn how to:

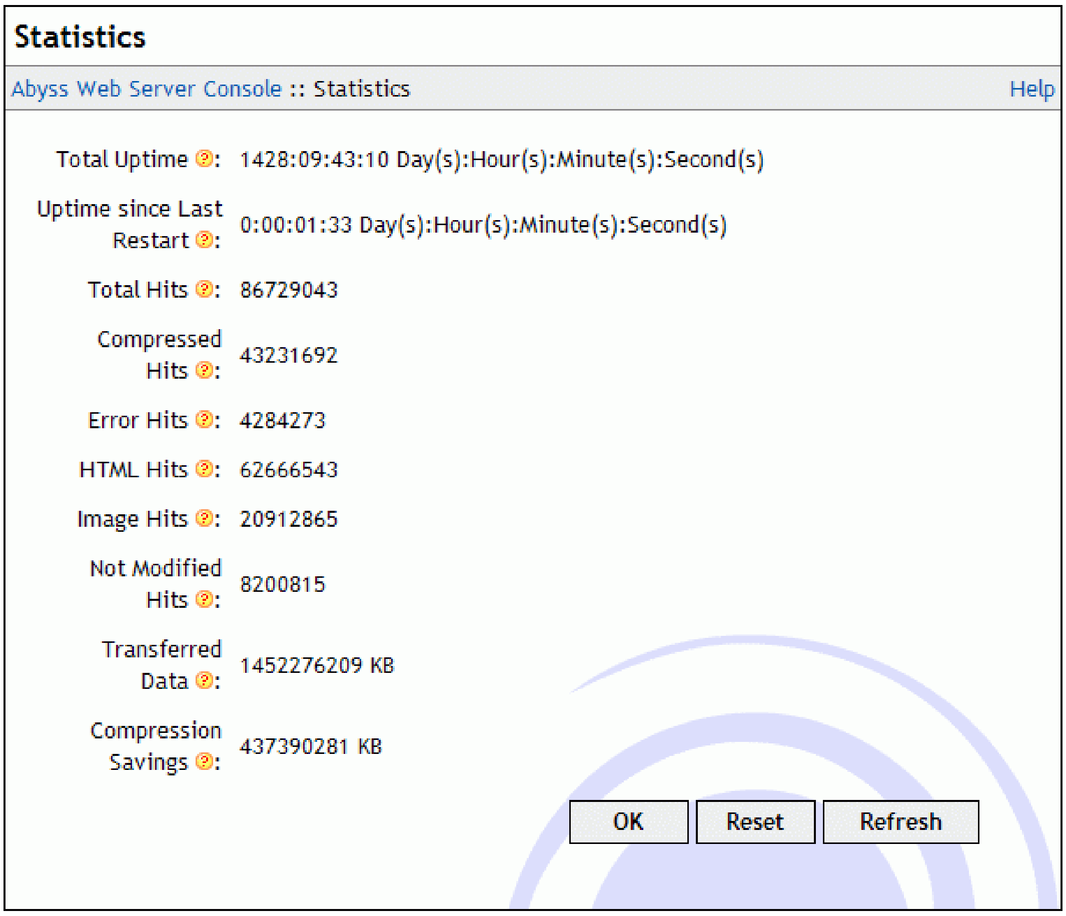
- Configure the global parameters of the web server.
- Add, delete, and configure hosts.
- Start or stop hosts.
- Monitor the accesses to the web server and to every host.
- Create, manage, and request signature of SSL/TLS certificates.

Server Statistics

Open the console and select **Server Statistics**. The console displays a dialog containing a set of statistics on the server's activity since the installation or the last server statistics reset:

- **Total Uptime**: The total time during which the server was up.
- **Uptime since Last Restart**: The time during which the server was up since the last start or restart.
- **Total Hits**: The total number of processed requests for all the hosts. It includes also bad requests that were not targeting a specific host and that resulted in an error.
- **Compressed Hits**: The number of processed requests that resulted in compressed responses across all the hosts.
- **Error Hits**: The number of requests for which the server replied by an error.
- **HTML Hits**: The number of requests the server replied to by a document which MIME type was **`text/html`**.
- **Image Hits**: The number of requests the server replied to by a document which MIME type starts with **`image/`**.
- **Not Modified Hits**: The number of requests for which the server detected that the requested document has not changed.
- **Transferred Data**: The total size of the payload sent by the server to the clients.

- **Compression Savings**: The amount of data that was saved thanks to the compressed responses sent by the server to the clients.



Statistics

Abyss Web Server Console :: Statistics

Total Uptime:	1428:09:43:10 Day(s):Hour(s):Minute(s):Second(s)
Uptime since Last Restart:	0:00:01:33 Day(s):Hour(s):Minute(s):Second(s)
Total Hits:	86729043
Compressed Hits:	43231692
Error Hits:	4284273
HTML Hits:	62666543
Image Hits:	20912865
Not Modified Hits:	8200815
Transferred Data:	1452276209 KB
Compression Savings:	437390281 KB

Figure 4-1. Server Statistics dialog

The server statistics are refreshed automatically every 10 seconds. You can also press **Refresh** for immediate refreshing. They can be reset by pressing **Reset**.

Press **OK** to go back to the main console page.

Note: *The statistics are preserved when the server is shut down.*

Note: *The server statistics are also available in real time to scripts and XSSI pages. Refer to "CGI environment variables" for more information.*

General server configuration

Open the console and select **Server Configuration** to display the general server configuration dialog.

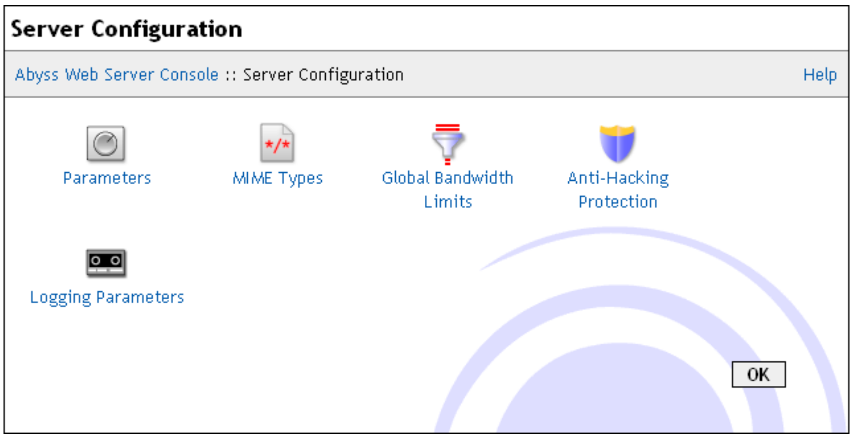
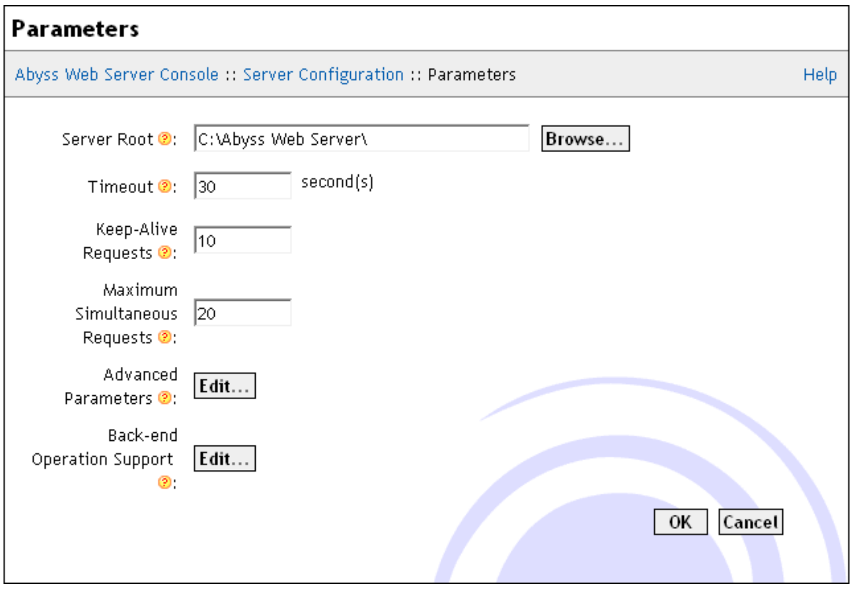


Figure 4-2. General server configuration dialog

The following subsections describe the server configuration options.

Parameters

Select **Parameters** in the **Server Configuration** dialog to display the server parameters dialog.


Parameters
Abyss Web Server Console :: Server Configuration :: Parameters
Help

Server Root: C:\Abyss Web Server\ Browse...

Timeout: 30 second(s)

Keep-Alive Requests: 10

Maximum Simultaneous Requests: 20

Advanced Parameters: Edit...

Back-end Operation Support: Edit...

Figure 4-3. Server Parameters dialog

The dialog includes the following fields:

- **Server Root**: The root path of the web server. By default, it is the directory where Abyss Web Server executable is installed. It is used as the base path for all relative real paths in the configuration.
- **Timeout**: How many seconds the server waits for an inactive connection before closing it.
- **Maximum Simultaneous Connections**: The maximum number of connections the server can serve in parallel.

- **Advanced Parameters**: Press **Edit...** to access the advanced parameters dialog. This dialog has the following fields:
  - **Maximum length of the HTTP request line**: The maximum length in bytes of the HTTP request line. Requests which do not respect this limit are refused and a HTTP error 414 (Request-URI Too Long) is reported to the client.
  - **Maximum length of the HTTP request headers**: The maximum length in bytes of all the HTTP request headers. Requests which exceed this limit are refused and a HTTP error 413 (Request Entity Too Large) is reported to the client.
  - **HTTP/1.1 Parameters**: It is possible to fine tune parameters of HTTP/1.1 connections by pressing **Edit...**. The displayed dialog has the following fields:
    - **Keep-Alive Requests**: The maximum number of requests that can be served over the same connection. Only HTTP/1.1 and some HTTP/1.0 compliant browsers can benefit from this feature.
  - **HTTP/2 Parameters**: HTTP/2 support can be configured by pressing **Edit...**. The associated dialog has the following fields:
    - **Disabled**: A global switch for disabling HTTP/2 support in all hosts and in the console.
    - **Initial Window Size**: The initial connection flow-control window size in bytes. By default, it is set to 65535 bytes. Changing it impacts initial buffer sizes on both the client and the server. It is recommended to not alter its default value unless you are fully aware of how it affects the latency and the quality of service of HTTP/2 connections.
    - **Maximum Concurrent Streams Count**: The maximum number of streams that could be multiplexed over each HTTP/2 connection. A stream is almost always associated with a HTTP request. By default, this parameter is set to 40. Do not decrease it too much if you have content heavy pages.
- **Back-end Operation Support**: When Abyss Web Server is used as a back-end for another reverse proxy, it can be configured to restore the original IP address of the reverse proxy client and other details before processing the request. This can help make the frontal reverse proxy completely transparent to Web applications running on Abyss Web Server and to the logging taking place on it.

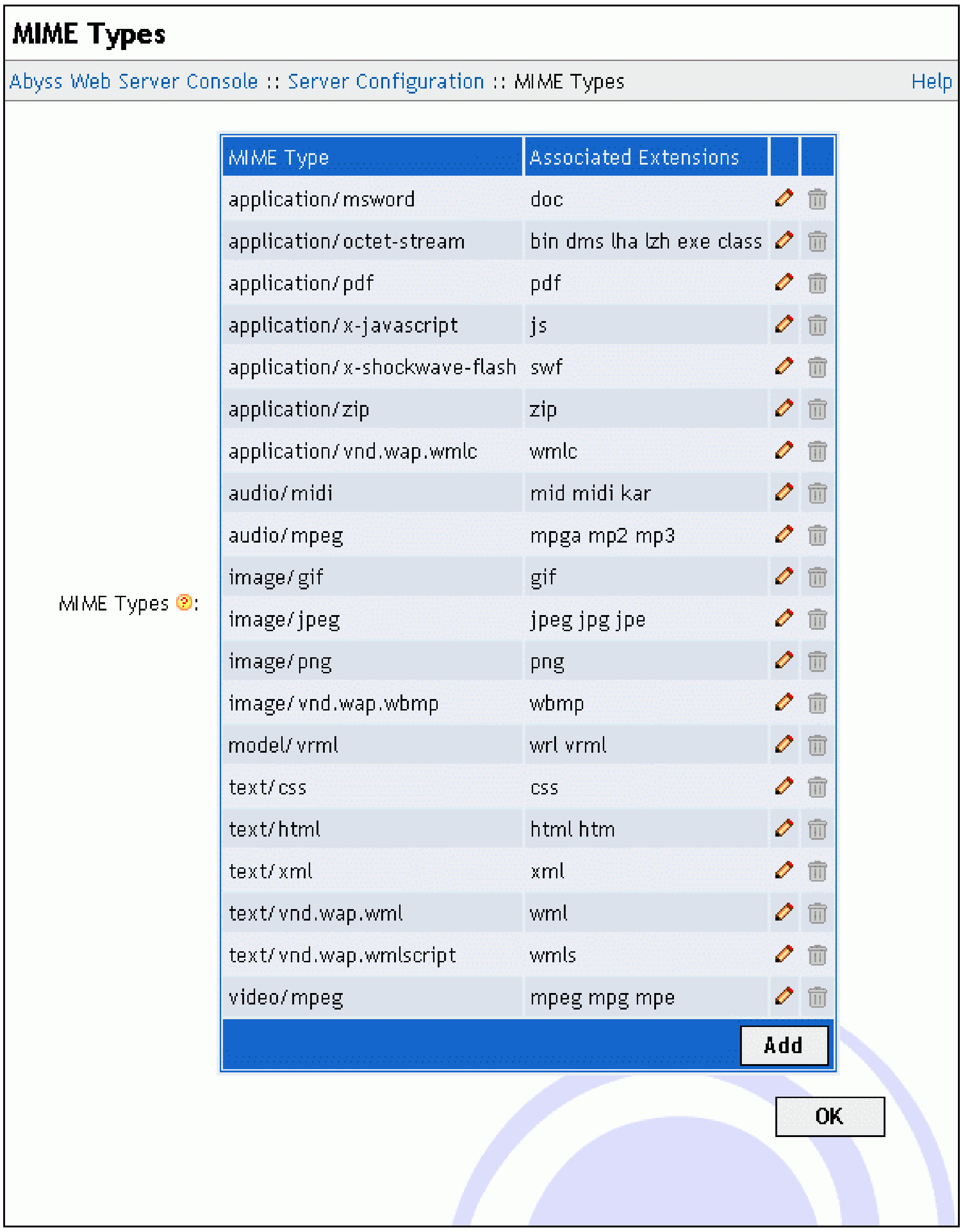
  Press **Edit...** to control the back-end operation support settings. This dialog contains the following fields:
  - **Proxies**: The list of IP addresses of computers which are recognized as reverse proxies and for which the server should act as a transparent back-end.

- **Forwarded-For Headers**: The names of the headers that should be checked for the original IP of the reverse proxy client. **`X-Forwarded-For`** and **`X-Real-IP`** are the most common ones widely used in reverse proxies.
- **Forwarded-Host Headers**: The names of the headers that should be checked for the original **`Host`** header value as the client sent it to the frontal reverse proxy. **`X-Forwarded-Host`** and **`X-Host`** are the most common header names used for that purpose. Note that when the **`Host`** value is restored, the request will be normally processed and checked against the host names of each of the hosts managed by the server.

MIME Types

When the server sends a document to a browser, it also sends its MIME type. This information helps the browser to know what kind of file it is (HTML, ZIP, JPEG image, etc…) and what to do with it (display it, save it on the disk, launch a configured application to read it, etc…). Abyss Web Server comes with a list of common MIME types used as a default.

A MIME Type has the format **`type/subtype`** and is associated to one or more extensions separated by spaces.



MIME Types

Abyss Web Server Console :: Server Configuration :: MIME Types — Help

MIME Types:

MIME Type	Associated Extensions
application/msword	doc
application/octet-stream	bin dms lha lzh exe class
application/pdf	pdf
application/x-javascript	js
application/x-shockwave-flash	swf
application/zip	zip
application/vnd.wap.wmlc	wmlc
audio/midi	mid midi kar
audio/mpeg	mpga mp2 mp3
image/gif	gif
image/jpeg	jpeg jpg jpe
image/png	png
image/vnd.wap.wbmp	wbmp
model/vrml	wrl vrml
text/css	css
text/html	html htm
text/xml	xml
text/vnd.wap.wml	wml
text/vnd.wap.wmlscript	wmls
video/mpeg	mpeg mpg mpe

Figure 4-4. MIME Types dialog

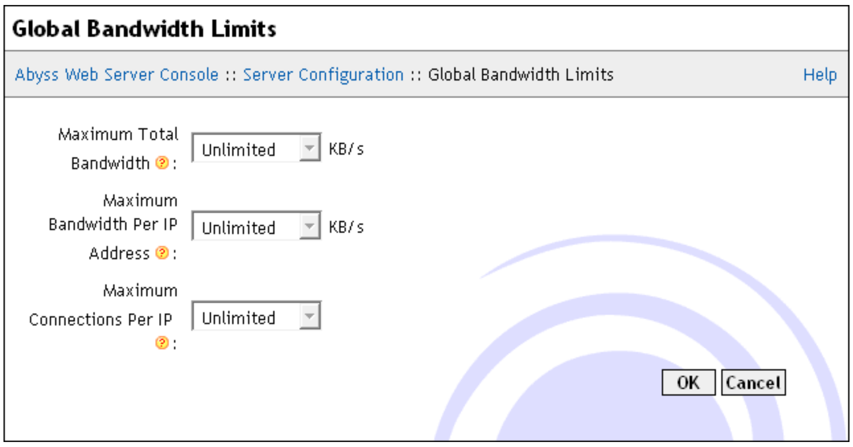
Use the **Custom MIME Types** table to edit, remove or add custom MIME types. If a MIME Type is declared in both the **Custom MIME Types** table and the **Default MIME Types** table, the custom definition prevails.

Example: *The text/html MIME type*

*HTML files have **`text/html`** as MIME type. By default, **`text/html`** is associated to extensions **`html`** and **`htm`**. As a consequence, when the server sends to a browser a document which extension is **`html`** or **`htm`**, it will also set the document's MIME type to **`text/html`**.*

Global Bandwidth Limits

Abyss Web Server gives you complete control over the bandwidth the server uses when answering to requests. Select **Global Bandwidth Limits** in the **Server Configuration** dialog to display the global bandwidth parameters dialog. For finer bandwidth settings, you should use **Bandwidth Limits** in every host configuration menu.


Global Bandwidth Limits

Abyss Web Server Console :: Server Configuration :: Global Bandwidth Limits

Maximum Total Bandwidth:	Unlimited KB/s
Maximum Bandwidth Per IP Address:	Unlimited KB/s
Maximum Connections Per IP:	Unlimited

Figure 4-5. Global Bandwidth Limits dialog

The dialog contains the following fields:

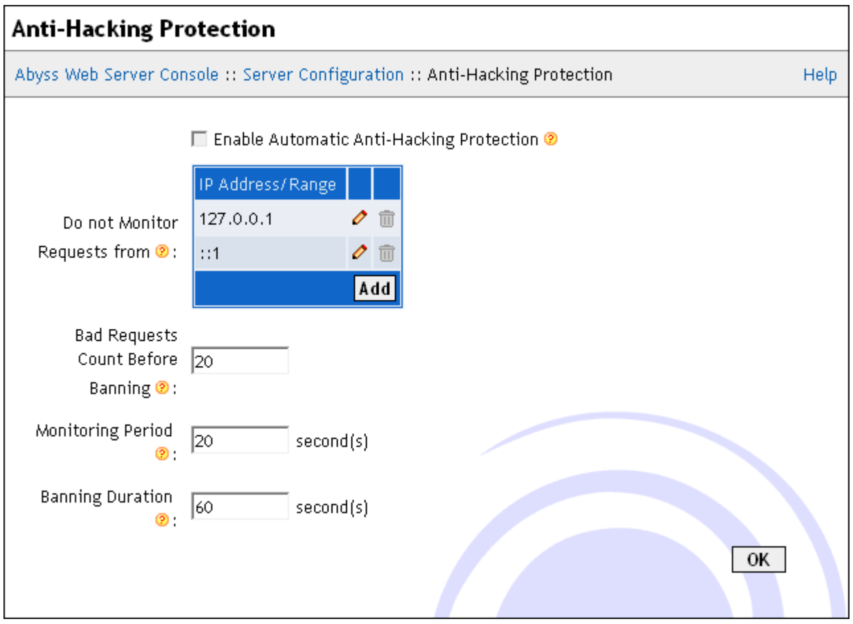
- **Maximum Total Bandwidth**: The amount of output bandwidth that the server should not exceed for all the connections. If set to **Unlimited**, the total output bandwidth will be unrestricted.
- **Maximum Bandwidth Per IP Address**: The total amount of output bandwidth that the server should not exceed for all the connections made by a single IP address. No such limit will ever be applied if this parameter is set to **Unlimited**.
- **Maximum Requests Per IP Address**: The maximum number of simultaneous requests originating from the same IP address that the server accepts to process. To disable such a limitation, set this parameter to **Unlimited**.

Note: *Abyss Web Server does always its best to deliver data to clients while respecting the restrictions set on the bandwidth. For example, assume that **Maximum Total Bandwidth** is set to **10** KB/s and **Maximum Bandwidth Per IP Address** to **4** KB/s. If there are 2 clients connecting from different IP addresses, the server allocates to each of them a bandwidth of **4** KB/s. But if there are 5 clients, the server will allocate to each of them only **2** KB/s.*

Anti-Hacking Protection

Although Abyss Web Server is secure and has an integrated system to prevent malicious accesses to the server, it was equipped with an automatic *anti-hacking protection* system to detect clients that are trying to attack the server and to ban them. This system improves the overall security, detects at an early stage *denial of service* attacks, and saves the bandwidth that could be wasted during attacks.

To configure the anti-hacking system, select **Anti-Hacking Protection** in the **Server Configuration** dialog.


Anti-Hacking Protection

Abyss Web Server Console :: Server Configuration :: Anti-Hacking Protection Help

☐ Enable Automatic Anti-Hacking Protection

Do not Monitor Requests from:

IP Address/Range		
127.0.0.1		
::1		

Add

Bad Requests Count Before Banning: 20

Monitoring Period: 20 second(s)

Banning Duration: 60 second(s)

OK

Figure 4-6. Anti-Hacking Protection dialog

The displayed dialog is made of the following items:

- **Enable Automatic Anti-Hacking Protection**: Check it to enable the automatic anti-hacking protection system.
- **Do not Monitor Requests from**: This table contains the IP addresses or IP address ranges that should not be protected against hacking. Refer to "IP Addresses and Ranges Format" appendix for more information about the IP addresses and ranges. Fill this table with trusted IP addresses only.
- **Bad Requests Count Before Banning**: The number of tolerated bad requests before considering the client as attempting to hack the server. A request that results in a reply which HTTP status code is 400 or 401 is counted as a bad request. A request that results in a reply which HTTP status code is 403, 404, 405, or 408 is counted as half a bad request.
- **Monitoring Period**: The server considers only the bad requests that are generated by a client during the last **Monitoring Period** to decide whether to ban it or not. The bigger this parameter is, the more memory the anti-hacking system needs to record all the bad requests.
- **Banning Duration**: How much time a client is banned when it is considered as hacking the server.

In other words, the anti-hacking system works as follows: If a client has generated **Bad Requests Count Before Banning** bad requests during the last **Monitoring Period** seconds, then ban it for the next **Banning Duration** seconds. When a client is banned, the server will not accept connections from it.

Note: *The server preserves the list of banned clients when it is shutdown. So if a client was banned for 2 hours at 10:00, and if the server was stopped at 10:15 and restarted at 11:00, the server will continue to not accept requests from the banned client until 12:00.*

Logging Parameters

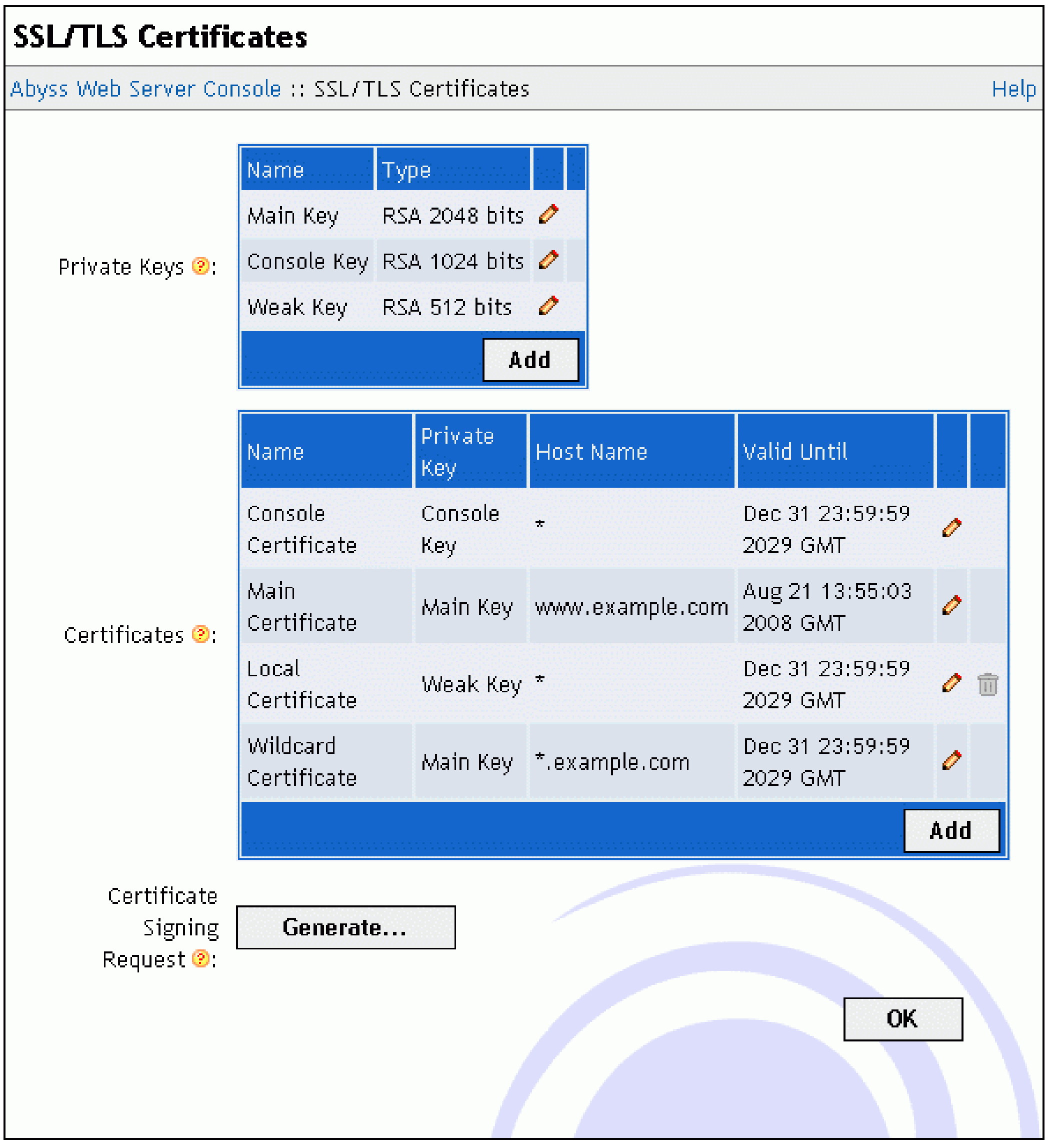
Beside the common log format and the combined log format, Abyss Web Server offers the possibility to define custom logging formats that can be used in the configuration of hosts. Select **Logging Parameters** in the **Server Configuration** dialog and use the **Custom Logging Formats** table to edit, remove or add custom logging formats definitions.

