

KerioWebStar⁵™

Technical References

© 1997-2006 Kerio Technologies. All rights reserved.

Release Date: June 15, 2006

This guide provides detailed description on *Kerio WebSTAR 5*, version 5.4. Any additional modifications and up-dates reserved.

For current product version, check <http://www.kerio.com/>.

Contents

1	Kerio WebSTAR 5 QuickStart	7
1.1	The Kerio WebSTAR 5 Architecture	7
1.2	System Requirements	8
1.3	Setting up a Web Site	9
1.4	Starting the Kerio WebSTAR 5	10
2	Configuring the Kerio WebSTAR 5 Servers	16
2.1	Kerio WebSTAR 5 Configuration Basics	16
3	Kerio WebSTAR 5 Admin Client Basics	23
3.1	Establishing Contact with Admin Server	23
3.2	Working with the Admin Client	27
3.3	The Admin Client Menubar	34
4	Basic Admin Server Settings	37
4.1	Adding Admin Users	37
4.2	Controlling Access via an Allow/Deny Table	41
4.3	Setting Access Rights for Host Admins	43
5	Creating Web Virtual Hosts	50
5.1	Adding the Virtual Host Files to your Server Machine	50
5.2	Entering and Configuring Virtual Hosts	51
5.3	Granting Access Rights to your Virtual Hosts	57
6	Web Security	58
6.1	Setting up Access Challenges	59
6.2	Setting up a Realm	66
7	Creating Web Folders with WebDAV	70
7.1	WebDAV Client Applications	70
7.2	Setting up WebDAV Service	71
7.3	Accessing the WebDAV Folder via the Internet	75
8	Kerio WebSTAR Rewrite	78
8.1	Basic Uses	78
8.2	WebSTAR Rewrite Modes	79
8.3	Implementation Details	82

8.4	Conditional Rules	84
8.5	Creating a Rule	90
9	Regular Expression Reference	92
9.1	Regular Expressions Defined	92
9.2	Wildcards	93
9.3	Character Classes	93
9.4	Repetition Characters	96
9.5	Combining Patterns	98
9.6	The Alternation Operator	99
9.7	Working with the Matched Pattern	99
10	Server-Wide and Virtual Host-Specific Settings	103
10.1	Server-Wide Settings	103
10.2	Virtual Host-Specific Settings	121
11	WebSTAR FTP Server	130
11.1	Setting Up an FTP Server	130
11.2	Creating FTP Users	133
11.3	Setting FTP Connections Options	136
11.4	Anonymous FTP	138
11.5	FTP Security	140
11.6	FTP Data Types Supported	141
11.7	FTP and Multiple Hosts	141
11.8	FTP Standards and Conventions	142
11.9	WebSTAR FTP Troubleshooting	142
12	WebSTAR Tomcat	145
12.1	Configuration	145
12.2	Quick Start	147
13	WebSTAR Flexmail	148
13.1	Configuring Flexmail	148
13.2	Flexmail Forms	149
13.3	Sending Messages with Other Methods	154
13.4	Adding Attachments to Messages	154
14	WebSTAR Search	156
14.1	Overview of WebSTAR Search	156
14.2	WebSTAR Search Basics	160
14.3	About Search Queries	163
14.4	Search Forms	164

14.5	Search Results	165
14.6	Customizing Search Results	167
14.7	Search Troubleshooting	170
15	WebSTAR Image Map	171
15.1	Using the WebSTAR Image Map Plug-In	172
16	SSL Web Security	174
16.1	About SSL Security	174
16.2	Configuring SSL	177
16.3	Obtaining and Installing Certificates	178
16.4	Creating Certificate Signing Requests	179
16.5	Using the Thawte Certificate Authority	186
16.6	Installing Your Server Certificate	192
16.7	Using SSL in Kerio WebSTAR	194
16.8	Testing Your Secure Server	197
16.9	Updating Your WebSTAR 4.x SSL Certificate	199
16.10	SSL Troubleshooting	199
17	Kerio WebSTAR Directory Indexer	202
17.1	Using Directory Indexer	204
17.2	Editing the Configuration Files	212
18	Suffix Mapping	214
18.1	File Suffixes	214
18.2	How Suffix Mapping Works	215
18.3	Examples of Suffix Mapping	215
18.4	Fields in the Suffix Mapping Table	216
18.5	Editing Suffix Mapping Table Entries	218
18.6	Suffix Mapping Example: Using SSI for All HTML Files	221
18.7	Suffix Mapping Defaults	222
19	Automatic Localization	224
19.1	Installation	224
19.2	Enabling and Configuring Localization	226
20	Using 4D Link	228
20.1	4D Link Requirements	228
20.2	Configuring 4D Link	230
20.3	Overview	232
20.4	4D Link Examples	234
20.5	Troubleshooting	257