Each format is defined by its **name** and a list of fields it contains. Fields refer to information from the requests, the responses, or the CGI variables. They can also refer to general information (such as the date and time) or to static text that is to be inserted in each log line.

SSL/TLS Certificates

Open the console and select **SSL/TLS Configuration** to display the SSL/TLS certificates management interface. With it, you can:

- Create and manage private keys.
- Create self-signed certificates.
- Add certificates signed by certification authorities.
- Generate certificate signing requests (CSR).
- Declare accounts to automate fetching certificates from certification authorities which support the ACME protocol.


SSL/TLS Certificates

Abyss Web Server Console :: SSL/TLS Certificates   Help

Private Keys:

Name	Type
Main Key	RSA 2048 bits
Console Key	RSA 1024 bits
Weak Key	RSA 512 bits

Certificates:

Name	Private Key	Host Name	Valid Until
Console Certificate	Console Key	*	Dec 31 23:59:59 2029 GMT
Main Certificate	Main Key	www.example.com	Aug 21 13:55:03 2008 GMT
Local Certificate	Weak Key	*	Dec 31 23:59:59 2029 GMT
Wildcard Certificate	Main Key	*.example.com	Dec 31 23:59:59 2029 GMT

Certificate Signing Request:

Figure 4-7. Certificates management dialog

Overview

SSL/TLS is a suite of communication protocols which encrypt data exchanged between a server and a client. It is used to securely transfer HTTP requests and responses on the network. SSL/TLS helps prevent communication spying and data sniffing.

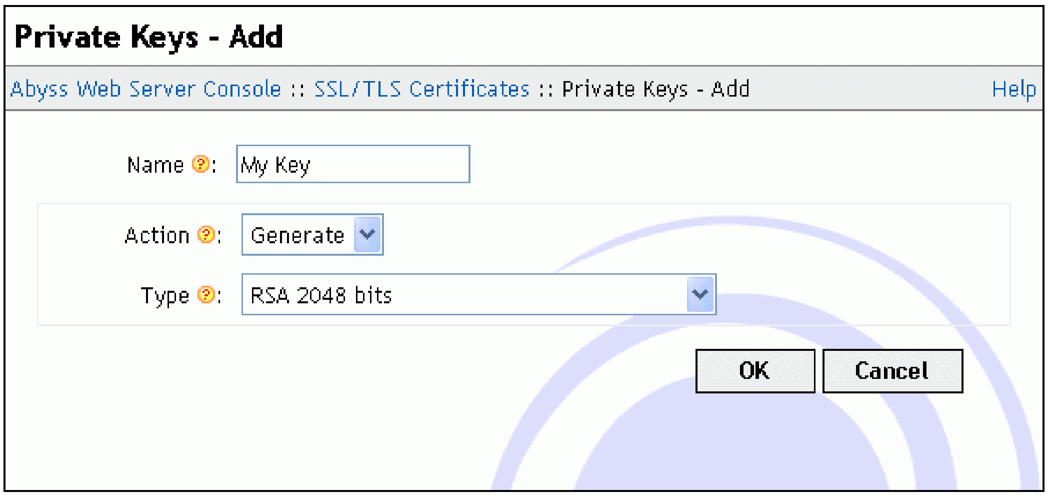
To secure one of your hosts or the console with SSL/TLS, you first need to generate a private key. A private key should never be disclosed to a thrid party as it is used to cipher (i.e. encode) transferred data.

SSL/TLS requires also that the server makes its real identity available to the client by sending a certificate as soon as a connection is established between both of them. A certificate contains information about the certificate holder. If you intend to host a business site or a site where sensitive data is expected to be gathered or displayed, we strongly recommend that the certificate is signed by an independent certification authority which will check the holder information. A certificate that is not signed by a certification authority will work but the visitors' browsers will always display a warning message and invite them to confirm that they trust your self-signed certificate.

To sign a certificate by a certification authority, you have to create a CSR (Certificate Signing Request). A CSR is generated by choosing a private key and by entering your information. Once the CSR generated, you will have to send it to a certification authority which will do the necessary to check your information and to generate a signed certificate. Note that only the CSR has to be sent to the certification authority; the private key used to generate it should never be sent to them. When you receive the signed certificate, all you have to do is to enter it in Abyss Web Server console to start using it.

Generating a private key

To create a new private key, press **Add** in the **Private Keys** table. In the displayed dialog, choose a distinctive name for the key and enter it in the **Name** field. Set **Action** to **Generate** and select the key type using the dropdown menu **Type**. When you press **OK**, the private key generation starts. It can take from a second to a minute depending on your computer speed and the type of the key you have chosen.


Private Keys - Add

Abyss Web Server Console :: SSL/TLS Certificates :: Private Keys - Add

Name: My Key

Action: Generate

Type: RSA 2048 bits

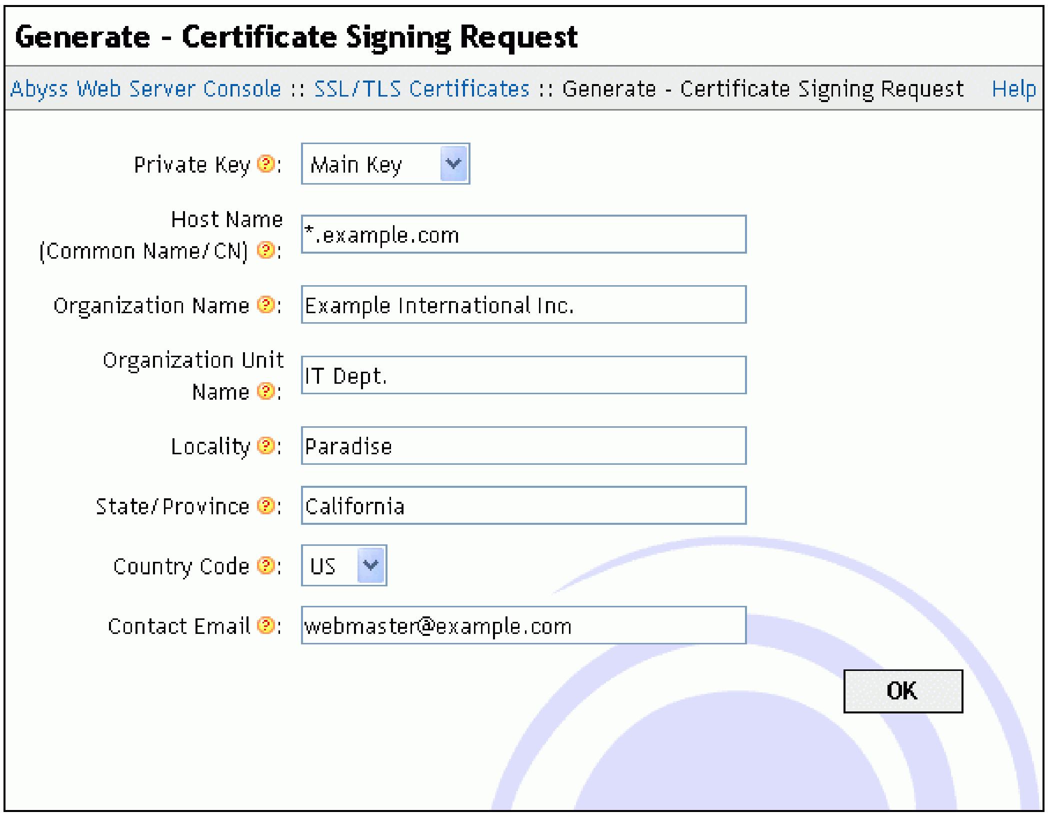
Figure 4-8. Private key generation dialog

Importing a private key

If you already have a private key and want to use it in Abyss Web Server, press **Add** in the **Private Keys** table. In the displayed dialog, choose a distinctive name for the key and enter it in the **Name** field. Set **Action** to **Import** and copy the private key text contents in the **Key Contents** text area. Note that a private key stored in a file using the PEM encoding can be opened with any text editor and its contents copied then pasted in **Key Contents** to import it.

Generating a CSR

To obtain a certificate signed by a certification authority, press **Generate** in front of **Certificate Signing Request**. In the displayed dialog, choose a private key that the certificate will be based on. Next fill the listed fields with your information. Note that you must enter accurate information as most certification authorities will verify them before issuing the signed certificate. Keep the **Secure Hash Algorithm** to its default unless instructed by your certification authority to use a specific signature algorithm or need to interoperate with old clients which only understand SHA1 (deprecated as of 2015 and planned to be phased out by most browsers by 2016.) Press **OK** when you are done. The console will then display the CSR contents in a text area. Depending on your certification authority, you may have to send it, copy it in an online form, or put it in a text file and forward it to them. We strongly recommend that you check with your certification authority about the best way to provide them with the CSR.


Generate - Certificate Signing Request
Abyss Web Server Console :: SSL/TLS Certificates :: Generate - Certificate Signing Request
Help
OK

Private Key:	Main Key
Host Name (Common Name/CN):	*.example.com
Organization Name:	Example International Inc.
Organization Unit Name:	IT Dept.
Locality:	Paradise
State/Province:	California
Country Code:	US
Contact Email:	webmaster@example.com

Figure 4-9. CSR generation dialog

__Note:__ The __Host name (Common Name)__ must be filled with the name of the host which will use the certificate. If the host name is **`www.example.com`**, *that field should contain* **`www.example.com`** *and not only* **`example.com`**.

__Note:__ Some certification authorities support wildcard certificates. In such a case, you can enter in that field **`*.example.com`** *which will create a CSR for a certificate that will be valid for* **`www.example.com`**, **`test.example.com`**, *or* **`mail.example.com`**. *However that certificate will not be valid for* **`example.com`** *or* **`test.mail.example.com`**.

Note: *Some certification authorities may also support certificates with more than one host name. To generate a CSR for such a certification authority, enter in the* ***Host name (Common Name)*** *all the host names separated with spaces. For example, if a certificate is to be associated with both* **`test.example.com`** *and* **`mail.example.com`***, enter* **`test.example.com mail.example.com`***.*

Note: *Some certification authorities will ask you about your server type. This information is mainly used for statistical purposes and makes no difference on the final signed certificate they will deliver. If you do not find Abyss Web Server on their list, select* ***Other Web Server*** *or* ***Other****. If no such choices are available, you can select* ***OpenSSL*** *or* ***OpenSSL-based server****. Again if no such choices are available, you can safely select* ***Apache*** *or* ***ModSSL*** *as our SSL/TLS implementation is based on OpenSSL which is also used by Apache and ModSSL.*

Note: *Abyss Web Server accepts IDN (International Domain Names) in the* ***Host name (Common Name)*** *field. It will recognize them and convert them to the proper format for use inside a CSR.*

Adding a signed certificate

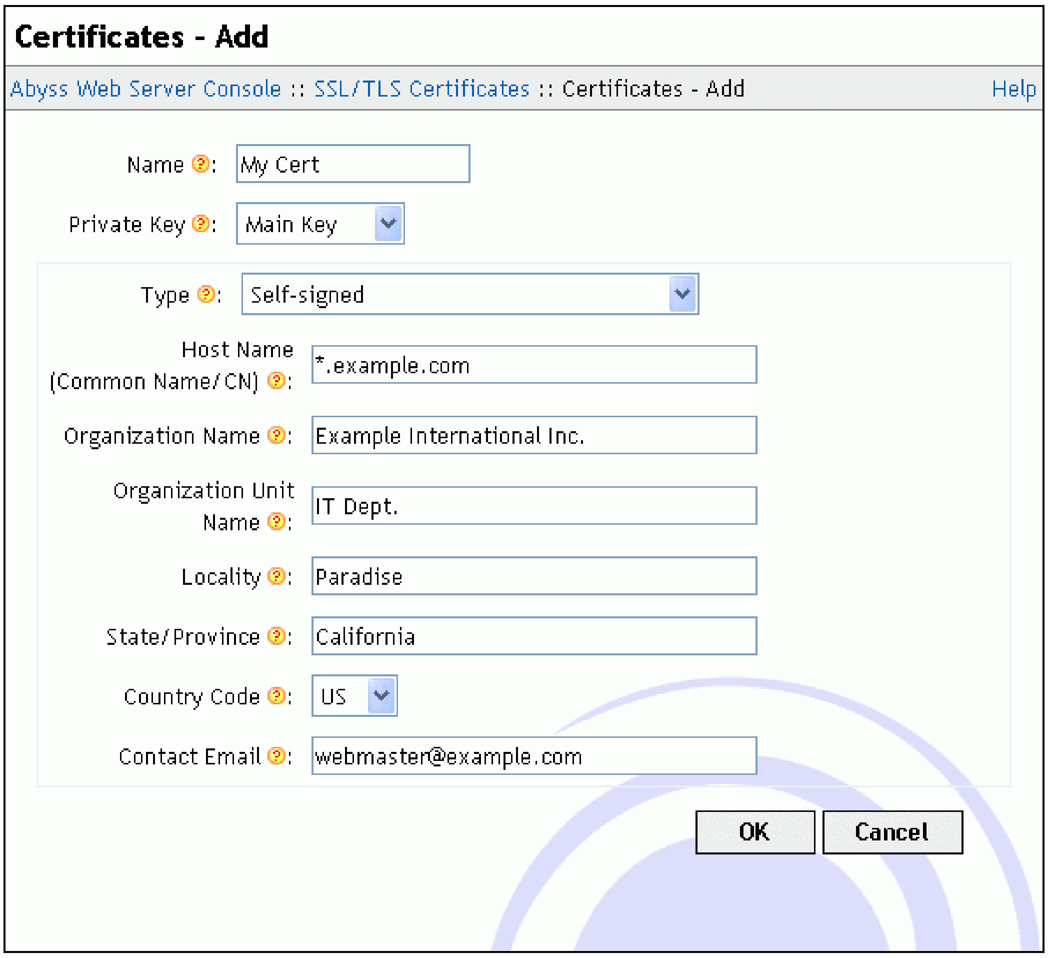
To import a certificate signed by a certification authority into Abyss Web Server, press **Add** in the **Certificate Store** table. In the displayed dialog, choose a name for the new certificate and enter it in **Name**. Set **Private Key** to the private key that the certificate is based on: It is the same private key that you selected when generating the CSR associated with that certificate. Next set **Type** to **Signed by a Certification Authority (CA)**. Enter the main certificate in **Main Certificate**. If it was delivered in a file, open it with a text editor and copy its contents to **Main Certificate**.

If the certification authority provided you with additional certificates that are necessary to establish the trust chain, they must be entered in the **Intermediate Certificates**. If more than a single intermediate certificate is available, enter their contents one after the other in that field.

The last field **CA Root Certificate** must be filled with the CA (Certification Authority) or root certificate if available. Press **OK** to validate the new certificate.

Creating a self-signed certificate

If your use of SSL is limited or if you do not mind having your visitors get a warning from their browser about your certificate each time they access your site, you can generate a self-signed certificate using the console.


Certificates - Add

Abyss Web Server Console :: SSL/TLS Certificates :: Certificates - Add Help

Name:	My Cert
Private Key:	Main Key
Type:	Self-signed
Host Name (Common Name/CN):	*.example.com
Organization Name:	Example International Inc.
Organization Unit Name:	IT Dept.
Locality:	Paradise
State/Province:	California
Country Code:	US
Contact Email:	webmaster@example.com

Figure 4-10. Self-signed certificate creation dialog

To generate a self-signed certificate, press **Add** in the **Certificate Store** table. In the displayed dialog, choose a name for the new certificate and enter it in **Name**. Use **Private Key** to select the private key that the certificate will be based on. Next set **Type** to **Self-Signed Certificate**. Fill the information fields with your details. Keep the **Secure Hash Algorithm** to its default unless you need to test a specific signature algorithm or need to interoperate with old clients which understand only SHA1. Finally press **OK** to create the certificate.

***Note:** A site using a self-signed certificate offers the same security as a site using a certificate signed by a certification authority since the encryption is private key dependent only. But consider that self-signed certificates cannot be trusted by visitors who do not know you. So use them only for tests or for sites which access is limited and which visitors trust you (for example in an Intranet or for a family Web site).*

Getting certificates from an ACME certification authority

The Automatic Certificate Management Environment (ACME) protocol automates the communication between a Web server and a certificate authority to fetch an install valid SSL/TLS certificates with no or very little user interaction.

ACME-Bot

ACME-Bot is an internal component of Abyss Web Server. It is responsible of interacting with ACME compliant certification authorities such as https://letsencrypt.org which delivers free SSL/TLS certificates trusted by all modern browsers and HTTPS clients. ACME-Bot installs obtained certificates and renews those near expiry. It also handles HTTP-01 challenges to prove domain ownership: For that end, ACME-Bot may create a responder host listening on port 80 (by default) to reply to ACME certification authorities challenges in an automatic way.

ACME-Bot current status and on-going activities are viewed by pressing **View...** in the **ACME-Bot Status** field. The console may also display notification about ACME-Bot errors or request your assistance to validate the ownership of a domain name for which a certificate will be delivered.

Creating an ACME account

To enable automatic generation of certificates from one or more hosts in Abyss Web Server, you should first create a new private key as detailed in "Generating a private key". Next, press **Edit...** in the **ACME-Bot Parameters** field. Then press **Add** in the **ACME Accounts** table. In the displayed dialog, choose a name for the new account and enter it in **Name**.

Use the **Directory URL** field to select the entry point of the ACME certification authority. Abyss Web Server comes preloaded with the default entry points for both regular and staging variants of https://letsencrypt.org (Version 2). Staging should only be used for tests as the certificates it generates are self-signed by the certification authority and not trusted by any browser.

Use **Private Key** to select the private key that the account will use when communicating with the ACME certification authority. It is highly recommended that this key is exclusively used for that purpose and not shared with another certificate you have.

Fill **Contact Email** with your email address which will be used to create the account with the ACME certification authority.

Once all the fields are filled, press **OK** to save the account information. This new account can later be used in any host to have the ACME-Bot automatically order, ftech, and install certificates.

Fine-tuning ACME-Bot

The dialog **ACME-Bot Parameters** exposes other fields meant for configuring certificate renewal and installation as well as debugging communication with ACME certification authorities:

- **Renew ACME certificates before their expiry by**: Controls the time by which ACME-Bot should place an order for the renewal of a certificate fetched from an ACME certification authority.

- **Force a server restart to use a new ACME certificate before the old one's expiry by**: Controls the time by which ACME-Bot should force a restart to have the server load a new certificate fetched from an ACME certification authority.
- **Retry failed orders due to network errors every**: The duration ACME-Bot waits before retrying an operation after a network failure.
- **Log File**: The path of the log file for ACME-Bot. If it is relative, it is considered as a subpath of the server root. If empty, logging is disabled.
- **Debugging Level**: The type of information ACME-Bot should log. It should be set to **Errors** unless you are debugging a problem with an ACME certification authority or trying to have more information about the internals of the ACME protocol and ACME-Bot implementation.

Chapter 5. Hosts Management

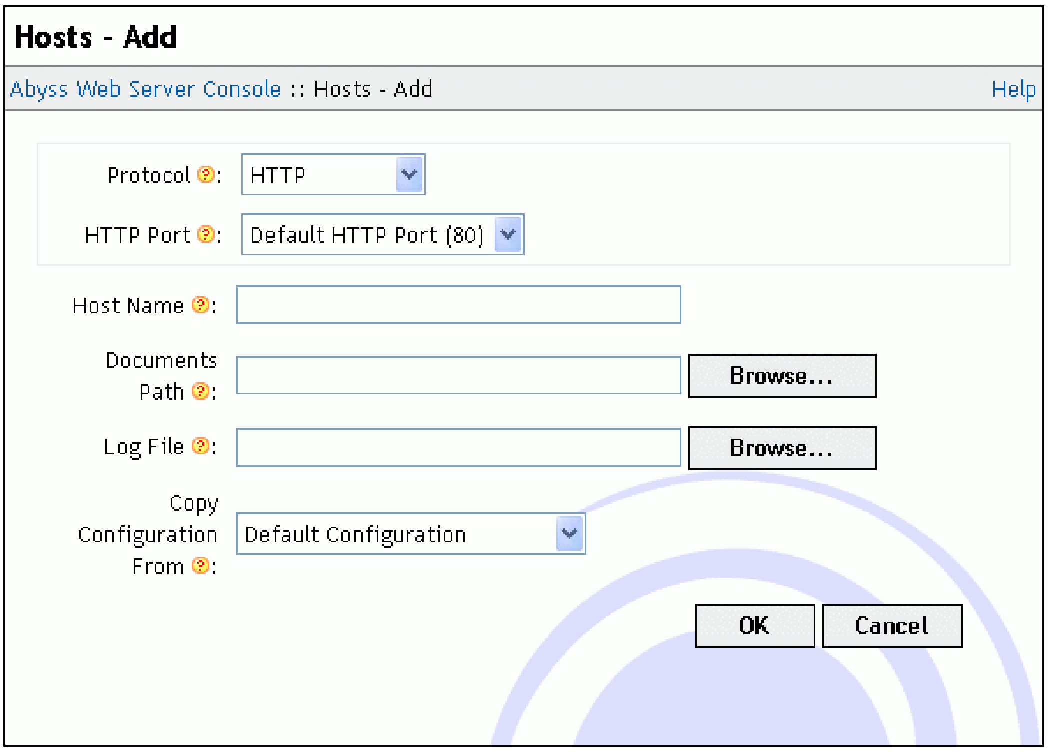
Overview

The main console dialog contains a table listing all the declared hosts in the server. Each line in this table is associated with a host and contains:

- Its name or identifier
- Its status: It can be **Running**, **Stopped**, or **Error** when the host cannot be started because of a configuration error; in such a case, click **Error** to have detailed information about the problem.
- A **Start** or **Stop** button.
- A **Configure** to access the host configuration dialog.

Declaring a new host

To add a new host to the web server, go to the main console dialog and press **Add** in the **Hosts** table.


Hosts - Add

Abyss Web Server Console :: Hosts - Add   Help

Protocol: HTTP

HTTP Port: Default HTTP Port (80)

Host Name:

Documents Path:

Log File:

Copy Configuration From: Default Configuration

Figure 5-1. New host dialog

The displayed dialog contains the following fields:

- **Protocol**: Set it to **HTTP** if the host will use normal non-secure connections. If you want to have the host serve content over secure connections only using SSL/TLS, set it to **HTTPS**. You can also choose to have the host serve content on both secure and non-secure connections by setting **Protocol** to **HTTP+HTTPS**. In such a situation, you can configure some virtual paths to be accessible on secure connections only by adding them to the **Exclusively Serve On HTTPS** table.
- **HTTP Port**: The port on which the new host will wait for HTTP connections. Its default value is 80. This field is not available if the **Protocol** is set to **HTTPS** only.
- **HTTPS Port**: The port on which the new host will wait for secure HTTPS connections. Its default value is 443. This field is not available if the **Protocol** is set to **HTTP** only.
- **Certificate Type**: The origin of the certificate that is used for secure connections with the new host. The certificate can be retrieved from the certificate store (a self-signed certificate or a certificate from a certificate authority) or generated by an ACME server using a declared ACME account. This field is not available if the **Protocol** is set to **HTTP** only.
- **Certificate**: The name of certificate from the certificate store that will be used for secure connections with the new host. Note that the certificate's **Host Name (Common Name)** should be equal to or match with the names associated with the new host. Otherwise, visitors' browsers will report a warning about mismatched names on every access to the host. This field is available only if **Certificate Type** is set to **From certificate store**.
- **ACME Account**: The name of the ACME account which ACME-Bot uses to request a certificate for secure connections with the new host. This field is available if **Certificate Type** is set to **From ACME account**.
- **Host Name**: The main name of the new host. If the field is empty, the host will not be associated with any name and will answer to all requests that reach it on the configured port. The host can also have more than one name: additional names should be declared later in the **Host Names** table in the **General** dialog related to the host. The host name may be an Internation Domain Name (IDN) containing non-Latin characters. The host name can also be a pattern such as `*.mysite.com`. Refer to the "Patterns Format" appendix for more information about patterns.
- **Documents Path**: The path of the web site files. If it is a relative path then it is considered as a subpath of the server root.
- **Log File**: The path of the log file. If it is relative, it is considered as a subpath of the server root. If empty, logging is disabled.
- **Copy Configuration From**: The configuration of the new host is cloned from the configuration of already defined hosts or from the default configuration.

Note: *Adding a new host is only possible in Abyss Web Server X2. Abyss Web Server X1 can only manage a single host.*

Configuring a host

To open the configuration dialog of a host, go to the main console dialog and press the **Configure** button located on the same line as the host name in the **Hosts** dialog.

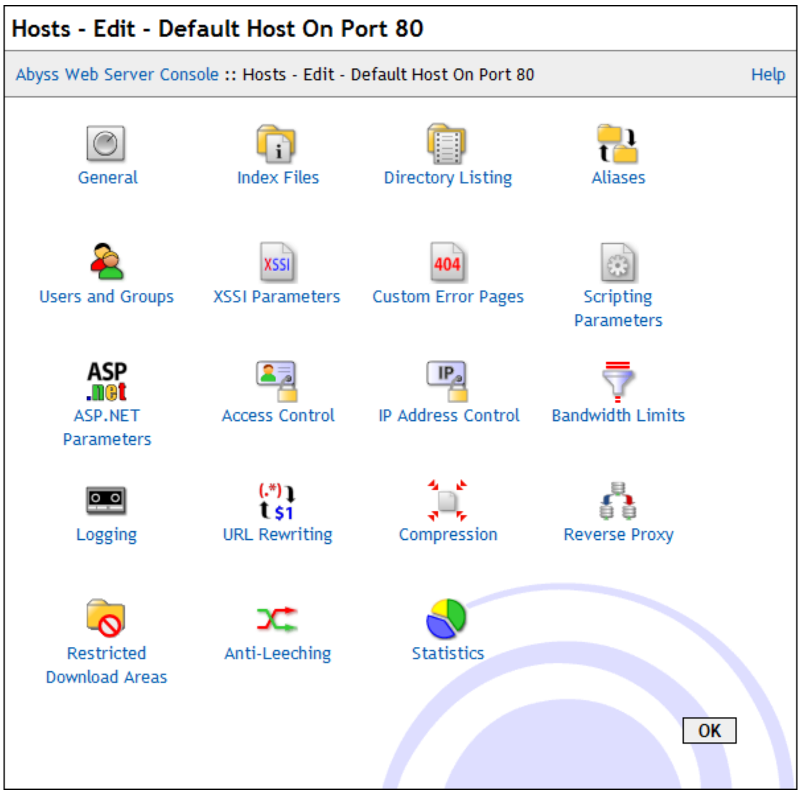
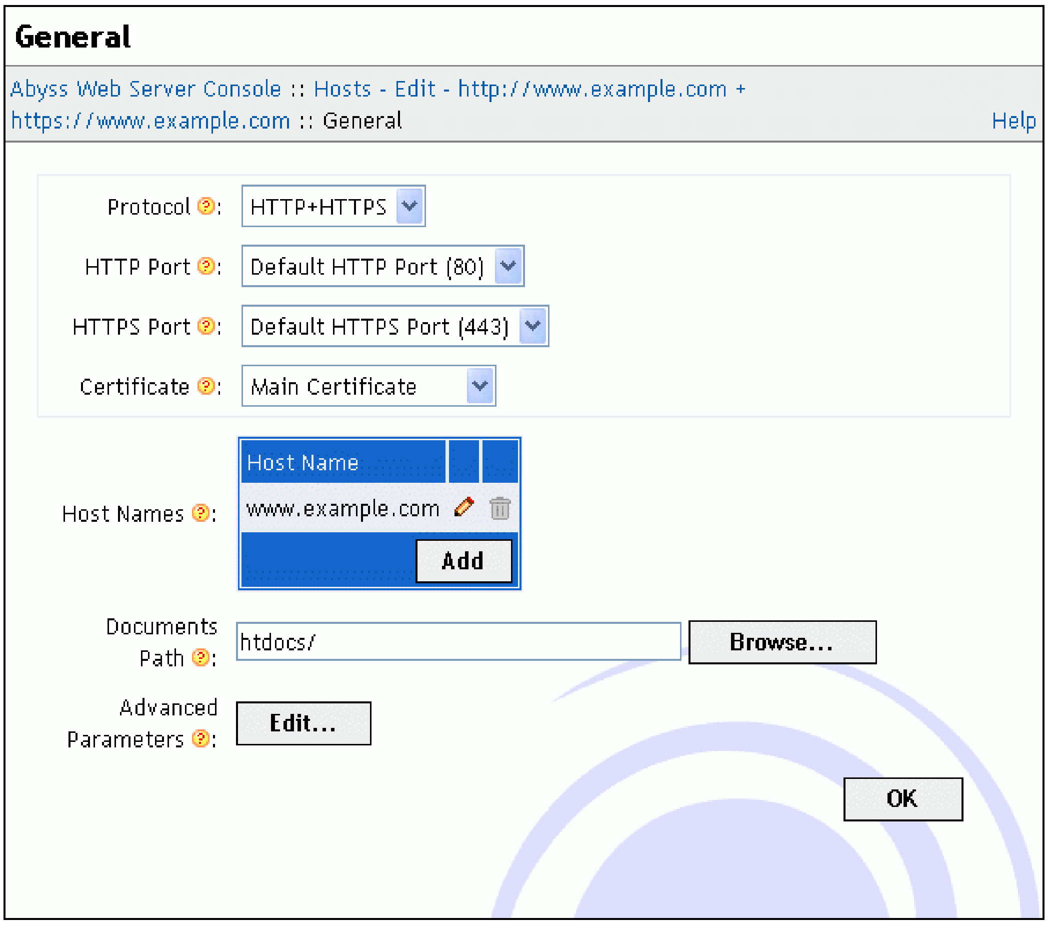


Figure 5-2. Host configuration dialog

The following subsections describe all the host configuration options.

General

The **General** dialog contains the general configuration parameters of a host.



General

Abyss Web Server Console :: Hosts - Edit - http://www.example.com + https://www.example.com :: General

Help

Protocol: HTTP+HTTPS

HTTP Port: Default HTTP Port (80)

HTTPS Port: Default HTTPS Port (443)

Certificate: Main Certificate

Host Names:

Host Name
www.example.com

Add

Documents Path: htdocs/

Browse...

Advanced Parameters: Edit...

OK

Figure 5-3. General dialog

This dialog has the following elements:

- **Documents Path**: The path of the web site files. If it is a relative path then it is considered as a subpath of the server root.
- **Protocol**: Set it to **HTTP** to configure the host to use normal non-secure connections. To have the host serve content over secure connections only using SSL/TLS, set it to **HTTPS**. The host can be configured to serve content on both secure and non-secure connections by setting **Protocol** to **HTTP+HTTPS**. In such a situation, you can configure some virtual paths to be accessible on secure connections only by adding them to the **Exclusively Serve On HTTPS** table.
- **HTTP Port**: The port on which the host waits for HTTP connections. Its default value is 80. This field is not available if the **Protocol** is set to **HTTPS** only.
- **HTTPS Port**: The port on which the host waits for secure HTTPS connections. Its default value is 443. This field is not available if the **Protocol** is set to **HTTP** only.
- **Certificate Type**: The origin of the certificate that is used for secure connections with the current host. The certificate can be retrieved from the certificate store (a self-signed certificate or a certificate from a certificate

authority) or generated by an ACME server using a declared ACME account. This field is not available if the **Protocol** is set to **HTTP** only.

- **Certificate**: The name of certificate from the certificate store that is used for secure connections with the current host. Note that the certificate's **Host Name (Common Name)** should be equal to or match with the names associated with the current host. Otherwise, visitors' browsers will report a warning about mismatched names on every access to the host. This field is available only if **Certificate Type** is set to **From certificate store**.
- **ACME Account**: The name of the ACME account which ACME-Bot uses to request or renew a certificate for secure connections with the current host. This field is available if **Certificate Type** is set to **From ACME account**.
- **Host Names**: This table contains the names associated with the current host. If the table is empty, the host is not associated with a specific name and will answer any request that reaches it on the configured port. A host can also have one or more names. Names may contain non-Latin characters ans Abyss Web Server supports International Domain Names (IDN) natively. A host name in this table can also be a pattern such as **`*.mysite.com`**. Refer to the "Patterns Format" appendix for more information about patterns.
- **Advanced Parameters**: Press **Edit...** to open the advanced parameters dialog which contains the following fields:
  - **Bind to IP Address**: The IP address of the network interface that the host should listen on. When empty or set to `*`, the host listens for incoming connections on all available network interfaces (which is the default and the recommended setting.)
  - **SSL/TLS Parameters**: Detailed parameters of the SSL/TLS secure layer are accessible by pressing **Edit...**.

    In the displayed dialog, SNI support (which is enabled by default) can be disabled. SNI stands for Server Name Indication and is an extension to the SSL/TLS protocol to allow virtual hosting on SSL/TLS. The original SSL protocol limited each pair of IP address and port to have a single certificate. This prevented HTTPS ports sharing. With SNI, cetificate negotiation became possible and legacy SSL restrictions are now lifted.

    The displayed dialog allows also changing the SSL/TLS profile. But it is highly recommended to let it set to its default and recommended value.

    If the profile is set to **Modern**, only strong ciphers and TLS 1.3 will be accepted. Selecting this profile will restrict access to modern browsers and clients which support TLS 1.3.

    If the profile is set to **Standard**, only modern ciphers with no known issues and both TLS 1.2 and 1.3 will be accepted. The **Standard** profile offers a good balance between security and reach. It will only disable access for legacy browsers and clients including Internet Explorer 10 and its previous versions which are already deprecated.

If you set the profile to **Custom**, it becomes possible to select the accepted protocols and configure the **Ciphers** parameter to select the ciphers that the server can accept to use when negotiating with a browser establishing a HTTPS connection to the console.

When the profile is set to **Strong**, no weak cipher (any cipher with a key length strictly less than 128 bits) will be used. But this can restrict access from old browsers which do not support modern and strong ciphers.

For custom ciphers specification, set this parameter to **Custom Specification** and fill the displayed text field with a string conforming to the https://www.openssl.org/docs/manmaster/man1/openssl-ciphers.html#CIPHER-LIST-FORMAT.

The **ACME Parameters** are also available to select the validation modes of the HTTP-01 and DNS-01 challenges and to allow storing a copy of an ACME-obtained certificate in a specified file for sharing with an external program. The HTTP-01 challenge is by default set to **Automatic validation**. If for some reason, port 80 cannot be accessed by the external ACME server to validate the ownership of a domain name, it is possible to set the challenge mode to **Automatic validation with custom configuration** to fine tune the binding IP address as well as the port used for validation (although that port has to be accessed externally as port 80 - port-forwarding on your router should be properly configured in such a case.) Both HTTP-01 and DNS-01 challenges support a manual validation mode where it is possible to perform the validation by following the ACME-Bot instructions on its status page on the console. HTTP-01 or DNS-01 challenges can also be disabled.

- **HTTP/2 Parameters**: HTTP/2 support in the current host can be configured by pressing **Edit...**. The displayed dialog has the following fields:

  - **Disabled**: A switch to disable HTTP/2 support in the current host. Note that even though this box is unchecked, the server can still not accept HTTP/2 connections because HTTP/2 support is disabled at the global level or because the current host is not configured to accept HTTPS secure connections.

  - **HTTP/1.1 Required**: If a HTTP/2 stream carries a request which path matches one of the virtual paths listed in this table, it is aborted with error code *HTTP1_1_REQUIRED*. This forces the browser to retry the request with HTTP/1.1 and to no more use HTTP/2 for that host during the rest of the session.

    By default, this table contains a pattern which matches with the naming convention of legacy NPH (Non Parsed Headers) scripts. For more details, refer to "CGI Internals".

- **Disable Download Resuming for**: Download resuming (through the **`Range`** HTTP header) is disabled for requests which path matches one of the virtual paths listed in this table.

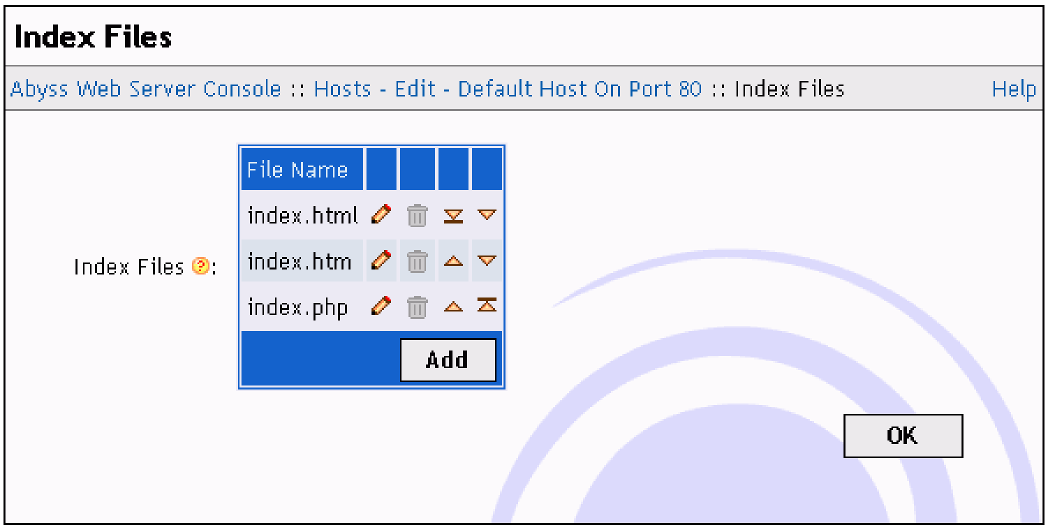
- **Disable Caching Negotiation for**: Caching negotiation with the browser (through the **`If-modified`** HTTP headers) is disabled for requests which path matches one of the virtual paths listed in this table.
- **Custom HTTP Headers**: Use this table to declare custom HTTP headers that will be included in the server responses. A custom header is added when the request's virtual path matches with the value or the pattern of its associated **Virtual Path** field. See the "Patterns Format" appendix for more information about patterns. The HTTP headers **`Server`**, **`Connection`**, **`Keep-Alive`**, **`Transfer-Encoding`**, and **`Content-Length`** cannot be customized.
- **Exclusively Serve On HTTPS**: If a request's virtual path matches with one of the paths or the patterns declared in that table, and if the request was received on a non-secure (HTTP) connection, the server generates a redirection to the same virtual path but using HTTPS to force it to be served on a secure connection. The **Exclusively Serve On HTTPS** table is available only when the current host's **Protocol** is set to **HTTP+HTTPS**.
- **Request Body Restrictions**: Restriction on the request body (sent by client for HTTP POST requests for example) are applied for requests which path matches one of the virtual paths listed in this table. Each of these virtual paths could be associated with a restriction on the total length of request body (controlled through the **Maximum Length** parameter.) Similarly, request bodies which are sent by the client using chunked encoding could be automatically dechunked and served as normal request bodies to scripts and Web applications which are not compatible with chunked encoding. Note that CGI and FastCGI compatible scripts and Web applications need dechunking as these specification do not support chunked encoding. The parameter **Dechunk** controls the support of that feature for a given virtual path and **Maximum Dechunk Length** defines the maximum size of the dechunk buffer. Since dechunking requires buffering in the memory, do not set this limit too high especially on busy servers.
- **File Expiration Times**: If the virtual path of a request matches with one of the paths or the patterns declared in this table and if it refers to a static file, the MIME type of the file is compared to the one in the matching rule. If it matches, an **`Expire`** HTTP header is injected in the reply. The expiration date is set using the value of the **Time Delta** field relatively to the **Time Base** which is either the file's last modification date or the current date (now.)
- **Do not serve file requests with Pathinfo in**: By default, if a virtual path such as **`/photos/2014/house.jpg`** refers to an actual static file or SSI page, a request for virtual path **`/photos/2014/house.jpg/extra/goes/here`** will be successful

and will serve the same content as the previous and shorter virtual path. **`/extra/goes/here`** is called the path information (or simply *Pathinfo*.)

If the virtual path of a request matches with one of the paths or the patterns declared in this table, if it refers to a static or SSI file, and if path information is present, the server replies with an error which status code is 404 (Not found) instead of serving the file.

Index Files

When a browser asks for a URL that does not contain a filename, the server checks for the existence of each index file in the mapped directory. If none is found and automatic directory indexing is enabled, a directory listing is generated and sent to the browser. Otherwise, an error is reported.


Index Files
Abyss Web Server Console :: Hosts - Edit - Default Host On Port 80 :: Index Files
Help
Index Files:
Add
OK

File Name
index.html
index.htm
index.php

Figure 5-4. Index Files dialog

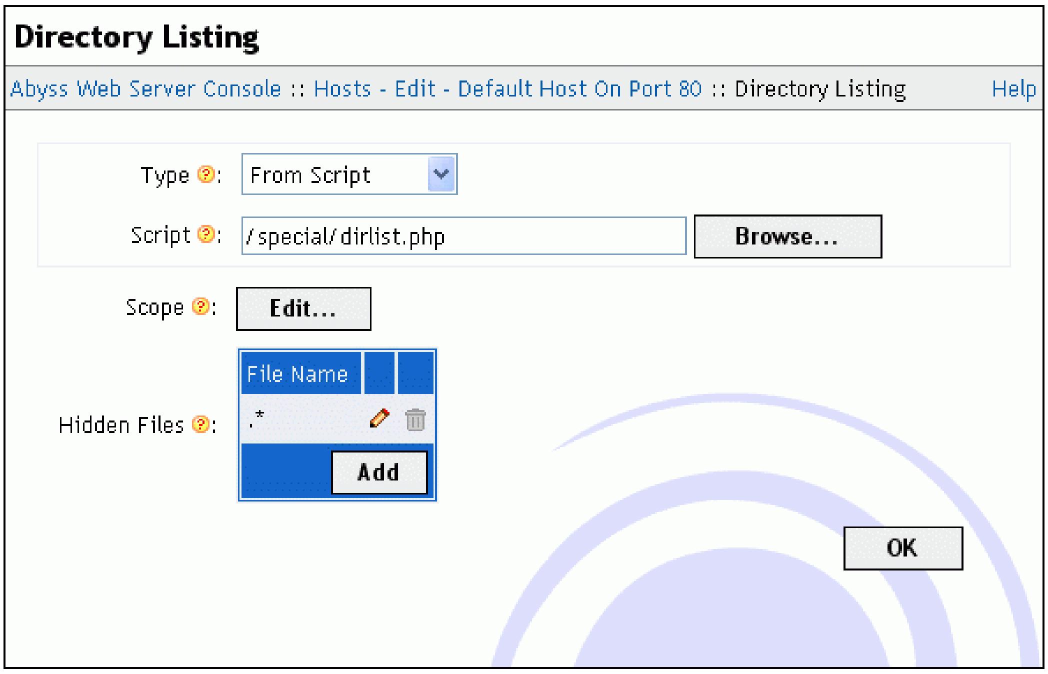
To edit, remove or add index filenames, use the **Index Files** table in the **Index Files** dialog.

Example: *index.htm and index.html as index files*

Assume that **`index.html`** *and* **`index.htm`** *are set as index filenames. If a browser asks for* **`http://<your host name>:<host port>/hello/`**, *the server checks if* **`<documents path>/hello/index.html`** *exists. If not, it checks if* **`<documents path>/hello/index.htm`** *exists. If so, it is sent to the browser. If not, a listing of the directory* **`<documents path>/hello/`** *is generated and sent to the browser if automatic directory indexing is enabled. If it is disabled, the server replies with a forbidden error message.*

Directory Listing

When a browser asks for a URL that does not contain a filename, and if the server does not find an index file in the mapped directory, it generates a directory listing.


Directory Listing

Abyss Web Server Console :: Hosts - Edit - Default Host On Port 80 :: Directory Listing   Help

Type: From Script

Script: /special/dirlist.php   Browse...

Scope: Edit...

Hidden Files:

File Name		
.*		

Add

OK

Figure 5-5. Directory Listing dialog

This dialog contains the following elements:

- **Type**: The type of listing that should be generated. It can be:
  - **Disabled**: Disables directory listing. In this case, the server generates error 403 instead of a listing.
  - **Standard Listing**: A listing based on a standard and fixed basic template is generated.
  - **From Template**: A listing is generated according to the configured custom template. See the "Custom Directory Listings" chapter for detailed information about creating custom templates.
  - **From Script**: A listing is generated by the script configured in the **Script** field. Refer to the "Custom Directory Listings" chapter for detailed information about creating directory listing scripts.
- **Scope**: Press **Edit...** to configure the virtual paths where directory listing is permitted. The displayed dialog has the following items:
  - **Order**: The order that the server follows to check if directory listing is permitted for a virtual path. If it is set to **Allow/Deny**, listing is denied by default and is allowed only if the virtual path is in the **Allow for** list and is not in the **Deny for** list. If it is set to **Deny/Allow**, listing is allowed by

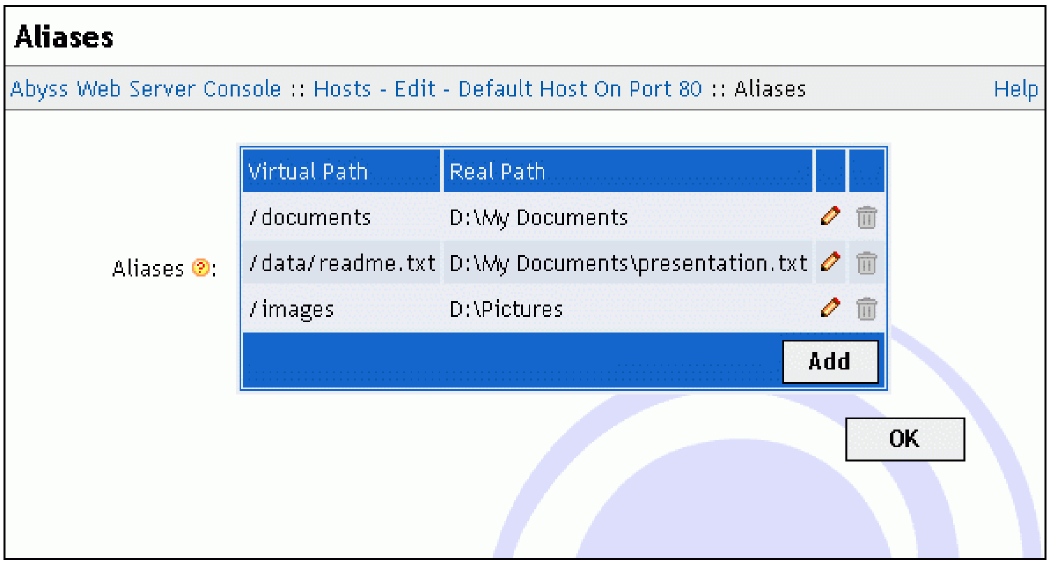
default and is denied only if the virtual path is in the **Deny for** list and is not in the **Allow for** list.

- **Allow for**: The list of virtual paths for which directory listing is allowed. The table can contain also path patterns. See the "Patterns Format" appendix for more information about patterns.
- **Deny for**: The list of virtual paths for which directory listing is denied. The table can contain also path patterns. See the "Patterns Format" appendix for more information about patterns.

- **Hidden Files**: The file names that are equal or that match with the listed file name patterns in this table are not included in the directory listings. See the "Patterns Format" appendix for more information about patterns. This table should only contain file names with no path references.

Aliases

Select **Aliases** in the host configuration menu to display the aliases table.



Aliases

Abyss Web Server Console :: Hosts - Edit - Default Host On Port 80 :: Aliases Help

Aliases:

Virtual Path	Real Path
/documents	D:\My Documents
/data/readme.txt	D:\My Documents\presentation.txt
/images	D:\Pictures

Figure 5-6. Aliases dialog

If a URL matches an alias' virtual path, the web server maps it to the alias' associated real path.

Use the displayed table to edit, remove or add aliases.