21	Using 4D Connect	259
21.1	4D Connect Requirements	259
21.2	Installation	259
21.3	Configuring 4D Connect	261
21.4	Overview	262
21.5	4D Connect Examples	264
21.6	Troubleshooting	286
22	Domain Name Servers	289
22.1	A Name Records	289
22.2	CNAME Records	290
22.3	MX Records	290
22.4	Secondary DNS Services	290
22.5	Checking Your DNS Entries	291
22.6	Running Your Own DNS Server	291
23	Specifying Pathnames	292
	Index	293

Chapter 1

Kerio WebSTAR 5 QuickStart

This chapter covers the basics you need to know to serve your web site on the internet. It will help you move your web site to *KerioWebSTAR 5*, launch the web server, and access it for the first time via a web browser.

1.1 The Kerio WebSTAR 5 Architecture

Starting with *Kerio WebSTAR 5*, the server suite has a new design. The server applications have been completely rewritten as extremely fast Unix applications that take advantage of Mac OS X's built-in multiprocessor support, faster networking, and inherent stability. Your interface to the server suite is through a new cross-platform application, the *Admin Client*.

The *Admin Client* application is a Java application, allowing you to monitor and administer all aspects of *Kerio WebSTAR 5* from any Mac OS X or Windows computer.

When you install the server suite, it creates a *KerioWebSTAR* folder in your *Applications* folder. It contains the following major items:

- The *WebServer* folder contains the web server, the *Plug-ins* folder, and folders that contain the files for each web site that the web server serves. Many of the server's features are implemented as plug-ins; the plug-ins in the *Plug-ins* folder are loaded automatically when the server is launched.
- The *Admin Server* folder contains the *Kerio WebSTAR Admin Server* application. It manages communications among the *Admin Client* and the servers.
- The *Kerio WebSTAR Launcher* application is the double-clickable application that launches the *Admin Server* and the *Kerio WebSTAR* servers.
- The *Admin Client* folder contains copies of the *Kerio WebSTAR Admin Client* application for Mac OS X and Windows. This is the application that provides the user interface to the *Kerio WebSTAR*. It uses TCP/IP to establish a secure connection to *WebSTAR Admin Server*. You can run it from this location or copy it to any other computer on your network that is capable of running Java applications.
- The *Monitor* folder contains a copy of the *Kerio WebSTAR Monitor* application. This application opens monitor windows that allow you to monitor the usage of your web

server, number of administrative connections, and the admin and web server logs. This functionality is also available via the *Admin Client* application.

- The *Documentation* folder contains the *Kerio WebSTAR* documentation in .pdf form, release notes, last-minute *Read Me* files, and *4th Dimension* connectivity examples which enable you to publish dynamic web sites using both *4th Dimension* and *Kerio WebSTAR*.

The relationships among these applications are shown in figure 1.1.