***Example:** Relative real path*

*Assume that there exists an alias which virtual path is **`/images`** and which real path is **`web/artwork`**. If a browser asks for **`http://<your host name>:<host port>/images/logo.jpg`**, the server maps the requested URL to the file*

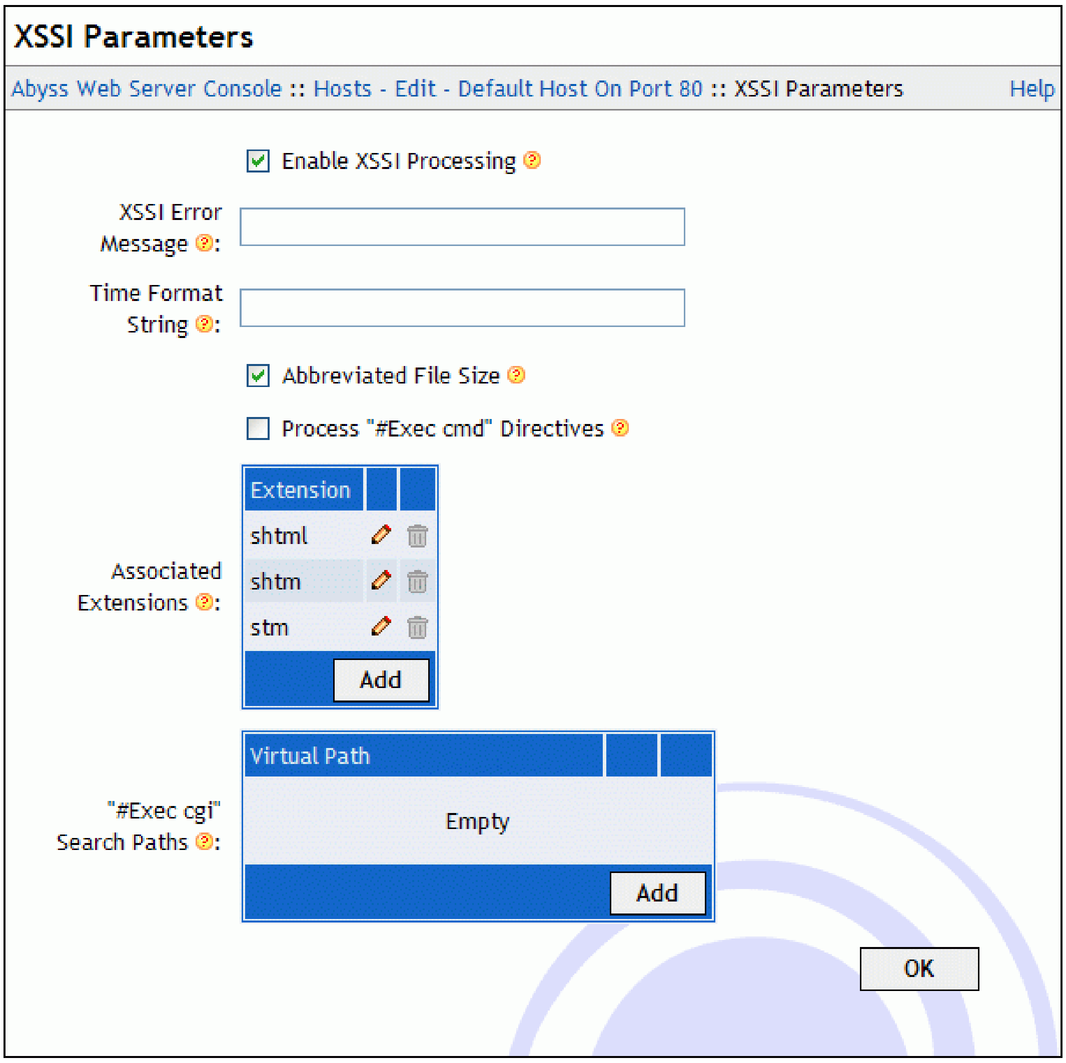
`<server root>/web/artwork/logo.jpg`. The `<server root>` is added because the real path was relative.

***Example:** Absolute real path*

Assume now that there exists an alias which virtual path is `/images` and which real path is `d:\web\artwork`. If a browser asks for `http://<your host name>:<host port>/images/logo.jpg`, the server maps the requested URL to the file `d:\web\artwork\logo.jpg`. The difference with the previous example is that the real path is absolute and not relative.

XSSI Parameters

To configure XSSI (eXtended Server Side Includes), select **XSSI Parameters** in the host configuration menu.


XSSI Parameters

Abyss Web Server Console :: Hosts - Edit - Default Host On Port 80 :: XSSI Parameters

[✓] Enable XSSI Processing

XSSI Error Message:

Time Format String:

[✓] Abbreviated File Size

[ ] Process "#Exec cmd" Directives

Associated Extensions:

Extension
shtml
shtm
stm

"#Exec cgi" Search Paths:

Virtual Path
Empty

Figure 5-7. XSSI Parameters dialog

This dialog includes the following fields:

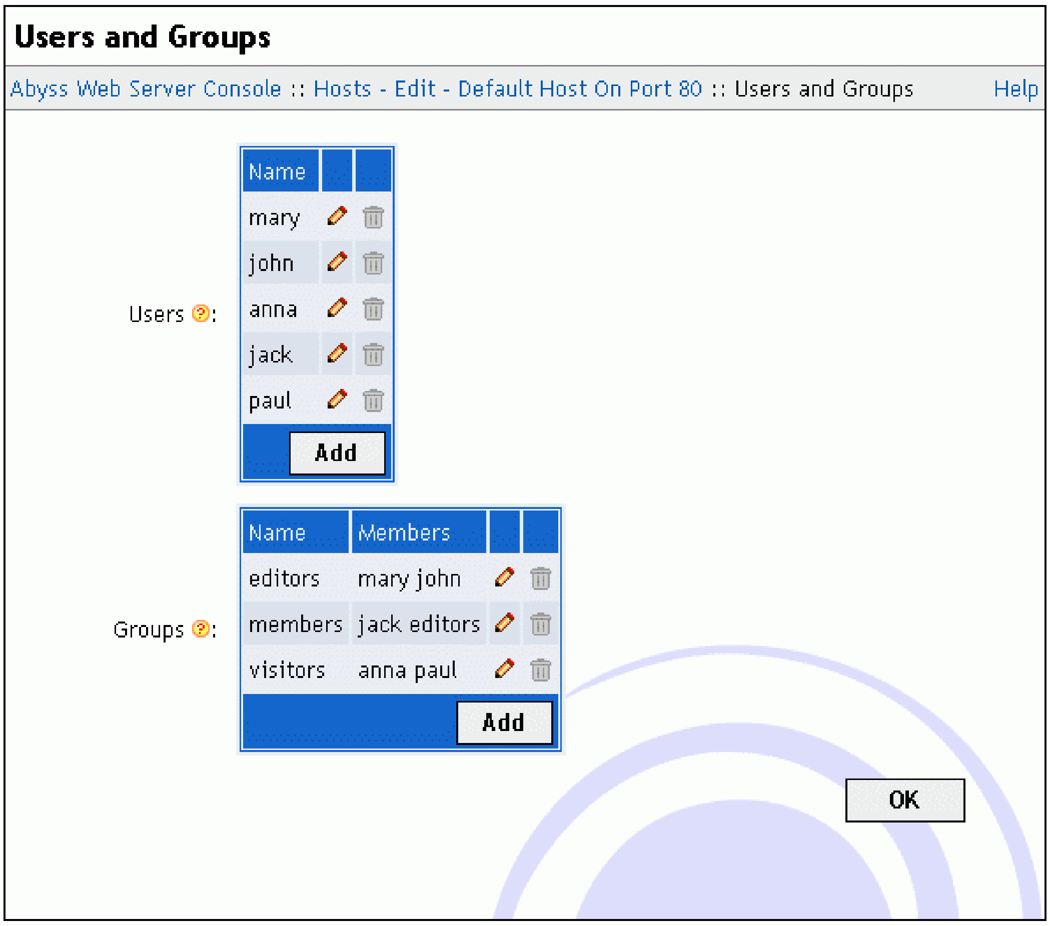
- **Enable XSSI Processing**: Enable/disable XSSI processing.

- **XSSI Error Message**: The default error message that the server inserts when an error is detected while processing an XSSI directive. If empty, an accurate error description with debugging information is generated.
- **Time Format String**: The default time format string that the server uses to display times while processing XSSI directives. If empty, the string **`%A, %d-%b-%Y %H:%M:%S %Z`** is used. For the complete reference of the time format string, read the description of **`<!-- #config timefmt="time_format" -->`** directive in "eXtended Server Side Includes" chapter.
- **Abbreviated File Size**: The default way to display file sizes. If it is checked, file sizes are displayed in KB or MB. Otherwise, they are displayed in bytes.
- **Process "#Exec cmd" Directives**: Enable/disable the execution of shell commands in XSSI.
- **Associated Extensions**: If a file name extension matches with one of these extensions or extensions patterns, it is processed as an XSSI file. Read the "Patterns Format" appendix for more information about patterns. By default, the server is configured to process XSSI directives in files which extensions are **`shtml`**, **`shtm`**, or **`stm`**.
- **"#Exec cgi" Search Paths**: If the argument of a **`#exec cgi`** directive is a relative path, the server will try to locate the file inside the directory containing the currently processed XSSI file. If it is not found there, it will search for it inside the virtual paths listed in **'#exec cgi' Search Paths**.

For more information about XSSI, refer to the "eXtended Server Side Includes" chapter.

Users and Groups

Select **Users and Groups** in the host configuration menu to display the users and groups tables.


Users and Groups

Abyss Web Server Console :: Hosts - Edit - Default Host On Port 80 :: Users and Groups   Help

Users:

Name		
mary	✎	🗑
john	✎	🗑
anna	✎	🗑
jack	✎	🗑
paul	✎	🗑

Groups:

Name	Members		
editors	mary john	✎	🗑
members	jack editors	✎	🗑
visitors	anna paul	✎	🗑

Figure 5-8. Users and Groups dialog

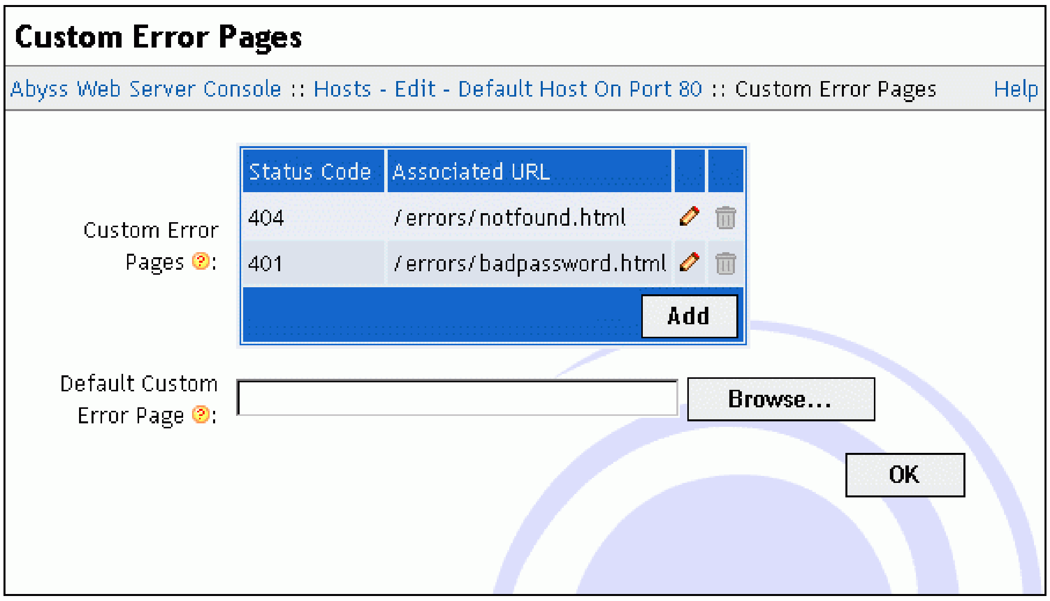
Use the displayed tables to edit, remove or add users and groups.

A user is defined by its name and its password. A group is defined by its name and its members which can be users and other groups.

Note: *The console hides automatically groups that can lead to circular references when editing a group.*

Custom Error Pages

With Abyss Web Server, you can override the standard error pages and replace them with yours. To do so, select **Custom Error Pages** in the host configuration menu.


Custom Error Pages

Abyss Web Server Console :: Hosts - Edit - Default Host On Port 80 :: Custom Error Pages   Help

Custom Error Pages:

Status Code	Associated URL		
404	/errors/notfound.html		
401	/errors/badpassword.html		

Default Custom Error Page:

Figure 5-9. Custom Error Pages dialog

This dialog includes the following elements:

- **Custom Error Pages**: This table contains the customized errors and their associated URLs.
- **Default Custom Error Page**: The URL used when an error which code is not listed in the **Custom Error Pages** table occurs. If empty, Abyss Web Server generates automatically a standard error page.

An error URL can be:

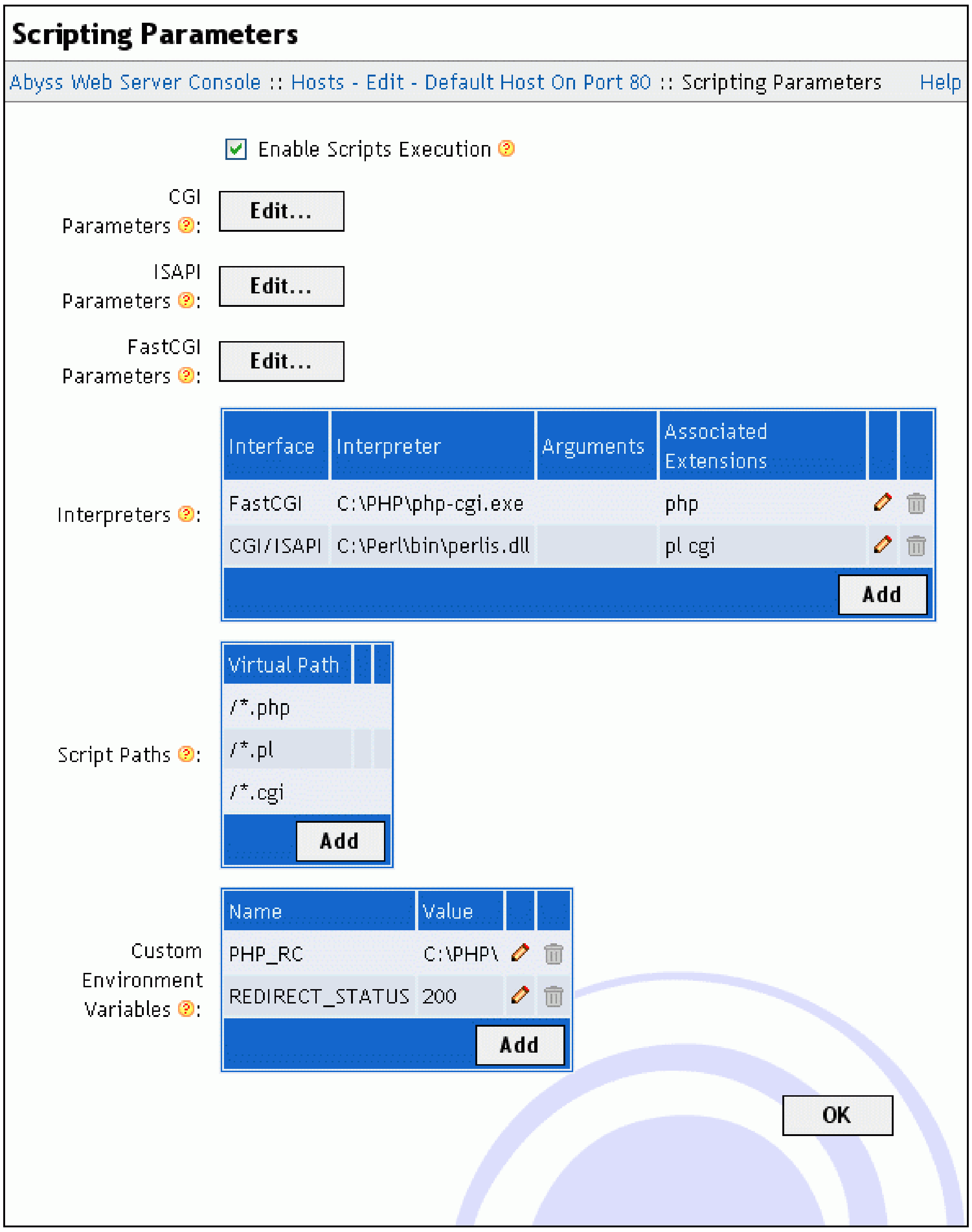
- Local: If it begins with a slash `/`, the URL is local to the web server.
- Global: If it begins with `http://` or `https://`, the URL is global and the web server informs the browser to redirect to that URL when an error occurs.

Note: *It is only relevant to set 4xx and 5xx error codes. Other error codes are handled internally in the web server and do not lead to displaying an error page.*

Note: *When using a local URL that is a CGI script or an XSSI page as a custom error page, the server operates an internal redirection and adds to the custom error page's environment variables all the faulty request environment variables prefixed with* ***`REDIRECT_`****. It adds also the special variables* ***`REDIRECT_STATUS`*** *and* ***`REDIRECT_STATUS_CODE`*** *which contain the status code of the faulty request. For more information, read "CGI environment variables" section in "CGI, FastCGI, and ISAPI" chapter.*

Scripting Parameters

To configure CGI, FastCGI, ISAPI, and scripts execution, select **Scripting Parameters** in the host configuration menu.



Scripting Parameters

Abyss Web Server Console :: Hosts - Edit - Default Host On Port 80 :: Scripting Parameters   Help

[✓] Enable Scripts Execution

CGI Parameters: Edit...

ISAPI Parameters: Edit...

FastCGI Parameters: Edit...

Interpreters:

Interface	Interpreter	Arguments	Associated Extensions
FastCGI	C:\PHP\php-cgi.exe		php
CGI/ISAPI	C:\Perl\bin\perlis.dll		pl cgi

Script Paths:

Virtual Path
/*.php
/*.pl
/*.cgi

Custom Environment Variables:

Name	Value
PHP_RC	C:\PHP\
REDIRECT_STATUS	200

Figure 5-10. Scripting Parameters dialog

This dialog includes the following fields:

- **Enable Scripts Execution**: Enable/disable CGI, FastCGI, ISAPI, and scripts execution.
- **CGI Parameters**: Press **Edit...** to access the CGI parameters dialog which contains the following elements:

    - **Error File**: The path of the file where CGI scripts write error messages. You can leave it empty if you do not want to trace CGI scripts' errors.
    - **I/O Timeout**: How long (in seconds) the server should wait for a CGI script or application to deliver content before aborting it.
    - **Resolve Interpreter from the Windows Registry**: Use the Windows Registry to automatically find the interpreter which can run a CGI script.

      This parameter should be used very carefully as it can make the server report Error 500 (Internal Server Error) for normal documents. For example, if a HTML file is in one of the CGI Paths (or matches with one of the CGI Paths patterns), and if this parameter is checked, the server asks the Windows Registry about the executable that is normally used to open HTML files (in a similar fashion to what does Windows Explorer to know what application to launch when you double-click on a document icon). The Windows Registry gives back your browser executable path and Abyss Web Server runs the HTML file as a CGI Script with this browser as its interpreter. But after launching it, Abyss Web Server understands that this executable is not a valid CGI Interpreter. So it aborts the executable and reports Error 500.
    - **Resolve Interpreter using the #! Line**: Read the first line of the CGI script. If it begins with `#!`, the rest of the line is considered as the path to the script's interpreter.

- **ISAPI Parameters**: Press **Edit...** to access the ISAPI parameters dialog which contains the following elements:
    - **Error File**: The path of the file where ISAPI extensions write error messages and where the server logs ISAPI activity when **Debugging Level** is set to a value other than **None**. You can leave it empty if you do not want to trace ISAPI errors.
    - **Debugging Level**: The type of information Abyss Web Server should log when an ISAPI is invoked. It should be set the **None** unless you are developing or debugging an ISAPI extension.
    - **ISAPI Filename Extensions**: This table contains the file name extensions that helps the server recognize if a file is an ISAPI or not. It is filled with `dll` by default. This table is used for example to know if a declared interpreter is an ISAPI extension or not.

- **FastCGI Parameters**: Press **Edit...** to access the FastCGI parameters dialog which contains the following elements:
    - **Error File**: The path of the file where FastCGI executables and interpreters write error messages and where the server logs their activity when **Debugging Level** is set to a value other than **None**. You can leave it empty if you do not want to trace FastCGI errors.

- **Debugging Level**: The type of information Abyss Web Server should log when a FastCGI executable or interpreter is running.
- **I/O Timeout**: How long (in seconds) the server should wait for a FastCGI process to deliver content before aborting the connection with it and reporting an error.
- **FastCGI Processes Timeout**: How long (in seconds) the server should let an unused FastCGI process wait before aborting it.

The dialog contains also the following tables:

- **Interpreters**: The server uses this table to know which interpreter to use to execute a script. The choice is based on the script's file name extension: an interpreter runs a script if the extension of that script matches with one of the associated extensions or extensions patterns of the interpreter.

  Each interpreter is defined by its:

  - **Interface**: It has to be set to **FastCGI (Local - Pipes)** or **FastCGI (Local - TCP/IP Sockets)** if the interpreter supports FastCGI and its executable is available on the computer where the Web server is installed. If you have a standalone or remote FastCGI server, set it to **FastCGI (Remote - Pipes)** or **FastCGI (Remote - TCP/IP Sockets)** depending on the type of network protocol it supports. Otherwise, **Interface** should be set to **CGI/ISAPI**. Note that for most FastCGI compliant interpreters, you can either choose **FastCGI (Local - Pipes)** or **FastCGI (Local - TCP/IP Sockets)** with no noticeable difference as both modes provide comparable performances. But some FastCGI interpreters such as PHP 5.1.3 and later only support a single mode of operation. So if an interpreter fails to work in a given mode, select the other and retry. You can also refer to Aprelium's Web site to know which mode is supported by that interpreter. Note also that both **FastCGI (Local - Pipes)** and **FastCGI (Remote - Pipes)** are not available on Windows 95, 98, and ME.
  - **Interpreter**: The path of the FastCGI/CGI executable or the ISAPI extension (not available if **Interface** is set to **FastCGI (Remote - Pipes)** or **FastCGI (Remote - TCP/IP Sockets)**).
  - **Arguments**: The additional arguments that are used to run a FastCGI/CGI executable (not available if **Interface** is set to **FastCGI (Remote - Pipes)** or **FastCGI (Remote - TCP/IP Sockets)**). These arguments are ignored if the interpreter is an ISAPI extension. Any occurrence of `%1` or `$1` in the arguments is replaced by the script file name when running the FastCGI/CGI interpreter.
  - **Advanced Parameters**: If **Interface** is set to **FastCGI (Local - Pipes)** or **FastCGI (Local - TCP/IP Sockets)**, press **Edit...** to access the advanced parameters dialog which contains the following elements:

  - **Minimum Processes**: The minimum number of processes of the FastCGI interpreter that the Web server will maintain available and running. If this field is empty, no processes are maintained running and new ones are created on demand.
  - **Maximum Processes**: The maximum number of processes of the FastCGI interpreter that the Web server will allow to be running at the same time. If this field is empty, no maximum limit is imposed.
  - **Conservative Behavior Process Count Threshold**: When the number of processes of the FastCGI interpreter reaches this parameter's value, new processes are not immediately created on demand: the Web server will wait for some amount of time for an already running process to be ready. If no process is ready after that wait, a new one is created. If empty, no such soft limit exists.
  - **Conservative Behavior Polling Interval**: The amount of time in milliseconds the Web server waits for before creating a new process for the FastCGI interpreter if the **Conservative Behavior Process Count Threshold** is reached and no process becomes available. If empty, it defaults to 1000 milliseconds (1 second.)
  - **Request Count Before Process Recycling**: When a given process of the current FastCGI interpreter has processed as many requests as indicated by this parameter, it is gracefully shut down and restarted. An **Unlimited** value means that no recycling will ever occur.
  - **Maximum Process Age Before Recycling**: When a given process of the current FastCGI interpreter has been running for the duration indicated by this parameter, it is gracefully shut down and restarted. No recycling will ever occur if this parameter is set to **Unlimited**.
  - **Recycle when the following file changes**: If this field is not empty, any change to the specified file will trigger the the restart of all the processes associated with the current FastCGI interpreter.

- **Pipe Path**: The full path of the pipe that will be used to communicate with the interpreter if **Interface** is set to **FastCGI (Remote - Pipes)**. If the FastCGI server is running locally, its pipe path should be of the form **`\\.\pipe\name`** where **`name`** is the name of the pipe. If the FastCGI server is available on another computer on the LAN, its pipe path should be of the form **`\\computer\pipe\name`** where **`computer`** is the name of the computer hosting the FastCGI server.
- **Remote Server IP Address**: The IP address of the computer that is hosting the interpreter's FastCGI server (available only if **Interface** is set to **FastCGI (Remote - TCP/IP Sockets)**).
- **Port**: The number of the port the interpreter's FastCGI server is listening on (available only if **Interface** is set to **FastCGI (Remote - TCP/IP Sockets)**).

  - **Check for file existence before execution**: If enabled, the server will report an error when the script file requested is not physically available on the disk. If this option is disabled, the server will launch the interpreter associated with the requested script without even if its file does not exist on the disk. It is then up to the interpreter to report an error or to act accordingly.
  - **Type**: Some interpreters are not fully conforming to the FastCGI/CGI or the ISAPI specifications. By setting this parameter to the correct value, the server activates a special *workaround* mode to support them. PHP interpreters (both FastCGI/CGI and ISAPI version) and Shorthand ISAPI should have their **Type** set to **PHP Style**. ActiveState ActivePerl ISAPI should have its **Type** set to **ActivePerl ISAPI**. For any other interpreter, set **Type** to **Standard**.
  - **Associated Extensions**: The list of file name extensions that are to be handled by the current interpreter. You can also define an extension pattern. Refer to the "Patterns Format" appendix for more information about patterns. To associate more than a single extension with an interpreter, press **Add...** for every extension to declare it.

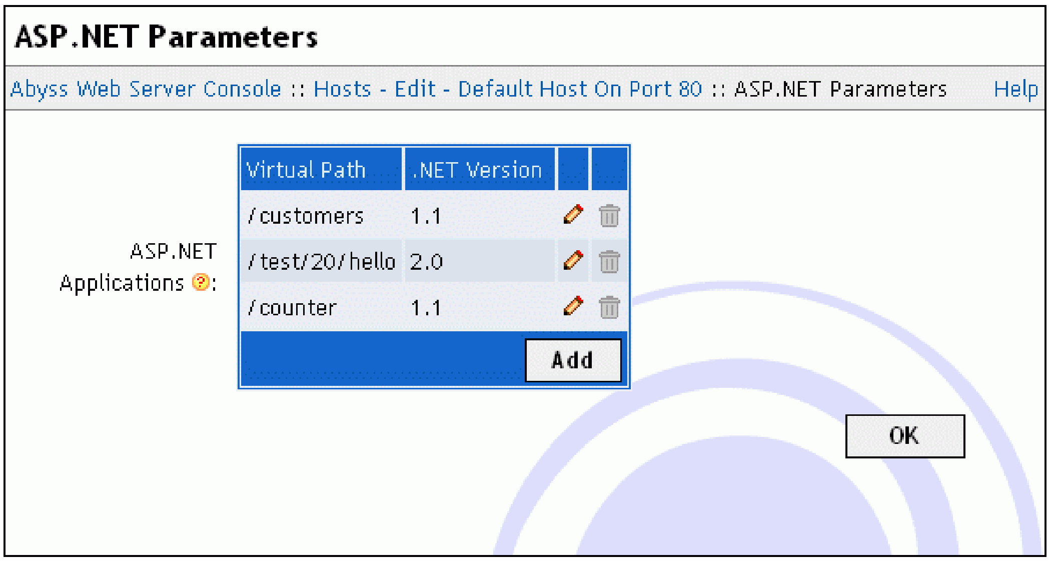
  When adding or editing an interpreter, letting **Use the associated extensions to automatically update the Script Paths** checked makes Abyss Web Server automatically add/remove the pattern **`/*.ext`** in the **Script Paths** table for every added/removed extension **`ext`** in the **Associated Extensions** list.

- **Script Paths**: Only files that are in one of the **Script Paths** or their subpaths, or that match with one of the **Script Paths** patterns can be processed as scripts. These paths are virtual paths. See the "Patterns Format" appendix for more information about patterns. An executable located in each path or its subpaths is considered as CGI or FastCGI applications according to the setting of **Interface type of executables in this path** of the corresponding script path. If **X-SendFile Support** is enabled, scripts or executables can emit an **`X-Sendfile`** CGI header which value will be used to serve a static file relative to the **Files Root Path**. For more details about that feature, refer to the "X-Sendfile Support" section. It is also possible to enable or disable bufering the output of scripts or executables in that path by setting the value of **Output Buffer Size**.
- **Custom Environment Variables**: This table contains the variables you want to add to the execution environment of the FastCGI/CGI interpreters and scripts. Each variable is defined by its name and its value.

For more information about scripts, see "CGI, FastCGI, and ISAPI" chapter.

ASP.NET Parameters

Select **ASP.NET Parameters** in the host configuration menu to display the ASP.NET applications table.


ASP.NET Parameters

Abyss Web Server Console :: Hosts - Edit - Default Host On Port 80 :: ASP.NET Parameters Help

ASP.NET Applications:

Virtual Path	.NET Version		
/customers	1.1		
/test/20/hello	2.0		
/counter	1.1		

Figure 5-11. ASP.NET Parameters dialog

Abyss Web Server features genuine support for ASP.NET applications. If you do not see the **ASP.NET Parameters** icon in the host menu, it is probably because you have not installed ASP.NET support in Abyss Web Server. To remedy to that, launch the installer package again to upgrade your current setup of Abyss Web Server and ensure that the **ASP.NET Support** component is selected.

If you do not have a version of Microsoft .NET Framework 1.1 (or higher) correctly installed on your system, Abyss Web Server will report the problem and provide you with a link to install the missing system component.

Unlike other scripting languages and platforms, ASP.NET has the concept of *application* which is a set of files related to the same ASP.NET program. ASP.NET must know where each program is to be able to locate the configuration files of the application and to process it correctly.

If ASP.NET support is correctly set up and a recognized Microsoft .NET Framework version is installed, the **ASP.NET Parameters** dialog shows a table to edit, remove or add ASP.NET applications.

Each ASP.NET application is defined by the following parameters:

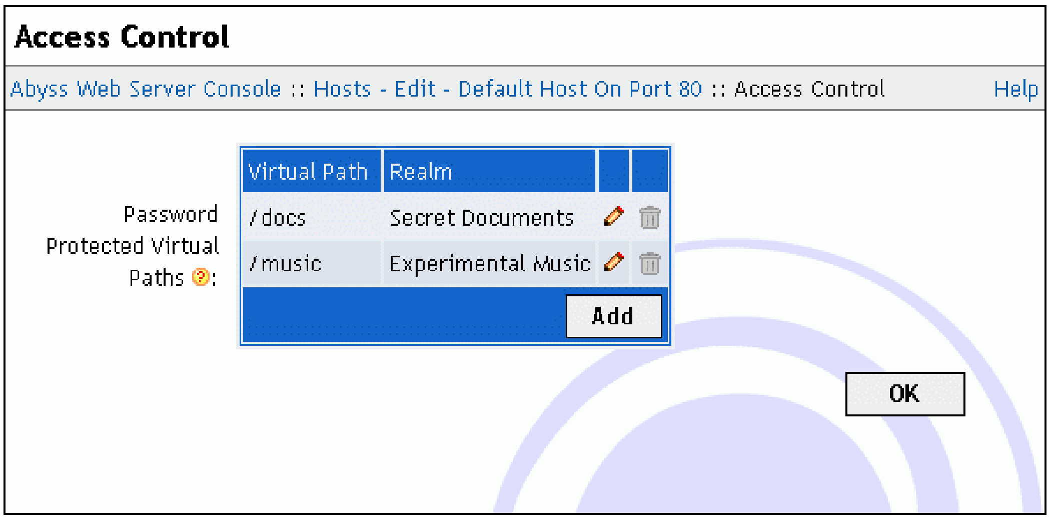
- **Virtual Path**: The virtual path containing all the ASP.NET application files and subdirectories.
- **.NET Version**: The version of .NET Framework that will be used to run the ASP.NET application.

- **Preload**: If checked, the ASP.NET application is loaded during server startup. Otherwise, it is loaded on demand when the first request for it is received by the Web server.
- **Request Count Before Process Recycling**: When the ASP.NET connector process (powering the particular **.NET Version** specified for the current ASP.NET application) has processed as many requests as indicated by this parameter, it is gracefully shut down and restarted. No recycling will ever occur if this parameter is set to **Unlimited**.
- **Maximum Process Age Before Recycling**: When the ASP.NET connector process (powering the particular **.NET Version** specified for the current ASP.NET application) has been running for a period of time indicated by this parameter, it is gracefully shut down and restarted. A value of **Unlimited** for this parameter means that no recycling will ever occur.
- **Preload Type**: If set to **Standard**, the ASP.NET application is preloaded by starting an ASP.NET connector process. To enable a full preload and possibly the warm-up of the ASP.NET application (by forcing an immediate compilation of its code behind files), **Preload Type** should be set to **Internal HTTP HEAD request** which simulates a first request at the server startup to force a full preload of the application and its components.
- **Preload Subpath**: The value of that field is appended to the value of **Virtual Path** to build a URL that will be requested internally by the server when preloading the ASP.NET application. Most old ASP.NET applications require that field set to **`default.aspx`**. Modern ASP.NET applications (with routes, MVC or AJAX) can be started with no value for that field.
- **Isolated**: By default, all ASP.NET applications that are run with **Isolated** not checked share the same .NET framework application domain.. When **Isolated** is checked, the ASP.NET application is loaded in its own .NET framework application domain.
- **Handling**: This parameter defines the parts of the virtual path that should be served by the ASP.NET application. When set to **All the subdirectory**, all the virtual paths (even those which do not relate to actual files) will be forwarded to the ASP.NET application. This is the default setting and it is recommended for modern ASP.NET applications especially those taking advantage of routes, MVC or AJAX. If you want to delegate the processing of some specific files to the ASP.NET application and have the rest of them handled by the Web server for maximum performance, **Handling** should be set to **Specific File Extensions**. In such a case, the ASP.NET application will only process files which exist and which file name extension is listed in the **Associated Extensions** table.
- **Associated Extensions**: Files inside the ASP.NET application virtual path which extensions are listed in this table are processed by the ASP.NET engine. Note that **Associated Extensions** is always filled with the full list of file name extensions that can be handled by the .NET version you have

selected when defining the ASP.NET application. So it is rarely necessary to update this table.

Access Control

To restrict and manage the access to paths in your web site, select **Access Control** in the host configuration.



Access Control

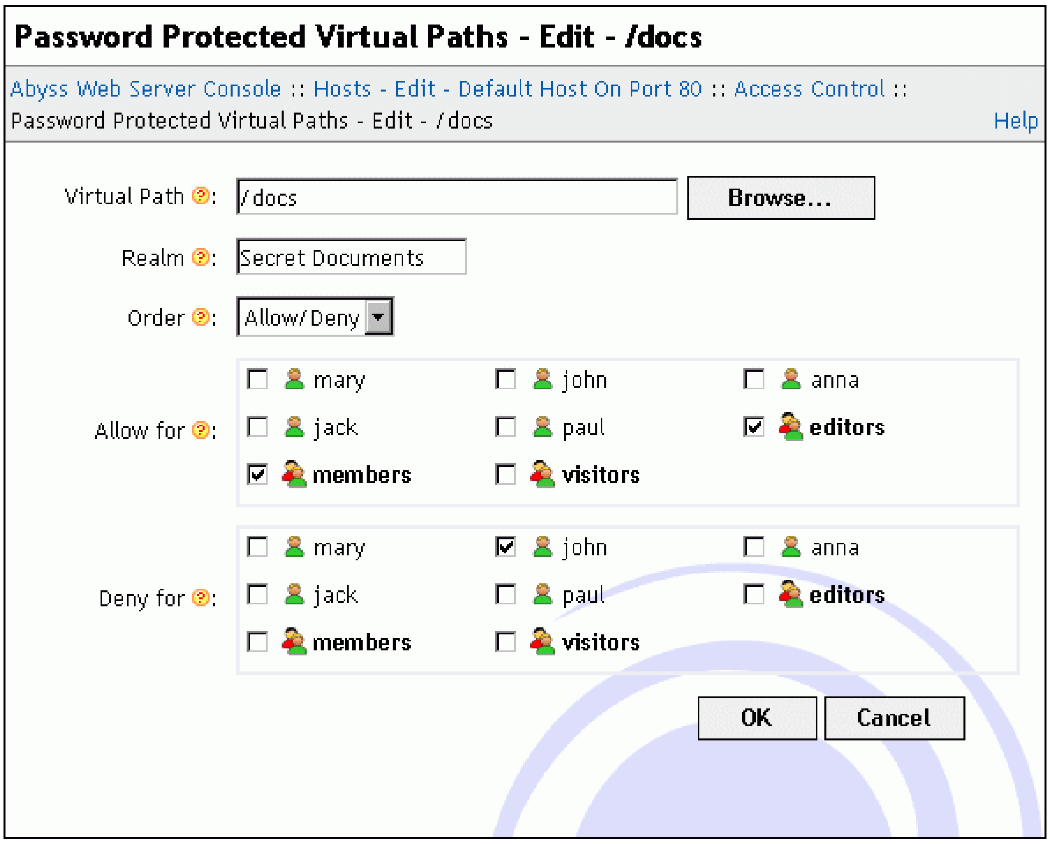
Abyss Web Server Console :: Hosts - Edit - Default Host On Port 80 :: Access Control   Help

Password Protected Virtual Paths:

Virtual Path	Realm		
/docs	Secret Documents		
/music	Experimental Music		

Figure 5-12. Access Control dialog

To edit, remove or add path access rules, use the displayed table.



Password Protected Virtual Paths - Edit - /docs

Abyss Web Server Console :: Hosts - Edit - Default Host On Port 80 :: Access Control :: Password Protected Virtual Paths - Edit - /docs   Help

Virtual Path: /docs [Browse...]

Realm: Secret Documents

Order: Allow/Deny

Allow for: [ ] mary [ ] john [ ] anna [ ] jack [ ] paul [✓] editors [✓] members [ ] visitors

Deny for: [ ] mary [✓] john [ ] anna [ ] jack [ ] paul [ ] editors [ ] members [ ] visitors

[OK] [Cancel]

Figure 5-13. Path access dialog

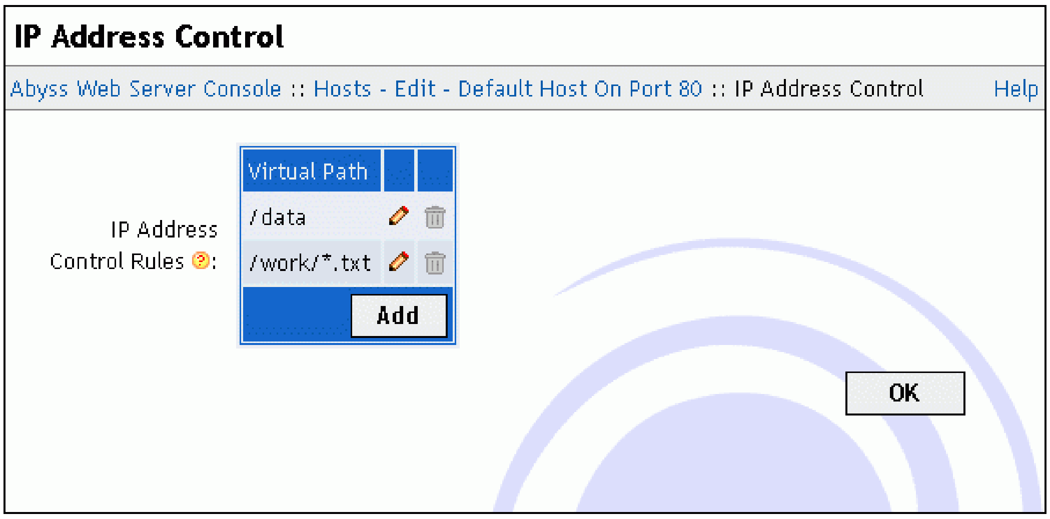
The path access edition dialog contains the following fields:

- **Virtual Path**: The virtual path which contents' access is to be restricted. It must always begin with a slash **/**. This field accepts also path patterns. In that case, the access rule applies to any virtual path that matches with the specified pattern. Refer to the "Patterns Format" appendix for more information about patterns.
- **Realm**: A short description of the path's contents. It is prompted to users by the browser when it asks them for credentials.
- **Order**: The order that the server follows to check if access is granted to a user. If it is set to **Allow/Deny**, access is denied by default and is allowed only if the user is in the **Allow for** list and is not in the **Deny for** list. If it is set to **Deny/Allow**, access is allowed by default and is denied only if the user is in the **Deny for** list and is not in the **Allow for** list.
- **Allow for**: The list of users and groups for whom access is allowed.
- **Deny for**: The list of users and groups for whom access is denied.

*Note: If **Deny for** list is empty and the order is **Deny/Allow**, access is granted to all declared users and groups.*

IP Address Control

To restrict and manage the access to paths in your web site, select **IP Address Control** in the host configuration.



IP Address Control

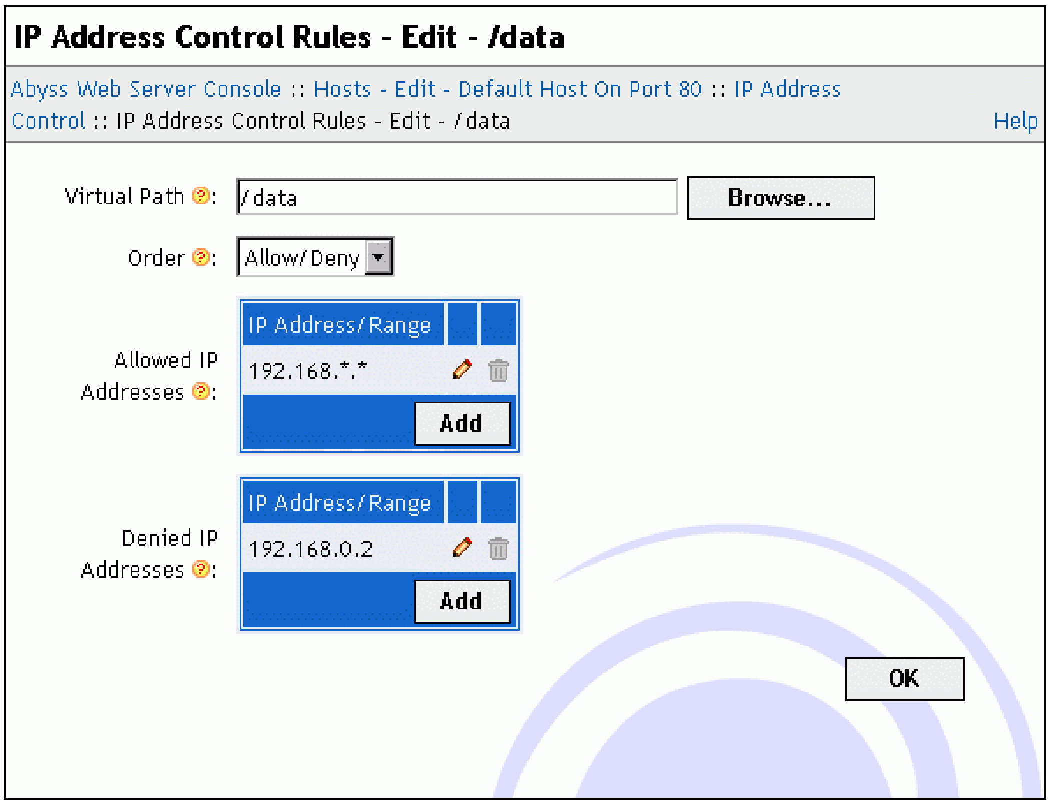
Abyss Web Server Console :: Hosts - Edit - Default Host On Port 80 :: IP Address Control Help

IP Address Control Rules:

Virtual Path		
/data		
/work/*.txt		

Figure 5-14. IP Address Control dialog

To edit, remove or add path IP Address Control rules, use the displayed table.



IP Address Control Rules - Edit - /data

Abyss Web Server Console :: Hosts - Edit - Default Host On Port 80 :: IP Address Control :: IP Address Control Rules - Edit - /data Help

Virtual Path: /data

Order: Allow/Deny

Allowed IP Addresses:

IP Address/Range		
192.168.*.*		

Denied IP Addresses:

IP Address/Range		
192.168.0.2		

Figure 5-15. IP Address Control Rule dialog

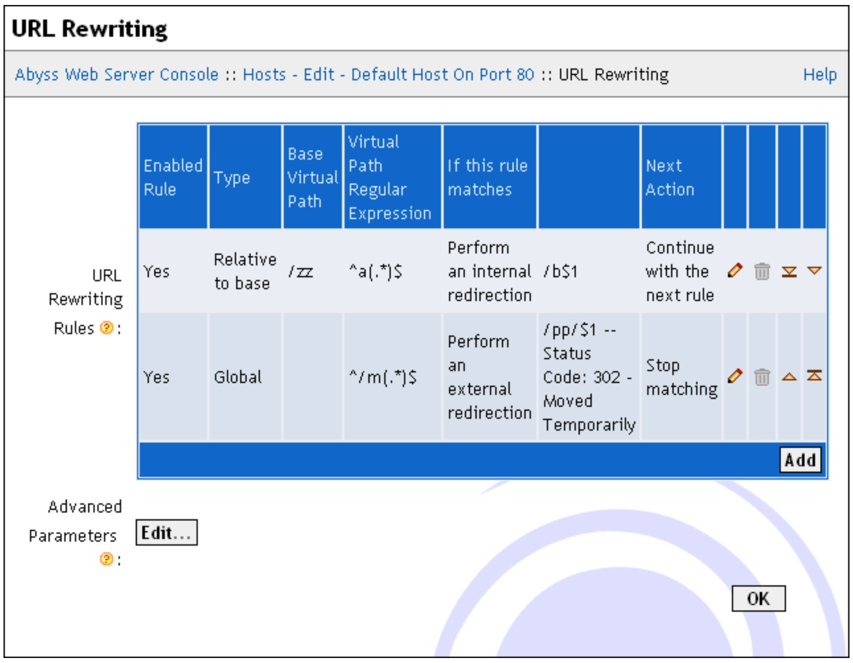
The IP address control rule edition dialog contains the following fields:

- **Virtual Path**: The virtual path which contents' access is to be restricted. It must always begin with a slash `/`. This field accepts also path patterns. In that case, the access rule applies to any virtual path that matches with the specified pattern. Refer to the "Patterns Format" appendix for more information about patterns.
- **Order**: The order that the server follows to check if access is granted to a client according to its IP address. If it is set to **Allow/Deny**, access is denied by default and is allowed only if the client IP address is in the **Allowed IP Addresses** list and is not in the **Denied IP Addresses** list. If it is set to **Deny/Allow**, access is allowed by default and is denied only if the client IP address is in the **Denied IP Addresses** list and is not in the **Allowed IP Addresses** list.
- **Allowed IP Addresses**: The list of IP addresses or IP address ranges for which access is allowed. Refer to the "IP Addresses and Ranges Format" appendix for more information about the IP addresses and ranges.
- **Denied IP Addresses**: The list of IP addresses or IP address ranges for which access is denied. Refer to the "IP Addresses and Ranges Format" appendix for more information about the IP addresses and ranges.

***Note:** If **Denied IP Addresses** list is empty and the order is **Deny/Allow**, access is granted to any client.*

URL Rewriting

To automatically redirect certain requests or rewrite their URL, select **URL Rewriting** in the host configuration menu.



URL Rewriting

Abyss Web Server Console :: Hosts - Edit - Default Host On Port 80 :: URL Rewriting   Help

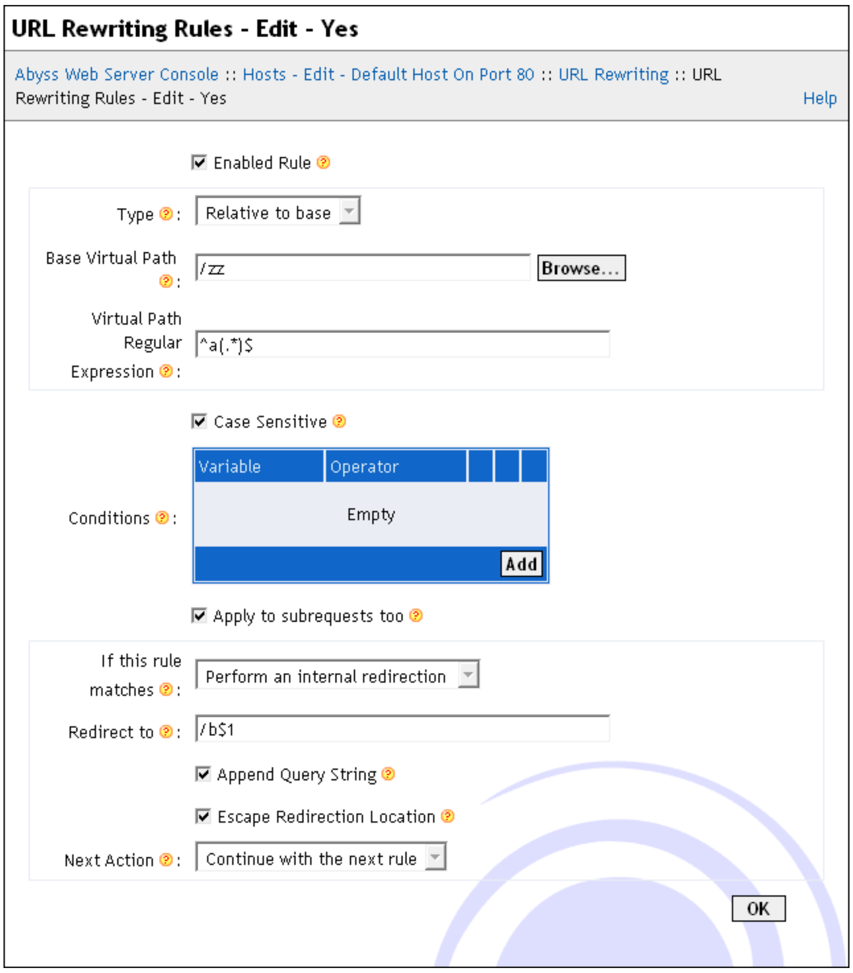
URL Rewriting Rules:

Enabled Rule	Type	Base Virtual Path	Virtual Path Regular Expression	If this rule matches		Next Action				
Yes	Relative to base	/zz	^a(.*)$	Perform an internal redirection	/b$1	Continue with the next rule				
Yes	Global		^/m(.*)$	Perform an external redirection	/pp/$1 -- Status Code: 302 - Moved Temporarily	Stop matching				

Figure 5-16. URL Rewriting dialog

This dialog includes the following elements:

- **URL Rewriting Rules**: Use this table to edit, remove or add URL rewriting rules. The URL rewriting rule definition dialog contains the following items:


URL Rewriting Rules - Edit - Yes

Abyss Web Server Console :: Hosts - Edit - Default Host On Port 80 :: URL Rewriting :: URL Rewriting Rules - Edit - Yes

Help

- [✓] Enabled Rule
- Type: Relative to base
- Base Virtual Path: /zz
- Virtual Path Regular Expression: ^a(.*)$
- [✓] Case Sensitive
- Conditions:

Variable	Operator
Empty	

- [✓] Apply to subrequests too
- If this rule matches: Perform an internal redirection
- Redirect to: /b$1
- [✓] Append Query String
- [✓] Escape Redirection Location
- Next Action: Continue with the next rule

Figure 5-17. URL Rewriting Rule dialog

- **Enabled Rule**: Enable/disable the current URL rewriting rule.
- **Type**: The rule can be **Global** or **Relative** (the paths will be defined relatively to a base path defined in the **Base Virtual Path**.)
- **Virtual Path Regular Expression**: If **Type** is set to **Global**, the current URL rewriting rule will be examined for virtual paths which match with this regular expression. But if **Type** is set to **Relative**, the URL rewriting rule will only be applied for virtual paths which head equals **Base Virtual Path** and which tail matches with this regular expression. For more information about regular expressions syntax, see the "Regular Expressions" appendix.
- **Case Sensitive**: The above regular expression is matched case sensitively if this option is checked. Otherwise, no difference between lower and upper cases is made while matching.
- **Conditions**: If all the conditions in this table are verified (are true), the URL rewriting rule will apply. Otherwise, it is skipped and ignored. Each condition consists in a test of the value of a first operand with an operator and possibly a second operand. The first operand can be a single CGI variable or a string which contains normal text mixed with references to one or many CGI variables using the syntax **`${VARIABLE}`**. Three kinds

of operators are available: string operators (regular expression matching, and lexicographical ordering), numerical operators (numerical ordering), and file operators (is a file, is a directory, is not a file or is empty).

- **Apply to subrequests too**: If checked, the rule is tested on subrequests too. Subrequests are requests that are internally generated by the server and not directly invoked by a client. For instance, XSSI include directives generate subrequests.
- **If the rule matches**: The action to perform when the virtual path matches with the regular expression and when all the conditions are verified:
  - **Perform an internal redirection**: The server will continue the processing of the request as if the requested virtual path was the one that is specified in **Redirect to**. That virtual path could contain backreferences to the groups of the regular expression specified in **Virtual Path Regular Expression**. Occurrences of **`$1`** are substituted with the backreference of the first group (if any), occurrences of **`$2`** are substituted with the backreference of the second group (if any), and so on. Occurrences of **`%1`**, **`%2`**, ..., **`%9`** are substituted with the backreferences of the groups (if any) of the last evaluated condition. For more information about backreferences, refer to the "Regular Expressions" appendix. Occurrences of **`%{VARIABLE}`** are substituted with the value of the CGI variable **`VARIABLE`** (see "CGI environment variables" in the "CGI, FastCGI, and ISAPI" chapter.) You also decide with **Next Action** if URL rewriting has to be stopped, restarted, or continued after the application of the current URL rewriting rule.
  - **Perform an external redirection**: The server will redirect the client to the URL or the virtual path set in **Redirect to**. That URL or virtual path could contain backreferences to the groups of the regular expression specified in **Virtual Path Regular Expression**. For more information about backreferences, refer to the "Regular Expressions" appendix. By default, the external redirection results in a HTTP response with a 302 status code but you can choose a more appropriate one by setting **Status Code** to another value between 300 and 399.
  - **Report an error to the client**: The server will report an error to the client. The **Status Code** of that error must be in the 400-999 range.
  - **Abort connection**: The server will abruptly abort the connection with the client without further processing. This action is not recommended as it breaks the HTTP protocol. It should only be used in a few special situations inside a set of rules aimed at banning bandwidth consuming bots or automated clients.

When a **Redirect To** field is available, the checkbox **Append Query String** is used to control the automatic addition of the original query string to the final redirected URL or virtual path. The original query string is the substring after the **`?`** character in the original request URL. If **Redirect To** already contains a query string and if the original URL's query string is not empty, both are concatenated in the final URL or virtual path that the request will be redirected to.

The **Escape Redirection Location** option should be unchecked only when the **Redirect To** field contains a location which is already URL escaped.

Note that if **Type** is set to **Relative** and the virtual path resulting from the processing of **Redirect to** is relative (which means that it does not begin with `/`), the value of **Base Virtual Path** is prepended to it.

  - **Advanced Parameters**: Press **Edit...** to access the advanced parameters dialog of the current URL rewriting rule which contains the following elements:
    - **Debugging Label**: An optional label which will be included with any reference of the current URL rewriting rule when logged inside the URL rewriting log file. Labels can be useful to locate a certain rule or group of rules inside the log file.

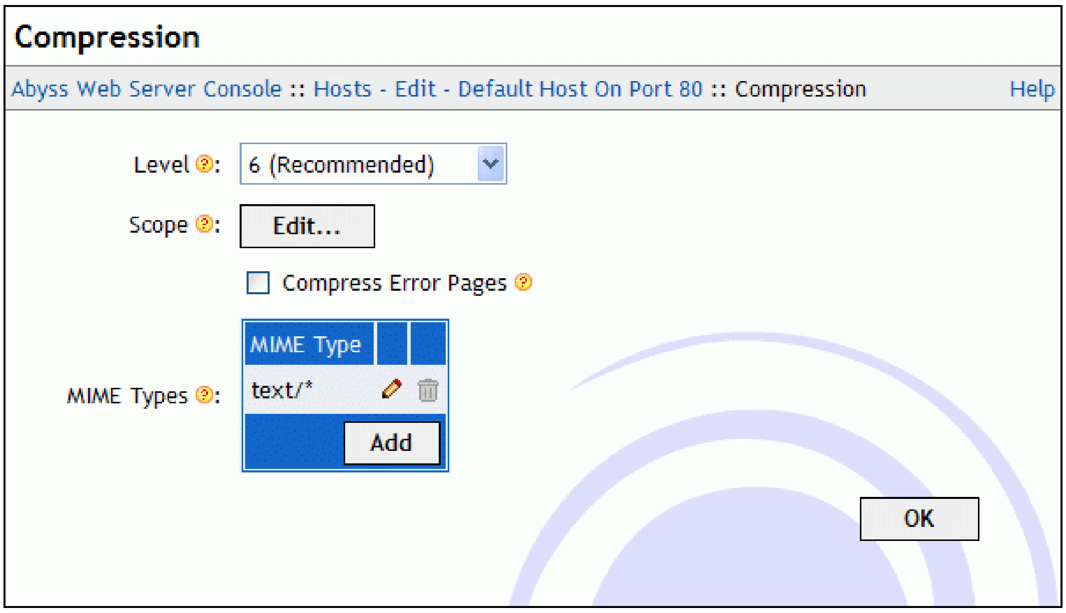
- **Advanced Parameters**: Press **Edit...** to access the advanced parameters dialog which contains the following elements:
  - **Log File**: The path of the file where the Abyss Web Server will log extensive debugging information about the processing of each request by the URL rewriting engine. The size of that file can become very huge very quickly as each request results in several dozen lines of logged debugging information. For that reason, it is recommended to turn it off on production servers.
  - **Log variables**: If checked, the CGI variables of each request are logged along with the URL rewriting processing details.

Compression

Compression is performed on any content (be it static or dynamically generated) provided that all the following conditions are satisfied:

- The browser requesting the content claims support for HTTP decompression by including an **`Accept-Encoding`** header in the HTTP request;
- The virtual path of the content belongs to the compression scope;
- The MIME type of the content matches with one of the MIME types listed in the MIME Types table.

If one or more of the above conditions is not met, the current content is sent back to the browser uncompressed.


Compression

Abyss Web Server Console :: Hosts - Edit - Default Host On Port 80 :: Compression

Level: 6 (Recommended)

Scope: Edit...

☐ Compress Error Pages

MIME Types:

MIME Type
text/*

Figure 5-18. Compression dialog

The **Compression** dialog contains the following elements:

- **Level**: The compression level. It ranges from 1 (Minimum compression - Faster) to 9 (Maximum compression - Slower). If it is set to 0, compression is disabled.
- **Scope**: Press **Edit...** to configure the virtual paths where compression is considered. The displayed dialog has the following items:
  - **Order**: The order that the server follows to check if compression is considered for a virtual path. If it is set to **Allow/Deny**, compression is denied by default and is allowed only if the virtual path is in the **Allow for** list and is not in the **Deny for** list. If it is set to **Deny/Allow**, compression is allowed by default and is denied only if the virtual path is in the **Deny for** list and is not in the **Allow for** list.
  - **Allow for**: The list of virtual paths for which compression is allowed. The table can contain also path patterns. See the "Patterns Format" appendix for more information about patterns.
  - **Deny for**: The list of virtual paths for which compression is denied. The table can contain also path patterns. See the "Patterns Format" appendix for more information about patterns.
- **Compress Error Pages**: Enable/Disable compression of responses which status codes correspond to an error.
- **MIME Types**: Compression is considered for contents which MIME types are equal or match with the listed MIME type patterns in this table. See the "Patterns Format" appendix for more information about patterns.

Bandwidth Limits

With Abyss Web Server, you can have fine control on the bandwidth every host uses. You can even control the bandwidth allowed for a particular file or script, or for the contents of a directory.

The **Bandwidth Limits** dialog has the following fields:

- **Enable Bandwidth Throttling**: Enable/Disable bandwidth throttling for the current host.
- **Limits**: This table contains the bandwidth limits that are configured for this host.

  Each limit is defined by its:

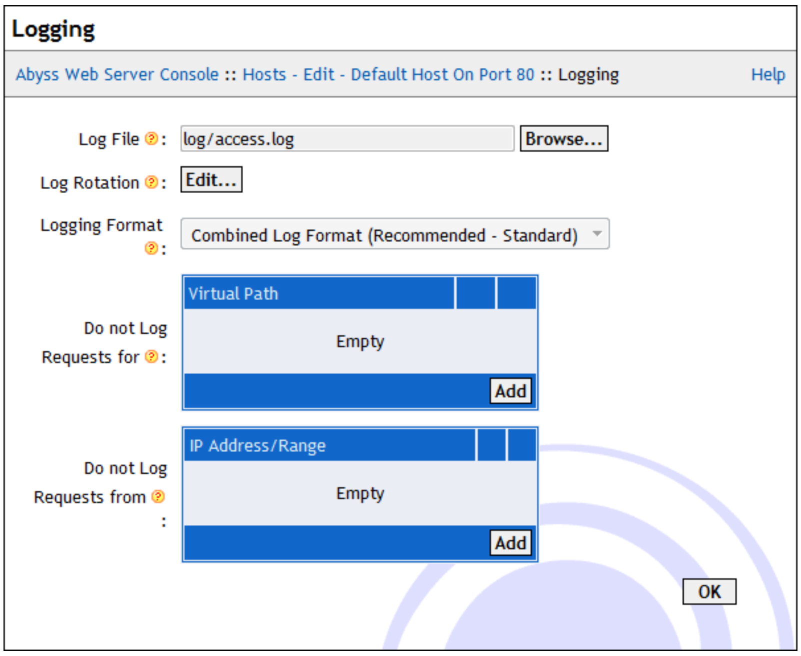
  - **Scope**: This table contains the virtual paths for which this bandwidth limit applies. A virtual path must begin with a slash `/`. It can also be a path patterns. In such a case, the limit applies for any virtual path that matches with the specified pattern. Refer to the "Patterns Format" appendix for more information about patterns.
  - **Maximum Bandwidth**: The amount of output bandwidth that the host should not exceed for all the requests which virtual path matches with one of paths listed in the **Scope** table. If set to **Unlimited**, no such bandwidth restriction will apply.
  - **Maximum Bandwidth Per IP Address**: The total amount of output bandwidth that the host should not exceed for all the requests made by a single IP address and which virtual path matches with one of paths listed in the **Scope** table. If this parameter is set to **`Unlimited`**, no limit will be applied.
  - **Maximum Requests Per IP Address**: The maximum number of simultaneous requests made by a single IP address to the host and which virtual path matches with one of paths listed in the **Scope** table. No limit will ever be applied if this parameter is set to **`Unlimited`**.

Note: *To set a limit on the bandwidth of the whole host, add a bandwidth limit with* `/` *as its scope.*

Note: *If the virtual path of a request matches with more than a single bandwidth limit, Abyss Web Server will respect all of them.*

Logging

Select **Logging** in the host configuration menu to display the **Logging** options.


Logging

Abyss Web Server Console :: Hosts - Edit - Default Host On Port 80 :: Logging

Log File: log/access.log

Log Rotation: Edit...

Logging Format: Combined Log Format (Recommended - Standard)

Do not Log Requests for:

Virtual Path
Empty

Do not Log Requests from:

IP Address/Range
Empty

Figure 5-19. Logging dialog

The dialog contains the following fields:

- **Log File**: The path of the log file. If it is relative, it is considered as a subpath of the server root. If empty, logging is disabled.
- **Log Rotation**: To prevent the log file from growing indefinitely, press **Edit...** to configure the way it is rotated:
  - No rotation is performed if **Type** is set to **Disabled**.
  - Rotation can occur when the size of the log file reaches a certain threshold set in **Maximum Log Size Before Rotation**. The rotated files are then stored in the directory set in **Rotated Files Path**. Rotated file names are made of the concatenation of the value in **Rotated File Names Prefix** and an ordinal number. The last rotated file has number 0. The previous one has number 1, the so forth.The maximum number of preserved rotated files is set in **Maximum Number of Retained Rotated Files** .
  - Rotation is time based when **Type** is set to **By Time**. Use **Rotation Frequency** to set when rotation should occur. If **Use Local Time** is unchecked, the clock is converted to UTC/GMT to determine if a period of time relative to the frequency lapsed and rotation has to happen or not. The rotated files are then stored in the directory set in **Rotated Files Path**. Rotated file names are made of the concatenation of the value in **Rotated File Names Prefix** and a time reference which depends on the value of **Rotation Frequency**. For example, if rotation is on yearly basis, the time

reference is the year only **`YYYY`**. If the rotation is on a hourly basis, the time reference is of the form **`HH-DD-MM-YYYY`**.

- **Logging Format**: This parameter sets the format to be used in the log file for the current host. It offers the choice between the standard formats **Common Log Format** or the **Combined Log Format**, and the custom logging formats defined under the **Logging Parameter** dialog in **Server Configuration**.
- **Do not Log Requests for**: This table contains the virtual paths for which logging is disabled. A virtual path must begin with a slash `/`. It can also be a path patterns. In such a case, the logging is disabled for any virtual path that matches with the specified pattern. Refer to the "Patterns Format" appendix for more information about patterns.
- **Do not Log Requests from**: This table contains the IP addresses or IP address ranges for which logging is disabled. Refer to the "IP Addresses and Ranges Format" appendix for more information about the IP addresses and ranges.

Reverse Proxy

Abyss Web Server can act as a reverse proxy to provide content transparently from another Web server behind it (commonly called a back-end server.) The proxied server can be another Web server or an application server with a restricted or even a trivial HTTP support that has to be shielded from external clients. Reverse proxying is also the recommended way to take advantage of JSP (Java Servlet Pages) and Tomcat Web applications.

Abyss Web Server automatically selects the best version of the HTTP protocol when communicating with a back-end server. It is also able to keep-alive connections with HTTP/1.1 back-end servers and sharing pools of connections between several requests and hosts. It also transparently supports the HTTP/1.1 Upgrade mechanism which allows it to relay WebSockets connections out-of-the-box. Abyss Web Server may instruct the client to downgrade from HTTP/2 to HTTP/1.1 if the back-end server is trying to use a HTTP/1.1 feature that cannot be proxied over HTTP/2.

Select **Reverse Proxy** in the host configuration menu to display a dialog containing the **Reverse Proxy Rules** table. Use that table to to edit, remove or add reverse proxy mapping rules rewriting rules.

Each reverse proxy mapping rule is defined by the following parameters:

- **Local Virtual Path**: The local virtual path on the current host which contents will be mapped to the remote virtual path on the reverse proxy.
- **Remote Host**: The host name or IP address of the back-end server.

- **Remote Port**: The port on which the Web server is listening on the back-end server.
- **Remote Virtual Path**: The virtual path on the remote host from which mapped contents are served.
- **HTTPS**: If checked, connect to the remote host using HTTPS instead of plain HTTP.
- **Advanced Parameters**: Press **Edit...** to access the advanced parameters dialog. This dialog exposes the following fields:
  - **Fix URLs in Cookies**: If checked, HTTP Cookies received from the back-end are rewritten to appear as if they were originated from the current host.
  - **Fix URLs in Location Response Headers**: If checked, the **`Location`**, **`Content-Location`**, and **`URI`** HTTP headers received from the back-end are rewritten to appear as if they were originated from the current host.
  - **Preserve Host Header**: If checked, the **`Host`** header of the original request is preserved and used in the request sent to the back-end server instead of the value of **Remote Host**. Some back-ends can use this feature to correctly generate links and redirection headers. Please note that **Preserve Host Header** should remain unchecked unless you are sure the back-end can deal with such altered requests.
  - **Extra Request Headers**: The HTTP headers defined in this table are added to each request targeting the the back-end.
  - **Timeout**: The timeout value in seconds is used to determine how long to wait for an inactive connection to the back-end before closing it.
  - **Connection Timeout**: The connection timeout value in seconds determines how long to wait for a connection to the back-end to be established.

Example: *An example of a reverse proxy rule*

Let's assume that the current host includes a reverse proxy rule defined with the following parameters:

- ***Local Virtual Path*: `/app/photo`**
- ***Remote IP Address*: `192.168.1.233`**
- ***Remote Port*: `8080`**
- ***Remote Virtual Path*: `/txp`**

If a client makes a request to the virtual path **`/app/photo/abc/test.py`** *on the current host, the local Web server will forward the request to the URL*

`http://192.168.1.233:8080/txp/abc/test.py`*. The retrieved result is sent back to the client.*

Restricted Downloads Areas

To prevent direct access to files contained in some directories, it is possible to create links to them which expire after a certain duration and which may be restricted to a certain IP address. Such links are to be generated using scripts and look like:

```
http://www.mysite.com/dir1/dir2/1d0dc35b10df59a45638a78228fdfe4d/5183d1da/file.txt
```

To configure virtual paths associated with such a behavior, select **Restricted Downloads Areas** in the host configuration menu. Each restricted downloads area is defined by the following parameters:

- **Virtual Path**: The virtual path associated with the restricted downloads area. This virtual path will be prepended to all URLs referring to that area.
- **Files Root Path**: The real directory which contains the files that are served from that area.
- **Secret String**: The secret string which is used to compute the hash included in the URL.
- **Timeout**: The maximum duration of validity of URLs referring to files in that area (in seconds.)
- **Limit To Originating IP**: If checked, the hash should be generated including the originating IP address and validity of URLs in that area takes into account that information.

An URL referring to a restricted download area is of the form: **`http://<your host name>:<host port>/vpath/hash/hextimestamp/file.ext`**:

- **`/vpath`** is the virtual path associated with the area.
- **`hextimestamp`** is the hexadecimal representation of the number of seconds since January 1st, 1970 GMT. The timestamp is used to determine if the link is still valid based on the **Timeout** value.
- **file.ext** is the path of the file inside the **Files Root Path**.
- **`hash`** is a hexadecimal string which is the value of **`MD5(secret + "/file.ext" + hextimestamp)`** if **Limit To Originating IP** is unchecked. Otherwise, it is the value of **`MD5(secret + "/file.ext" + hextimestamp + ip)`** where **`ip`** is the IP of the client.

***Example:** Script to generate a restricted download URL (PHP version)*

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
    <TITLE>
        Test
    </TITLE>
  </HEAD>

  <BODY>

 <?php

 $secret_string = "secret string";
 $virtual_path = "/downloads/";
 $limit_by_ip = false;
 $hex_timestamp = dechex(time());
 $file_name = "/file_to_protect.txt";

 if ($limit_by_ip) {
 $t = $secret_string . $file_name . $hex_timestamp . $_SERVER['REMOTE_ADDR'];
 }
 else {
 $t = $secret_string . $file_name . $hex_timestamp;
 }

 $hash = md5(mb_convert_encoding($t, "UTF-8"));
 $url = $virtual_path . $hash . "/" . $hex_timestamp . $file_name;

 echo "<TR><TD><A HREF=\"$url\">Click here to download the file</A>";
 ?>

   </BODY>
</HTML>
```

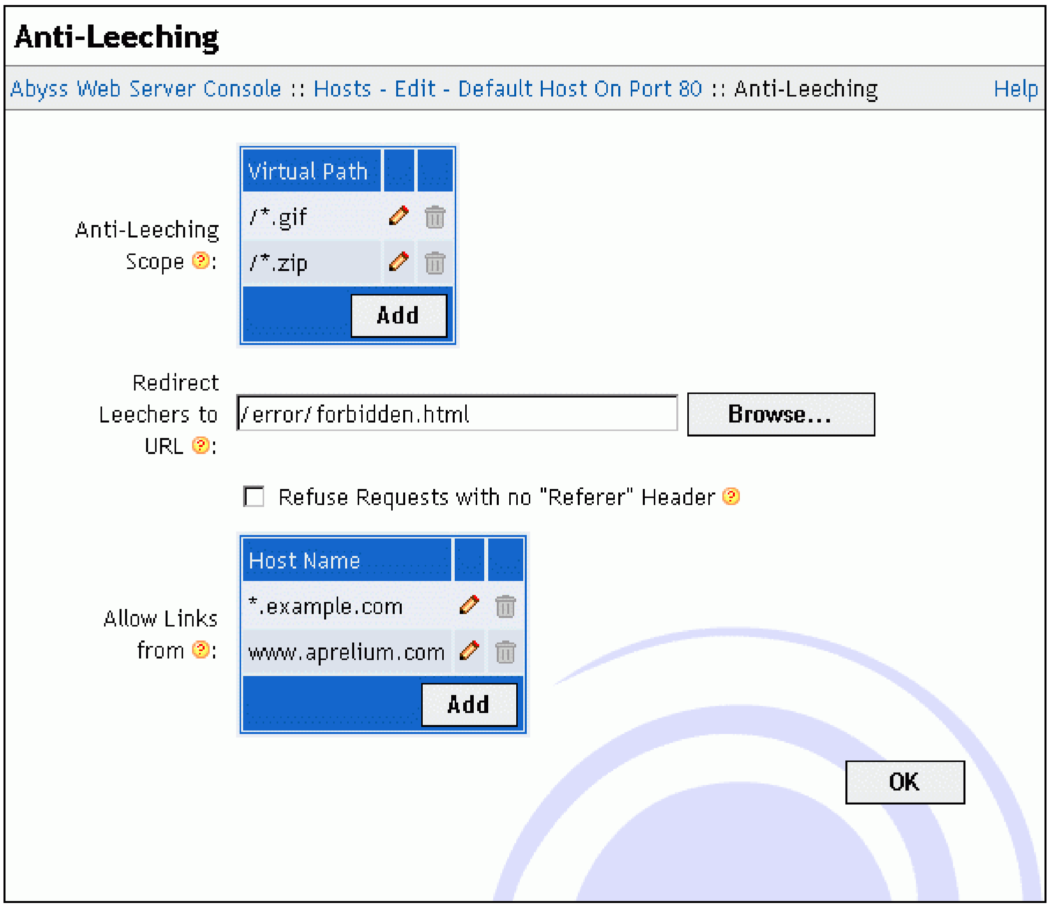
***Example:** Script to generate a restricted download URL (Python version)*

```
import os, time, hashlib

secret_string = "secret string"
virtual_path = "/downloads/"
limit_by_ip = False
hex_timestamp = "{0:x}".format(int(time.time()))
file_name = "/file_to_protect.txt"

if limit_by_ip:
  t = secret_string + file_name + hex_timestamp + os.environ["REMOTE_ADDR"]
else:
  t = secret_string + file_name + hex_timestamp

hash = hashlib.md5(t.encode('utf-8')).hexdigest()
url = virtual_path + hash + "/" + hex_timestamp + file_name

# Output CGI headers
print ("Content-Type : text/html")
print ()

# Output body
print ('<HTML><BODY><A SRC="%s">Click here to download the file</A></BODY></HTML>' % url)
```

Anti-Leeching

Anti-Leeching is a system that prevents web pages not belonging to your host from referring or linking to materials available in your web site. *Leeching* is also known as *cross-site linking*.

When a request is sent to the host, and if its virtual path is in the configured **Anti-Leeching Scope**, the server checks if the **`Referer`** header in the request matches with the current host or with one of the host names or the patterns in the **Allow Links from** table. The **`Referer`** header is usually set by the browser to indicate to the server from which web page the current URL was requested (or linked to.) If it does not match, the request is considered as a leeching attempt and is redirected to the URL configured in the **Redirect Leechers to URL** parameter.


Anti-Leeching

Abyss Web Server Console :: Hosts - Edit - Default Host On Port 80 :: Anti-Leeching — Help

Anti-Leeching Scope:

Virtual Path
/*.gif
/*.zip

Redirect Leechers to URL: /error/forbidden.html

☐ Refuse Requests with no "Referer" Header

Allow Links from:

Host Name
*.example.com
www.aprelium.com

Figure 5-20. Anti-Leeching dialog

The dialog contains the following fields:

- **Anti-Leeching Scope**: Any request which virtual path is in or matches with one of listed virtual paths in this table is protected against leeching.
- **Redirect Leechers to URL**: A request that is considered as a leeching request is redirected to this URL. It can be a full URL referencing another web site or a virtual path pointing on a document in the current host. If this field is empty, the server returns error 403 to the client.

- **Refuse Requests with no "Referer" Header**: If checked, the system adopts a strict behavior and will consider requests that do not contain a `Referer` header as anti-leeching requests. It is not recommended to check this parameter since it may prevent browsers configured to not send the `Referer` header from viewing documents in your host.
- **Allow Links from**: This table contains the names of hosts that are allowed to link to files and materials in the current host and that are not monitored against leeching. A host name in this table can also be a pattern such as `*.mysite.com`. Refer to the "Patterns Format" appendix for more information about patterns.

Statistics

The **Statistics** dialog displays a set of statistics on the host's activity since the server installation or the last host statistics reset:

- **Total Uptime**: The total time during which the host was up.
- **Uptime since Last Restart**: The time during which the host was up since the last server start or restart. If the host is stopped, its value is zero.
- **Total Hits**: The total number of requests processed by the current host.
- **Compressed Hits**: The number of requests processed by the current host that resulted in compressed responses.
- **Error Hits**: The number of requests targeting the current host that resulted in an error.
- **HTML Hits**: The number of requests the host replied to by a document which MIME type was `text/html`.
- **Image Hits**: The number of requests the host replied to by a document which MIME type starts with `image/`.
- **Not Modified Hits**: The number of requests for which the host detected that the requested document has not changed.
- **Transferred Data**: The total size of the payload sent by the current host to the clients.
- **Compression Savings**: The amount of data that was saved thanks to the compressed responses sent by the current host to the clients.

The host statistics are refreshed automatically every 10 seconds. You can also press **Refresh** for immediate refreshing. They can be reset by pressing **Reset**.

Note: *The statistics are preserved when the server is shut down or when the host is stopped.*

Note: *The host statistics are also available in real time to scripts and XSSI pages. Refer to "CGI environment variables" for more information.*

Chapter 6. CGI, FastCGI, and ISAPI

Abyss Web Server supports CGI (Common Gateway Interface), FastCGI, and ISAPI Extensions (Internet Server Advanced Programming Interface Extensions) standards for dynamic content generation. It can process scripts by CGI, FastCGI, and ISAPI interpreters and serve content generated by CGI, FastCGI, and ISAPI applications.

Setting up an interpreter

To setup a language interpreter to be used by scripts in Abyss Web Server, follow these instructions:

- Download the language package.
- Install it. Refer to the documentation accompanying it for details about this operation.
- In the **Scripting Parameters** dialog, use the **Interpreters** table to add the new interpreter and the filename extensions it handles.
- Scripts requiring this interpreter to run must have their filename extensions set to one of the declared extensions.

Visit `https://aprelium.com/abyssws` to get updated information about downloading and setting up PHP, Perl, Python, and other scripting languages.

How are they run?

When the server is asked for a document, it first checks if it is a script. Scripts must belong to one of the script paths or their subpaths, or must match with one of the path patterns listed in the Script Paths table.

Abyss Web Server is able to run standalone CGI or FastCGI executables. These executables can be GUI or non-GUI Windows applications or MS-DOS EXE or COM programs. The **Script Paths** table must contain explicitly the general patterns `/*.exe` and/or `/*.com` (or more path restrictive patterns ending with `*.exe` and/or `*.com`) in order to have them executed. Otherwise, the executable files are downloaded to the client browser.

Abyss Web Server can also run standalone ISAPI extensions. The **Script Paths** table must contain explicitly the virtual path of a module or a matching pattern to have it executed. A file is considered as being an ISAPI module if its file name extension matches one of the declared ISAPI file name extensions.

For other kinds of scripts, the server tries to find the suitable interpreter able to run them by:

- Checking the **Interpreters** table and trying to match the script's extension with a declared interpreter.
- Using the Windows Registry to match the script's extension with an available interpreter.
- Reading the first line of the script. If it begins with `#!`, the rest of the line must contain a valid interpreter path.

If none of these steps is successful, the file is considered as a normal document and will be sent to the browser.

Abyss Web Server automatically detects NPH (Non Parsed Headers) scripts. NPH scripts output is sent directly to the browser without prior header decoding by the server. It is up to NPH scripts to generate correctly all the necessary HTTP response headers. Abyss Web Server considers a script to be NPH if the first line it outputs starts with the string `HTTP/`.

With the advent of HTTP/2, problems may arise with NPH scripts which carry low-level HTTP/1.0 or HTTP/1.1 protocol details. So it is recommended to disable HTTP/2 for them by adding their virtual paths in the **HTTP/1.1 Required** table. By default, the mentionned table already contains `*/nph-` pattern which matches with any script which file name starts with `nph-`. This naming convention was the defacto standard for NPH in legacy Web servers and was enforced by legacy Perl language CGI packages.

CGI environment variables

Before running a CGI application or interpreter, Abyss Web Server sets its environment variables in conformity with the CGI/1.1 specification (as described in `https://www.ietf.org/rfc/rfc3875.txt`) and adds variables declared in the **Custom Environment Variables** table as well as the system environment variables. Some Abyss Web Server specific variables are also set.

The following list contains the variables documented in the CGI/1.1 specification and some variables commonly set by web servers:

- `PATH_INFO`: The extra path information, as given in the requested URL. In fact, scripts can be accessed by their virtual path, followed by extra information at the end of this path. The extra information is sent in `PATH_INFO`.

- **`PATH_TRANSLATED`**: The virtual-to-real mapped version of **`PATH_INFO`**.
- **`SCRIPT_NAME`**: The virtual path of the script being executed.
- **`SCRIPT_FILENAME`** and **`REQUEST_FILENAME`**: The real path of the script being executed.
- **`SCRIPT_URI`**: The full URL to the current object requested by the client.
- **`URL`**: The full URI of the current request. It is made of the concatenation of **`SCRIPT_NAME`** and **`PATH_INFO`** (if available.)
- **`SCRIPT_URL`** and **`REQUEST_URI`**: The original request URI sent by the client.
- **`REQUEST_METHOD`**: The method used by the current request; usually set to **`GET`** or **`POST`**.
- **`QUERY_STRING`**: The information which follows the **`?`** character in the requested URL.
- **`CONTENT_TYPE`**: The MIME type of the request body; set only for POST or PUT requests.
- **`CONTENT_LENGTH`**: The length in bytes of the request body; set only for POST or PUT requests.
- **`AUTH_TYPE`**: The authentication type if the client has authenticated itself to access the script.
- **`AUTH_USER`** and **`REMOTE_USER`**: The name of the user as issued by the client when authenticating itself to access the script.
- **`ALL_HTTP`**: All HTTP headers sent by the client. Headers are separated by carriage return characters (ASCII 13 - **`\n`**) and each header name is prefixed by **`HTTP_`**, transformed to upper cases, and **`-`** characters it contains are replaced by _ characters.
- **`ALL_RAW`**: All HTTP headers as sent by the client in raw form. No transformation on the header names is applied.
- **`SERVER_SOFTWARE`**: The web server's software identity.
- **`SERVER_NAME`**: The host name or the IP address of the computer running the web server as given in the requested URL.
- **`SERVER_ADDR`**: The IP address of the computer running the web server.
- **`SERVER_PORT`**: The port to which the request was sent.
- **`GATEWAY_INTERFACE`**: The CGI Specification version supported by the web server; always set to **`CGI/1.1`**.
- **`SERVER_PROTOCOL`**: The HTTP protocol version used by the current request.
- **`REMOTE_ADDR`**: The IP address of the computer that sent the request.
- **`REMOTE_PORT`**: The port from which the request was sent.

- **DOCUMENT_ROOT**: The absolute path of the web site files. It has the same value as **Documents Path**.
- **INSTANCE_ID**: The numerical identifier of the host which served the request. On Abyss Web Server X1, it is always set to 1 since there is only a single host.
- **APPL_MD_PATH**: The virtual path of the deepest alias which contains the request URI. If no alias contains the request URI, the variable is set to **/**.
- **APPL_PHYSICAL_PATH**: The real path of the deepest alias which contains the request URI. If no alias contains the request URI, the variable is set to the same value as **DOCUMENT_ROOT**.
- **IS_SUBREQ**: It is set to **true** if the current request is a subrequest, i.e. a request not directly invoked by a client. Otherwise, it is set to **true**. Subrequests are generated by the server for internal processing. XSSI includes for example result in subrequests.

If the current request is served by an SSL enabled host, the following SSL related variables are also available:

- **SSL_PROTOCOL**: The SSL protocol version. It could be **SSLv2**, **SSLv3**, **TLSv1**, **TLSv1.1**, **TLSv1.2**, or **TLSv1.3**.
- **SSL_CIPHER** and **HTTPS_CIPHER**: The SSL cipher used to encrypt the current connection.
- **SSL_CIPHER_EXPORT**: It is set to **true** if the cipher is conforming to US export restrictions (i.e. it is limited to 56 bit symmetric keys and 1024 asymmetric keys).
- **SSL_CIPHER_USEKEYSIZE** and **HTTPS_SECRETKEYSIZE**: The number of cipher bits currently used.
- **SSL_CIPHER_ALGKEYSIZE** and **HTTPS_KEYSIZE**: The maximum number of cipher bits that could be used.
- **SSL_SERVER_M_VERSION**: The version of the server's certificate.
- **SSL_SERVER_M_SERIAL**: The serial number of the server's certificate.
- **SSL_SERVER_V_START**: The date corresponding to the start of the validity the server's certificate.
- **SSL_SERVER_V_END**: The date corresponding to the end of the validity the server's certificate.
- **SSL_SERVER_A_SIG**: The algorithm used to sign the server's certificate.
- **SSL_SERVER_A_KEY**: The encryption type of the private key of the server's certificate.

- **`SSL_SERVER_S_DN`** and **`HTTPS_SERVER_SUBJECT`**: The value of the subject's DN (distinguished name) of the server's certificate.
- **`SSL_SERVER_S_DN_`***component*: The value of the *component* of the subject's DN (distinguished name) of the server's certificate. *component* can be **`CN`** (Common Name), **`C`** (Country), **`ST`** or **`SP`** (State/Province), **`L`** (Locality), **`O`** (Organization), **`OU`** (Organization Unit), **`T`** (Title), **`I`** (Initials), **`G`** (Given Name), **`S`** (Surname), **`D`** (Description), **`Email`** (Contact Email).
- **`SSL_SERVER_I_DN`** and **`HTTPS_SERVER_ISSUER`**: The value of the issuer's DN (distinguished name) of the server's certificate.
- **`SSL_SERVER_I_DN_`***component*: The value of the *component* of the issuer's DN (distinguished name) of the server's certificate.

If the current request is served over a HTTP/2 connection, the following variables are also available:

- **`HTTP2`**: It is set to **`on`**.
- **`H2_STREAM_ID`**: A number uniquely identifying the stream within the HTTP/2 connection on which the current request is being served.
- **`H2_STREAM_TAG`**: A tag uniquely identifying current stream within all HTTP/2 connections of the server. The tag is made of two numbers separated by a hyphen character.

The Abyss Web Server specific variables provide a means to access to server and host statistics. They are listed below:

- **`X_ABYSS_STATS_SERVER_UPTIME`**: The total time in seconds during which the server was up.
- **`X_ABYSS_STATS_SERVER_CURRENT_UPTIME`**: The total time in seconds during which the server was up since the last start or restart.
- **`X_ABYSS_STATS_SERVER_TOTAL`**: The total number of processed requests for all the hosts. It includes also bad requests that were not targeting a specific host and that resulted in an error.
- **`X_ABYSS_STATS_SERVER_ERROR`**: The number of requests for which the server replied by an error.
- **`X_ABYSS_STATS_SERVER_HTML`**: The number of requests the server replied to by a document which MIME type was **`text/html`**.
- **`X_ABYSS_STATS_SERVER_IMAGE`**: The number of requests the server replied to by a document which MIME type starts with **`image/`**.
- **`X_ABYSS_STATS_SERVER_NOTMODIFIED`**: The number of requests for which the server detected that the requested document has not changed.

- **`X_ABYSS_STATS_SERVER_OUTPUT`**: The total size in bytes of the payload sent by the server to the clients.
- **`X_ABYSS_STATS_SERVER_COMPRESS_TOTAL`**: The total number of compressed responses for all the hosts.
- **`X_ABYSS_STATS_SERVER_COMPRESS_OUTPUT_TOTAL`**: The total size in bytes of the payload sent by the server to the clients in compressed responses.
- **`X_ABYSS_STATS_SERVER_COMPRESS_ORIGINAL_OUTPUT`**: The total original uncompressed size in bytes of the payload sent by the server to the clients in compressed responses.
- **`X_ABYSS_STATS_HOST_UPTIME`**: The total time in seconds during which the current host was up.
- **`X_ABYSS_STATS_HOST_CURRENT_UPTIME`**: The time in seconds during which the current host was up since the last server start or restart.
- **`X_ABYSS_STATS_HOST_TOTAL`**: The total number of requests processed by the current host.
- **`X_ABYSS_STATS_HOST_ERROR`**: The number of requests targeting the current host that resulted in an error.
- **`X_ABYSS_STATS_HOST_HTML`**: The number of requests the current host replied to by a document which MIME type was **`text/html`**.
- **`X_ABYSS_STATS_HOST_IMAGE`**: The number of requests the current host replied to by a document which MIME type starts with **`image/`**.
- **`X_ABYSS_STATS_HOST_NOTMODIFIED`**: The number of requests for which the current host detected that the requested document has not changed.
- **`X_ABYSS_STATS_HOST_OUTPUT`**: The total size in bytes of the payload sent by the current host to the clients.
- **`X_ABYSS_STATS_HOST_COMPRESS_TOTAL`**: The total number of compressed responses for the current host.
- **`X_ABYSS_STATS_HOST_COMPRESS_OUTPUT_TOTAL`**: The total size in bytes of the payload sent by the current host to the clients in compressed responses.
- **`X_ABYSS_STATS_HOST_COMPRESS_ORIGINAL_OUTPUT`**: The total original uncompressed size in bytes of the payload sent by the current host to the clients in compressed responses.

In addition to these variables, all header lines received in the request are added to the environment with the prefix **`HTTP_`** followed by the header name in upper cases. All **`-`** characters in the header name are changed to underscore _ characters. For example, **`User-Agent`** is translated to **`HTTP_USER_AGENT`**.

If the request results from an internal redirection (from an XSSI document or if it is used as a custom error page for example), the environment variables of the parent request are also added and each variable name is prefixed by **`REDIRECT_`**. The parent request's status code is stored in the special variables **`REDIRECT_STATUS`** and **`REDIRECT_STATUS_CODE`**. The cookies of the parent request are also passed to the redirected request in the **`COOKIES`** environment variable. **`REDIRECT_STATUS_CODE`** may seem redundant but it is actually useful when with PHP scripts as some PHP configurations require setting **`REDIRECT_STATUS`** to a fixed value.

Example: *Using the redirection information in scripts*

If a script is invoked after an internal redirection, it can retrieve some interesting information from the environment about its parent request such as:

- ***`REDIRECT_SCRIPT_NAME`****: The virtual path of the parent document.*
- ***`REDIRECT_STATUS_CODE`****: The status code of the parent request. It is useful when writing scripts that generate custom error pages.*

Note: *The CGI environment variables are not only useful for CGI applications and interpreters. They are also available and used in XSSI pages and ISAPI interpreters and applications.*

FastCGI

Abyss Web Server features a full support for the FastCGI responder role as described in the FastCGI specification version 1.0 originally published in **`https://fastcgi-archives.github.io/FastCGI_Specification.html`**.

It supports mono-threaded and multi-threaded FastCGI applications and automatically spawns the required number of FastCGI processes to accommodate the current server load. Processes that have not been used after a period of time (as set in **FastCGI Processes Timeout**) are aborted. Abyss Web Server supports also connecting to and using remote FastCGI servers.

FastCGI developers benefit also from advanced debugging capabilities in Abyss Web Server since it can be configured to log errors only or all the input/output and low level information exchanged between the FastCGI application and the server.

ISAPI Extensions

Abyss Web Server supports a large subset of the ISAPI 5.0 specification suitable for running most of the ISAPI extensions available to date. ISAPI extensions are loaded in the server process space on demand. They remain there until the server is restarted or stopped. Since ISAPI extensions *live* in the server process space (contrarily to CGI application which are launched in their own process), any ISAPI error or crash affects the whole server. But thanks to the APX (Anti-crash Protection eXtension) architecture of Abyss Web Server such problems can be immediately detected and the server is restarted with almost no noticeable downtime.

Abyss Web Server makes ISAPI development easier thanks to its advanced debugging capabilities: With the help of the **Debugging Level** parameter in **ISAPI Parameters**, you can configure the server to log unhandled callback to the ISAPI extension, all callbacks, or all callbacks with all the exchanged input/output between the extension and the server.

X-Sendfile Support

Scripts and Web applications using CGI, FastCGI or ISAPI (as well as ASP.NET applications) can accelerate file service by emitting the special CGI header **`X-Sendfile`**. Such an acceleration is actually performed by delegating to the low-level parts of the Web server the service of the static file to the client.

For that header to be processed, the script or Web application must be in a script path declared with **X-Sendfile Support** enabled as outlined in the description of "Script Paths".

Example: *A very simple script taking advanatge of X-Sendfile support (PHP version)*

```
<?php

/* Output the special header */
header("X-Sendfile: hello.txt")

?>
```

The virtual path of the above script should be added to ***Script Paths****. The declared script path should have its* ***X-Sendfile Support*** *enabled and its* ***Files Root Path*** *set to the real directory from which the file* **`hello.txt`** *should be served.*

Chapter 7. eXtended Server Side Includes

What are XSSI?

XSSI (eXtended Server Side Includes) are a set of directives that can be put in a HTML page and interpreted by the web server while it sends the page to a client. You can use XSSI to insert in your HTML page:

- The value of an environment variable.
- The size and the last modification time of a file.
- A file.
- The output of a script or a shell command.

XSSI also supports conditional processing as well as declaring variables and setting their values.

XSSI Syntax

```
<!-- #directive attr1="value1" attr2="value2" ... -->
```

Directives are enclosed in standard HTML comments. So if XSSI processing is not enabled, your browser will ignore them. Otherwise, each directive is evaluated and replaced by its results.

Note: *Enclosing an attribute's value in double quotes is not mandatory if it does not contain whitespaces or references to variables.*

Note: *A whitespace should precede the comment terminator* `-->`.

Attribute Values

An attribute value can be enclosed between single quotes (`'`) or backticks (`` ` ``). In such a case, only character escaping is performed on it; and the special characters are the quote or the backtick and the backslash (`\`).

An attribute value can also be enclosed between double quotes (`"`). In such a case, both character escaping and variables expansion are performed on it; and the special characters are `/`, `"`, `{`, `}` and `$`.

Attribute values can be unquoted if they do not contain space characters.

Character Escaping

To preserve the literal value of a special character (\ and depending on the context **/**, **"**, **{**, **}**, **$**, **'** or **\`**), precede it with a backslash \ character.

Variables Expansion

Parts of the attribute value string that are of the form **`$VARNAME`** are substituted with the actual value of variable **`VARNAME`**. If **`VARNAME`** is not declared, it is replaced by an empty string. The variable name must be enclosed between braces as in **`${VARNAME}`** if the characters following it may be considered as part of the name.

Example: *Variable Expansion Example*

*If variable **`DOCUMENT_NAME`** is set to **`test.shtml`** and variable **`REMOTE_PORT`** is set to **`12345`**, then the directive:*

```
<!-- #set var="Foo" value="${DOCUMENT_NAME}_$REMOTE_PORT" -->
```

*will set the variable **`Foo`** to **`test.shtml_12345`**.*

Directives

#config

This directive is used to override the default XSSI settings while processing the XSSI file.

Form 1: `<!-- #config errmsg="error message" -->`

`#config errmsg` is used to change the message sent to the client when an XSSI error occurs. By default, XSSI errors are reported in a detailed manner to allow easy debugging.

Form 2: `<!-- #config sizefmt="size_format" -->`

`#config sizefmt` controls the way the server displays XSSI file sizes. **`size_format`** can have two values:

- **`ABBREV`**: displays file sizes in kilobytes (KB) or megabytes (MB).
- **`BYTES`**: displays file sizes in bytes. It is the default.

Form 3: `<!-- #config timefmt="time_format" -->`

`#config timefmt` specifies the format in which XSSI dates are inserted. **`time_format`** must contain formatting tokens that are substituted by the corresponding parts of the date. The available tokens are listed below with the description of their substitution:

- **`%a`**: Abbreviated English name for day of the week.
- **`%A`**: Complete English name for day of the week.
- **`%b`**: Abbreviated English month name.
- **`%B`**: Complete English month name.
- **`%c`**: Date and time representation. It is the equivalent of **`%m/%d/%y %H:%M:%S`**.
- **`%d`**: Day of the month (01-31).
- **`%e`**: Day of the month (1-31). If it is a single digit, it is padded with a space.
- **`%H`**: Hour in 24-hour format (00-23).
- **`%I`**: Hour in 12-hour format (01-12).
- **`%j`**: Day of the year as a decimal number (001-366).
- **`%m`**: Month as a decimal number (01-12).
- **`%M`**: Minute as a decimal number (00-59).
- **`%p`**: A.M. or P.M. indicator for 12-hour format.
- **`%S`**: Second as a decimal number (00-59).
- **`%U`**: Week of the year as a number (00-51). Sunday is the first day of the week.
- **`%w`**: Day of the week as a number (0-6). Sunday is the first day of the week.
- **`%W`**: Week of the year as a number (00-51). Monday is the first day of the week.
- **`%x`**: Date representation. It is the equivalent of **`%m/%d/%y`**.
- **`%X`**: Time representation. It is the equivalent of **`%H:%M:%S`**.
- **`%y`**: Year without the century (00-99).
- **`%Y`**: Year with the century (0000-9999).
- **`%z`** and **`%Z`**: Time zone name; empty if unknown.
- **`%%`**: Percent sign.

#echo

Form: `<!-- #echo var="variable_name" -->`

`#echo` inserts the value of an XSSI environment variable. XSSI environment variables include those set using the **`#set`** directive, those available for CGI scripts (see "CGI environment variables" in the "CGI, FastCGI, and ISAPI" chapter), and five special variables:

- **`DOCUMENT_NAME`**: The name of the processed XSSI file.
- **`DOCUMENT_URI`**: The virtual path to the current processed file.
- **`DATE_LOCAL`**: The current local date.
- **`DATE_GMT`**: The current date expressed in UTC (coordinated universal time, formerly known as Greenwich Mean Time, or GMT).
- **`LAST_MODIFIED`**: The last modification date of the processed file expressed in UTC (coordinated universal time).

If **`variable_name`** is not a valid XSSI environment variable, the directive is ignored.

#exec

Form 1: `<!-- #exec cgi="cgi_file" -->`

`#exec cgi` executes a script and inserts its output. If **`cgi_file`** begins with a **`/`**, it is considered as a virtual path and the directive will behave as if it is **`<!-- #include virtual="cgi_file" -->`**. If **`cgi_file`** does not begin with a **`/`**, it is considered as a relative file name and the server will try to locate it inside the directory containing the currently processed XSSI file. If it is found there, **`#exec cgi`** will behave as if it is **`<!-- #include file="cgi_file" -->`**. Otherwise, **`cgi_file`** will be searched inside the virtual paths listed in **'#exec cgi' Search Paths** in the **XSSI Parameters** of the current host.

Form 2: `<!-- #exec cmd="shell_command" -->`

`#exec cmd` executes a shell command and inserts its output. The shell executable path is found in the system environment variable **`COMSPEC`**. If this variable is empty or does not exist, **`command.com`** is used as the default shell.

*Note: **`#exec cmd`** execution must be used with extreme care. If users can edit or modify HTML pages in your web sites (which is the case if you have a forum or a guestbook for instance), you must disable it. In fact, they can include XSSI tags with dangerous **`#exec cmd`** directives and remotely invoke them.*

#flastmod

Form 1: `<!-- #flastmod file="file_name" -->`

`#flastmod file` inserts the last modification date of the file **`file_name`**. **`file_name`** is a relative path from the directory containing the processed XSSI file. It file can only be in the same directory or in one of its subdirectories.

Form 2: `<!-- #flastmod virtual="virt_path" -->`

`#flastmod virtual` inserts the last modification date of the file which virtual path is **`virt_path`**.

#fsize

Form 1: `<!-- #fsize file="file_name" -->`

`#fsize file` inserts the size of the file **`file_name`**. **`file_name`** is a relative path from the directory containing the processed XSSI file. It can only be in the same directory or in one of its subdirectories.

Form 2: `<!-- #fsize virtual="virt_path" -->`

`#fsize virtual` inserts the size of the content resulting from requesting **`virt_path`** from the server. If **`virt_path`** is a static file, its size will be inserted. If it is a CGI script or a XSSI file, it is evaluated by the server and the size of the resulting output is inserted.

#include

Form 1: `<!-- #include file="file_name" -->`

`#include file` instructs the web server to insert a file which must contain valid HTML contents. **`file_name`** is a relative path from the directory containing the processed XSSI file. The included file can only be in the same directory or in one of its subdirectories.

Form 2: `<!-- #include virtual="virt_path" -->`

`#include virtual` inserts the content resulting from requesting **`virt_path`** from the server. If **`virt_path`** is a static file, its contents will be included "as is". If it is a CGI script or a XSSI file, it is evaluated by the server and the results are inserted. **`virt_path`** must contain or generate valid HTML content.

#printenv

Form: `<!-- #printenv -->`

`#printenv` outputs the current CGI environment variables. For more information, refer to the "CGI environment variables" section.

#set

Form: `<!-- #set var="variable_name" value="variable_value" -->`

`#set` sets the value of the variable **`variable_name`** to **`variable_value`**. If no variable called **`variable_name`** exists, it is created before being set. **`#set`** can be used to change the value of CGI variables or any already declared variable.

Conditional blocks

Syntax

Conditional blocks in XSSI look as follows:

```
<!-- #if expr="test_expr_1" -->
        Content to process if test_expr_1 is true
<!-- #elif expr="test_expr_2" -->
        Content to process if test_expr_2 is true
        .
        .
<!-- #elif expr="test_expr_n" -->
 Content to process if test_expr_n is true
<!-- #else -->
 Content to process if all the above tests are false
<!-- #endif -->
```

`#elif` sections are optional. A conditional block can have zero or many of them. The **`#else`** is optional and a conditional block can have either no **`#else`** or exactly one.

Conditional blocks can be nested.

Expression evaluation

The value of the **`expr`** attribute inside **`#if`** and **`#elif`** directives is considered as an expression which evaluates to true or false according to these rules:

string

Evaluates to true if *string* is not empty.

string_1 **<** *string_2*
string_1 **<=** *string_2*
string_1 **>** *string_2*
string_1 **>=** *string_2*

Performs a lexicographic comparison between *string_1* and *string_2*.

string_1 **=** *string_2*
string_1 **==** *string_2*
string_1 **!=** *string_2*

Tests the equality (with **=** or **==**) or inequality (with **!=**) of *string_1* and *string_2*.

If *string_2* has the form **/***regex***/**, a regular expression matching operation will take place instead. Any literal slash character (**/**) in *regex* has to be escaped by preceding it with a backslash character (****).

Matched sub-strings are captured inside the variables **$0**, **$1**, ..., **$9**. For more information about backreferences, refer to the "Regular Expressions" appendix.

test_expr_1 **&&** *test_expr_2*

Evaluates to true if both *test_expr_1* and *test_expr_2* expressions are true.

test_expr_1 **||** *test_expr_2*

Evaluates to true if at least one of the expressions *test_expr_1* and *test_expr_2* is true.

! *test_expr*

Evaluates to true if *test_expr* is false.

(*test_expr* **)**

Evaluates to true if *test_expr* is true. Parentheses are used to enclose sub-expressions and supersede the priorities of their operators.

! has higher priority than **&&** and **||** which at their turn have higher priority than **=**, **!=**, **==**, **<=**, **>=**, **<**, and **>**.

Example: *Variable Expansion in Expressions*

If a string operand (such as string, string_1 or string_2 in the above syntax list) includes a variable expansion that may contain a sequence similar to one of the operators, it is recommended to enclose it between quotes to avoid any ambiguity.

*For example, if variable **HTTP_COOKIE** is set to **lang=en**, the following directive will fail with an error because the expansion of the variable includes the character **=** which is considered as an equality test:*

```
<!-- #if expr="$HTTP_COOKIE = /lang=(.*)/" -->
 Language code is <!-- #echo text="$1" -->
<!-- #else -->
 No language is set.
<!-- #endif -->
```

The proper way to do the test without errors is to enclose the expansion between quotes as in :

```
<!-- #if expr="'$HTTP_COOKIE' = /lang=(.*)/" -->
 Language code is <!-- #echo text="$1" -->
<!-- #else -->
 No language is set.
<!-- #endif -->
```

Encoding

In Abyss Web Server, the directives **`#echo`**, **`#exec`**, **`#fsize`**, **`#flastmod`**, and **`#include`** accept an optional argument **`encoding`**.

The value of this argument determines which conversion method is to be applied to the output of an XSSI directive. It can be:

- **`none`**: No conversion is performed and the output is transmitted without any modification.
- **`entity`**: The output is converted using the XML/HTML escaping: **`<`** is converted to **`&lt;`**, **`>`** to **`&gt;`**, **`&`** to **`&amp;`**, **`"`** to **`&quot;`**, and **`'`** to **`&#39;`**.
- **`url`**: All non-alphanumerical characters in the output except **`-_.!~*'()/`** are replaced with a percent sign `%` followed by two hexadecimal digits. This conversion conforms to the URL encoding scheme as described in https://www.ietf.org/rfc/rfc1738.txt and is useful to output well formed URLs.
- **`reverse-url`**: Sequences made of a percent sign `%` followed by two hexadecimal digits are decoded according to the URL encoding scheme.

If no **`encoding`** argument is present in **`#echo`**, **`#fsize`**, or **`#flastmod`** directives, the server will perform **`entity`** conversion.

If the **`encoding`** argument is present in **`#exec`** or **`#include`** directives, the server will perform no conversion of the output.

Example: *Using the encoding argument in #include*

```
<HTML>
    <HEAD><TITLE>XSSI Test</TITLE></HEAD>
    <BODY>
         The HTML source code of page test.html is:
         <PRE><!-- #include file="test.html" encoding="entity" --></PRE>
    </BODY>
</HTML>
```

Chapter 8. Custom Directory Listings

Directory listings are fully customizable in Abyss Web Server. If you do not like their standard look, you can provide the server with your listing template. Advanced users can do better by writing a script to generate custom listings.

Defining a template

A directory listing template is defined by its:

- **MIME Type**: It is by default **`text/html; charset=utf-8`**. Since file names are provided using the UTF-8 text encoding, the template MIME type should always contain this text encoding (charset) value.
- **Header**: It is the header of the listing page that will be rendered.
- **Body Line**: It is the template of each file or directory line in the listing.
- **Footer**: It is the footer of the listing page.

When generating a directory listing using a template, the server creates a special XSSI environment before processing the header, the line description for every file in the listing, and the footer. So you can (in fact, you should) use XSSI directives to insert information about every file in the generated page.

While processing the line description for a given file, the server sets the following environment variables in addition to the standard CGI environment variables:

- **`DIRLIST_FILE_NAME`**: The name of the current file. This name is UTF-8 encoded.
- **`DIRLIST_FILE_URL`**: The relative URL of the current file.
- **`DIRLIST_FILE_SIZE`**: The size of the current file. It will be conforming to the current XSSI size format. This parameter can be configured in the XSSI section of the console or temporarily modified using the **`<!-- #config sizefmt="size_format" -->`** directive. If the current file is a directory, this variable is set to the **-** (hyphen) symbol.
- **`DIRLIST_FILE_DATE`**: The last modification date of the current file expressed in UTC (coordinated universal time). It will be conforming to the current XSSI time format. This parameter can be configured in the XSSI section of the console or temporarily modified using the **`<!-- #config timefmt="time_format" -->`** directive.

- **`DIRLIST_FILE_LOCAL_DATE`**: The last modification date of the current file expressed in local time. It will be conforming to the current XSSI time format. This parameter can be configured in the XSSI section of the console or temporarily modified using the **`<!-- #config timefmt="time_format" -->`** directive.
- **`DIRLIST_FILE_MIME_TYPE`**: The MIME type of the current file. If the file is a directory, the value of this variable is set to **`Directory`**.
- **`DIRLIST_FILE_MIME_MAIN_TYPE`**: The main MIME type of the current file. For example, if its MIME type is **`text/html`**, this variable will contain **`text`**. If the file is a directory, the value of this variable is set to **`Directory`**
- **`DIRLIST_FILE_MIME_SUB_TYPE`**: The MIME sub-type of the current file. For example, if its MIME type of is **`text/html`**, this variable will be set to **`html`**. If the file is a directory, the value of this variable will contain **`Directory`**.
- **`DIRLIST_FILE_TYPE`**: The type of the current file. It is set to **`file`** if the current file is a regular file, to **`dir`** if it is a directory, or to **`parentdir`** for the special directory `..` which points to the parent directory
- **`DIRLIST_FILES_COUNT`**: The total number of the files in the directory listing (including the regular files, the directories, and the special directory `..` if available.)
- **`DIRLIST_FILE_INDEX`**: The index of the current file in the listing. The value of this variable is always between 1 and **`DIRLIST_FILES_COUNT`**.

Example: *A simple directory listing template*

- ***MIME Type:***

```
text/html; charset=utf-8
```

- ***Header:***

```
<HTML>
  <HEAD>
    <TITLE>
      Index of <!-- #echo var="URL" encoding="reverse-url" -->
    </TITLE>
  </HEAD>

  <BODY>
    <TABLE BORDER=0>
      <TR>
        <TD>Name</TD>
        <TD>Size</TD>
        <TD>Date</TD>
        <TD>MIME Type</TD>
      </TR>
```

- ***Body Line:***

```
      <TR>
        <TD>
          <A HREF="<!-- #echo var="DIRLIST_FILE_URL" -->">
            <!-- #echo var="DIRLIST_FILE_NAME" -->
          </A>
        </TD>
        <TD> <!-- #echo var="DIRLIST_FILE_SIZE" --> </TD>
        <TD> <!-- #echo var="DIRLIST_FILE_DATE" --> </TD>
        <TD> <!-- #echo var="DIRLIST_FILE_MIME_TYPE" --> </TD>
      </TR>
```

- ***Footer***:

```
      </TABLE>
    </BODY>
  </HTML>
```

Scripts

When a directory listing is generated through a script, the server gathers the information related to the files inside the listed directory and packs it in a special string. Then the script is executed and the special string is *POST*ed to it inside a variable called **`files`**. The server also *POST*s the directory virtual path in a variable which name is **`path`** as well as an URL encoded (escaped) version of the virtual path in a variable named **`encoded_path`**.

This special string passed in the **`files`** variable is of the form:

name$_1 \longrightarrow$	url$_1 \longrightarrow$	size$_1 \longrightarrow$	date$_1 \longrightarrow$	MIME-type$_1$[CR]
name$_2 \longrightarrow$	url$_2 \longrightarrow$	size$_2 \longrightarrow$	date$_2 \longrightarrow$	MIME-type$_2$[CR]
.	.	.	.	
.	.	.	.	
name$_{n-1} \longrightarrow$	url$_{n-1} \longrightarrow$	size$_{n-1} \longrightarrow$	date$_{n-1} \longrightarrow$	MIME-type$_{n-1}$[CR]
name$_n \longrightarrow$	url$_n \longrightarrow$	size$_n \longrightarrow$	date$_n \longrightarrow$	MIME-type$_n$

$\longrightarrow$ represents the tabulation character (ASCII code 9 - **`\t`**) and [CR] represents the carriage return character (ASCII code 13 - **`\n`**). The tuple **(name$_i$, url$_i$, size$_i$, date$_i$, MIME-type$_i$)** is the information related to the i^{th} file in the listing.

The name and the URL of every file are UTF-8 encoded. The size is in bytes. The date is conforming to the format **`YYYY-MM-DD hh:mm:ss`** which is equivalent in **`strftime`** semantics to **`%Y-%m-%d %H:%M:%S`**.

The MIME type of a directory is empty, i.e. it is equal to the empty string. The name of a directory contains always a trailing slash.

The script should split the **`files`** variable value on the carriage return characters then split each line on the tabulation character to have the information related to each file.

Example: *A very simple directory listing script (PHP version)*

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
    <TITLE>
        Index of <?php echo $_POST['path']; ?>
    </TITLE>
  </HEAD>

  <BODY>
```

```
    <TABLE BORDER="0">

      <TR>
        <TD>Name</TD>
        <TD>Size</TD>
        <TD>Date</TD>
        <TD>MIME Type</TD>
      </TR>

      <?php

        /* Split and get the lines */
        $lines = explode("\n", $_POST[’files’]);

        /* For each line do... */
        foreach ($lines as $line)
        {
          /* Split the line and get the file information */
          list($name, $url, $size, $date, $mimetype) = explode("\t", $line);

          if ($mimetype == "")
            $mimetype = "Directory";

          echo "<TR><TD><A HREF=\"$url\">" . htmlentities($name) .
                    "</A></TD><TD>$size</TD><TD>$date</TD><TD>$mimetype</TD></TR>";
        }
     ?>

     </TABLE>
   </BODY>
</HTML>
```

Example: *A very simple directory listing script (Python 3 version with the legacy cgi package)*

```
# This script makes use of the cgi package which was available by default
# until Python 3.12.
# If using Python 3.13 or later, you’ll need to install the legacy-cgi
# package from https://pypi.org/project/legacy-cgi/ or with the command:
#     python3 -m pip install legacy-cgi
# Alternatively, check the next example for a way to write an equivalent
# script without using the cgi package.

import cgi, html

posted_data = cgi.FieldStorage()

# Write the CGI header
print (’Content-Type: text/html; charset=utf-8’)
print ()

print (’<HTML><HEAD>’)
print (’<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">’)
print (’<TITLE>Index of %s</TITLE></HEAD>’ % posted_data[’path’].value)

print (’<TABLE><TR><TD>Name</TD><TD>Size</TD><TD>Date</TD><TD>MIME Type</TD></TR>’)

# Split and get the lines
lines = (posted_data[’files’].value).split(’\n’)

# for each line do...
for line in lines:
 # Split the line and get the file information
 (name, url, size, date, mimetype) = line.split(’\t’)

 if (mimetype == ”):
 mimetype = ’Directory’

 print (’<TR><TD><A HREF="%s">%s</A></TD>’ % ( url, html.escape(name) ))
 print (’<TD>%d</TD><TD>%s</TD><TD>%s</TD></TR>’ % (int(size), date, mimetype))

print (’</TABLE></BODY></HTML>’)
```

***Example:** A very simple directory listing script (Python 3 version with no legacy or deprecated packages)*

```
# Alternative version which works with the default
# wsgiref.handlers.CGIHandler module and requires
# no additional packages.

from wsgiref.handlers import CGIHandler
from urllib.parse import parse_qs
import html

def app(environ, start_response):
 # the environment variable CONTENT_LENGTH may be empty or missing
 try:
 request_body_size = int(environ.get(’CONTENT_LENGTH’, 0))
 except (ValueError):
 request_body_size = 0

 # When the method is POST, the variables are sent
 # in the HTTP request body which is passed by the WSGI server
 # in a file inside the wsgi.input environment variable.
 request_body = environ[’wsgi.input’].read(request_body_size).decode(’utf-8’)
 d = parse_qs(request_body)

 path = d.get(’path’, [”])[0] # Returns the first path value.
 files = d.get(’files’, [”])[0] # Returns the first files value.

 response_body = [ ’<HTML><HEAD>’ ]
 response_body.append(’<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">’)
 response_body.append(’<TITLE>Index of %s</TITLE></HEAD>’ % path)

 response_body.append(’<TABLE><TR><TD>Name</TD><TD>Size</TD><TD>Date</TD><TD>MIME Type</TD></TR>’)

 # Split and get the lines
 lines = files.split(’\n’)

 # for each line do...
 for line in lines:
 # Split the line and get the file information
 (name, url, size, date, mimetype) = line.split(’\t’)

 if (mimetype == ”):
 mimetype = ’Directory’

 response_body.append(’<TR><TD><A HREF="%s">%s</A></TD>’ % ( url, html.escape(name) ))
 response_body.append(’<TD>%d</TD><TD>%s</TD><TD>%s</TD></TR>’ % (int(size), date, mimetype))

 response_body.append(’</TABLE></BODY></HTML>’)

 status = ’200 OK’

 response_headers = [
 (’Content-Type’, ’text/html; charset=utf-8’)
 ]

 start_response(status, response_headers)
 return ( r.encode(’utf-8’) for r in response_body )

CGIHandler().run(app)
```

Glossary

A

Automatic Certificate Management Environment

> *ACME* is a communications protocol for automating requesting, fetching and renewing SSL/TLS certificates from certificate authorities.

C

Common Gateway Interface

> *CGI* defines how web servers interact with an external program in order to generate dynamic content. Such programs can be written in any programming language including C, Perl and PHP.

H

Host Name

> Each computer on a *TCP/IP* network has usually one or more *host names*. Networking software transparently translates them to *IP Addresses* for internal use.

HyperText Markup Language

> *HTML* is a markup language used to describe web pages.

HyperText Transfer Protocol

> *HTTP* defines how browsers and web servers communicate over a *TCP/IP* network.

I

Internet Protocol

IP is a low level networking protocol. It defines how computers can exchange messages over a network. *IP* is usually used in conjunction with *TCP*, a higher-level protocol.

IP Address

An *IP Address* is a unique identifier for a computer connected to a *TCP/IP* network. It is made of four numbers separated by dots. Each number is between 0 and 255.

S

eXtended Server Side Includes

XSSI are a set of directives that can be included in *HTML* pages and processed by the web server to create simple dynamic documents.

T

Transport Control Protocol

TCP is a high level networking protocol. It allows establishment of virtual connections between computers to send and receive data streams. *TCP* relies on *IP*.

TCP/IP

TCP/IP is a protocol suite developed by the U.S. Department of Defense for communications between computers. It has become the de facto standard for data transmission over networks, including the Internet. *TCP* and *IP* are the main protocols of *TCP/IP*.

U

Universal Resource Locator

An *URL* is the address of a document or a resource available on the Internet.

Appendix A. Command Line Parameters

Abyss Web Server command line parameters are as follows:

abyssws [-h] [-c *`configuration file`*] [-p *`console port`*] [--service-install] [--service-uninstall] [--stop] [--service]

- **`-h`**: Display the command line parameters help.
- **`-c`**: Specify an alternate configuration file. If the file does not exist, it is created and filled with the default options values.
- **`-p`**: Specify an alternate console port. It overrides the console port value in the configuration file.
- **`--service-install`**: Install Abyss Web Server as a system service.
- **`--service-uninstall`**: Remove Abyss Web Server from the system services list.
- **`--stop`**: Send a stop signal to the currently running instance of Abyss Web Server.
- **`--service`**: Run the server as a system service. Abyss Web Server is executed in the background with no user interaction and no messages or windows are displayed on the screen.

Appendix B. Startup Configuration

Select **Startup Configuration** in the **Server** menu to choose how Abyss Web Server should start up. The displayed dialog offers the following choices:

- **Manual startup**: Abyss Web Server is to be launched manually by the user (by double-clicking on its icon for example.)
- **Automatic startup on user logon**: Abyss Web Server is automatically launched when a user session is started.
- **Install as a System Service**: Abyss Web Server is installed as a Windows system service. This choice is not supported on Windows 95, 98, and Millennium Edition (ME) due to the operating system limitations. Check **Start automatically on computer startup** to have Abyss Web Server launched automatically at computer boot time without requiring having a user logged on.

Appendix C. IP Addresses and Ranges Format

IPv4 Addresses and Ranges

An IPv4 address is made of 4 numbers separated by dots as in **`aaa.bbb.ccc.ddd`**, each number ranges from 0 to 255.

An IPv4 address range must conform to one of the following forms:

- **`*`**: the range includes all valid IP addresses,
- **`aaa.*`**: the range includes the IPv4 addresses which first part is equal to **`aaa`**,
- **`aaa.bbb.*`**: the range includes the IPv4 addresses which first and second parts are respectively equal to **`aaa`** and **`bbb`**,
- **`aaa.bbb.ccc.*`**: the range includes the IPv4 addresses which first, second, and third parts are respectively equal to **`aaa`**, **`bbb`**, and **`ccc`**,
- **`aaa.bbb.ccc.ddd-iii.jjj.kkk.lll`**: the range includes the IPv4 addresses between **`aaa.bbb.ccc.ddd`** and **`iii.jjj.kkk.lll`**,
- **`aaa.bbb.ccc.ddd/n`**: the range includes all the IPv4 addresses that belong to the subnet which network prefix or CIDR (Classless Interdomain Routing) notation is **`aaa.bbb.ccc.ddd/n`**. An IPv4 address that is included in this range has its first **`n`** bits identical to the first **`n`** bits of **`aaa.bbb.ccc.ddd`**,
- **`aaa.bbb.ccc.ddd-iii.jjj.kkk.lll/n`**: An IPv4 address that is included in this range is comprised between **`aaa.bbb.ccc.ddd`** and **`iii.jjj.kkk.lll`**, and has its first **`n`** bits identical to the first **`n`** bits of **`aaa.bbb.ccc.ddd`** and **`iii.jjj.kkk.lll`**.

***Example:** CIDR Notation Example*

***`220.78.168.0/21`** represents IPv4 addresses between **`220.78.168.0`** and **`220.78.175.255`**. Actually, 220.78.168.0 in binary is 11011100 01001110 10101000 00000000. So **`220.78.168.0/21`** includes all the IPv4 addresses which 21 first bits are 11011100 01001110 10101. Therefore, it represents the range **`220.78.168.0-220.78.175.255`** (in binary 11011100 01001110 10101000 00000000 - 11011100 01001110 10101111 11111111.)*

IPv6 Addresses and Ranges

An IPv6 address is made of 8 groups of 16-bit hexadecimal values separated by colons as in **`2001:0db8:85a3:0000:0000:8a2e:0370:7334`**.

IPv6 addresses can be abbreviated by omitting leading zeroes in 16-bit values and by replacing each group of consecutive zeroes by a double colon. As such, **`2001:0db8:85a3:0000:0000:8a2e:0370:7334`** is equivalent to **`2001:db8:85a3::8a2e:370:7334`**.

An IPv6 address range must conform to one of the following forms:

- **`*`**: the range includes all valid IP addresses,
- **`ipA-ipB`**: the range includes the IPv6 addresses between **`ipA`** and **`ipB`**,
- **`ipA/n`**: the range includes all the IPv6 addresses which first **`n`** bits are identical to the first **`n`** bits of **`ipA`**,
- **`ipA-ipB/n`**: An IPv6 address that is included in this range is comprised between **`ipA`** and **`ipB`**, and has its first **`n`** bits identical to the first **`n`** bits of **`ipA`** and **`ipB`**.

Appendix D. Patterns Format

When matching a string (a sequence of characters containing an extension or a virtual path) with a pattern, the following rules apply:

- `*` matches any sequence of characters with any length (zero or more),
- `?` matches any character,
- `[set]` matches any character in the specified set,
- `[!set]` or `[^set]` matches any character not in the specified set,
- `\` suppresses the syntactic significance of a special character.

A set is made of characters or ranges. A range is formed by two characters with a `-` in the middle (as in `0-9` or `a-z`).

Preceding a special character with `\` makes it loose its syntactic significance and match that character exactly. The special characters are `[]*?!^-\`.

Example: *Examples of Patterns*

- `*`: *any string including the empty string matches with this pattern.*
- `*.*`: `list.txt` *and* `holiday.jpeg` *match while* `my-directory` *doesn't.*
- `/*.php`: `/hello.php` *and* `/mysite/scripts/test.php` *match but* `/mysite/test2.pl` *doesn't.*
- `/*/*.php`: `/mysite/test.php` *and* `/mysite/scripts/test.php` *match with this pattern while* `/test.php` *doesn't.*
- `mp?`: *Any 3 character long string that begins with* `mp` *matches.*
- `?????`: *Any 5 character long string matches.*
- `mp[2-4]`: *Only* `mp2`, `mp3`, *and* `mp4` *match.*
- `mp[!2-4]`: *All 3 character long strings that begin with* `mp` *match except* `mp2`, `mp3`, *and* `mp4`.
- `mp[2-4g]`: *Only* `mp2`, `mp3`, `mp4`, *and* `mpg` *match.*
- `mp[2\-4]file`: *Only* `mp2file`, `mp-file`, *and* `mp4file` *match.*

Appendix E. Regular Expressions

Regular expressions in Abyss Web Server conform to the PCRE syntax (Perl Compatible Regular Expressions). This appendix is a quick guide to understand the basics of regular expressions. For an extensive description of their syntax, refer to the PCREPATTERN section in **`https://pcre.org/pcre.txt`**.

When matching a string (a sequence of characters) with a regular expression, the following rules apply:

- `.` matches any character,
- `*` repeats the previous match zero or more times,
- `+` repeats the previous match one or more times,
- `?` repeats the previous match zero or one time at most,
- `{n,m}` repeats the previous match n times at least and m times at most (n and m are positive integers),
- `{n}` repeats the previous match exactly n times,
- `{n,}` repeats the previous match n times at least,
- `{,m}` repeats the previous match m times at most,
- `^` is an anchor which matches with the beginning of a string,
- `$` is an anchor which matches with the end of a string,
- `[set]` matches any character in the specified set,
- `[^set]` matches any character not in the specified set,
- `\` suppresses the syntactic significance of a special character,
- `(expression)` groups the characters between the parentheses into a single unit and captures a match for later use as a backreference (**`$1`**, ... , **`$9`**).

A set is made of characters or ranges. A range is formed by two characters with a **`-`** in the middle (as in **`0-9`** or **`a-z`**).

Preceding a special character with **`\`** makes it loose its syntactic significance and match that character exactly. Outside a set, the special characters are **`()[]{}.*+?^$\`**. Inside a set, the special characters are **`[]\-^`**.

Example: *Examples of Regular Expressions*

- ***`abc`****: any string containing the substring* ***`abc`*** *matches with this regular expression.*
- ***`abcd*`****: any string containing the substring* ***`abc`*** *followed by zero or more* ***`d`*** *characters matches with this regular expression.*

- ***`abcd?`**: any string containing the substring **`abc`** or **`abcd`** matches with this regular expression.*
- ***`ab(cd)?`**: any string containing the substring **`ab`** or **`abcd`** matches with this regular expression.*
- ***`^/dir`**: any string starting with the substring **`/dir`** matches with this regular expression.*
- ***`\.exe$`**: any string ending with the substring **`.exe`** matches with this regular expression. Note here that the character **`.`** has been escaped to remove its special meaning.*
- ***`^/dir/.*\.exe$`**: any string beginning with **`/dir`** and ending with **`.exe`** matches with this regular expression.*
- ***`^/dir/[^./]+\.exe$`**: any string starting with **`/dir`** followed by 1 or more characters except **`.`** and **`/`**, and followed by **`.exe`** matches with this regular expression.*

Example: *Examples of Backreferences*

- ***`/dir/test.exe`** matches with **`^/dir/([^./]+)\.exe$`**. This regular expression has a single group **`([^./]+)`** which defines the backreference **`$1`**. In this case, **`$1`**'s value is **`test`**.*
- ***`/dir/test.exec`** matches with **`/dir/([^./]+)\.(exe.*)$`**. This regular expression has two groups: **`([^./]+)`** which defines the backreference **`$1`**, and **`(exe.*)`** which defines the backreference **`$2`**. In this case, **`$1`**'s value is **`test`** and **`$2`**'s value is **`exec`**.*

Appendix F. Uninstalling The Software

To run the uninstaller, open the **Start** menu, choose **Programs**, then **Abyss Web Server** and select **Uninstall**. An alternative way is to open the **Add/Remove Programs** section in the Window's **Control Panel**, select **Abyss Web Server** from the dialog's list box and press **Install/Uninstall...**.

Follow the on-screen information to remove Abyss Web Server. The uninstaller may ask you if you wish to completely delete Abyss Web Server's directory. Be careful! If you confirm, it erases the server's configuration file and eventually the hosted web site.

Appendix G. Solving Ports Listening Problems

TCP/IP ports range from 1 to 65535. Abyss Web Server reports a listening problem on a port when one of the following situations is happening:

- Another instance of Abyss Web Server is already running on your computer.
- The port is already used by another application: Try stopping the other applications running on your computer to know which one is already using this port. More specifically, port 80 is usually used by other web servers. So if Abyss Web Server cannot listen to this port, there are chances that another web server is already running. Some Windows versions install by default Microsoft IIS (Internet Information Server) which uses port 80. If you do not use IIS, disable it or uninstall it to free this port.

An alternative solution would be to configure Abyss Web Server to use another web host or console port. For more information about changing the host port, refer to "Hosts Management".

If you have to change the console port, select **Settings...** in the main window, enter a new value in the **Port** field , press **OK** and confirm starting the server.

Appendix H. Troubleshooting Guide

1. The console or the web site cannot be accessed from the computer on which Abyss Web Server is running.

If you cannot browse **`127.0.0.1`**, **`::1`**, or **`localhost`**, check if you have configured a HTTP proxy in the browser. If so, disable proxy usage for **`127.0.0.1`**, **`::1`**, and **`localhost`**.

2. Abyss Web Server reports the following error at startup: "Cannot start the host XYZ (listening problem)"

This error arises when the XYZ's host port is already used by another application or if you do not have the privilege to use it. You have to choose another port for the host using the console. Refer to "Solving Ports Listening Problems" for more information.

3. Abyss Web Server reports the following error at startup: "Cannot start the console (listening problem)"

This error occurs when the console port is already used by another application or if you do not have the privilege to use it. Refer also to "Solving Ports Listening Problems" for more information.