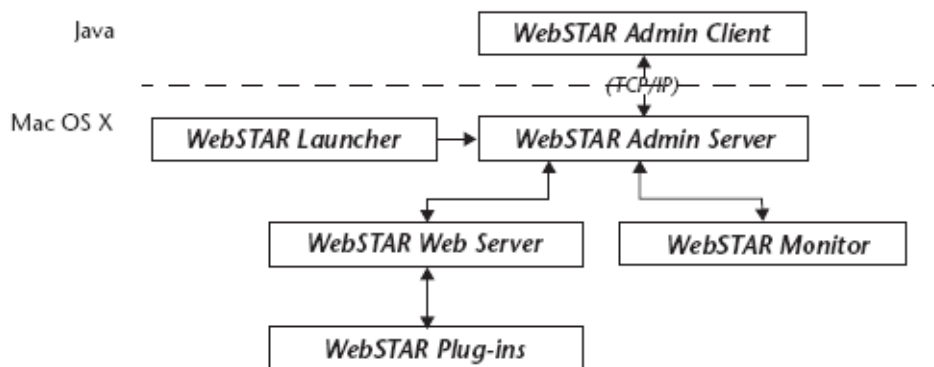


Figure 1.1 The Kerio WebSTAR 5 Server Suite Architecture

In the documentation, we refer to the Mac OS X computer that is running the *Admin Server* and the servers as the server machine and the computer that is communicating with *Admin Server* via *Admin Client* as the client machine. You can run the *Admin Client* from the server machine, but you don't need to.

1.2 System Requirements

Server Machine

The server machine must be a Macintosh computer running Mac OS X. A G4-level computer (or above) is recommended.

- Mac OS X 10.4,
- 128 MB RAM, as required by Mac OS X (256 MB or more recommended),
- 130MB free hard disk space for the server suite,
- TCP/IP connectivity (DSL or faster Internet connection recommended).

In a production environment, Make sure that your *Energy Saver* sleep settings are set to *Never*. If the computer goes to sleep and then is reactivated, you may need to quit and relaunch *Kerio WebSTAR* since networking must be reinitialized.

Client Machine

The Macintosh version of the *Admin Client* application requires Mac OS X version 10.4; the Windows version requires Windows 2000 or later.

The *Admin Client* application is written in Java; therefore it requires the Java runtime environment. This is included by default with Mac OS X; for Windows, you may need to install it yourself. It is available at:

<http://java.sun.com/products/jdk/1.2/jre/download-windows.html>

If you want to run the *Admin Client* application from another computer, simply copy the appropriate version to the other computer after installation.

1.3 Setting up a Web Site

In *Kerio WebSTAR 5*, every host (i.e., web site) is called a web virtual host, even if you are using *Kerio WebSTAR* to serve only one web site. A virtual host really isn't virtual — each host is perfectly real — the term means that you can configure one copy of *Kerio WebSTAR 5* to serve many web sites simultaneously.

Virtual hosting allows you to serve several web sites with different host names and even IP addresses. Visitors to these sites will be routed by IP address, host name, and/or the browser language setting to the correct virtual host, with its own default pages and web links.

You organize the files for each virtual host in its own folder. The convention is to place each folder in the *WebServer* folder, at the same level as the *Kerio WebSTAR* web server. Unlike *4D WebSTAR 4*, no *.html* files are placed at the same level as the web server application.

Kerio WebSTAR 5 ships with one default virtual host folder, named *DefaultSite*. It is placed in the *WebServer* folder automatically by the *Kerio WebSTAR 5 Installer*. If you are porting a *4D WebSTAR 4* site, you will need to move the *.html* files that were placed at the same level as the *4D WebSTAR 4* server to this folder. If you are using a different computer for *Kerio WebSTAR 5*, this will involve moving the files over the network to the new computer.

The *Examples* folder within the *DefaultSite* folder contains examples of some of *Kerio WebSTAR 5*'s features, including image maps, searching and web crawling, the directory indexer (which displays a doubleclickable list of the contents of the folder), CGIs, database connectivity, and FlexMail.

Within the *DefaultSite* folder, you should place your .html error files in the errors folder within the *DefaultSite* folder. Name each document using the error number, e.g., 404.html, etc. Default error files are provided.

By default, *Kerio WebSTAR 5* expects that your home page is named `index.html`, `index.htm`, or `default.html`, but you can modify this preference later.

After your web site documents have been moved to the *DefaultSite* folder, you are ready to launch the *Kerio WebSTAR 5* and begin web service!

1.4 Starting the Kerio WebSTAR 5

The *Kerio WebSTAR 5* includes a launcher utility whose only function is to start the server applications and the *Admin Server* utility. Its only interface is a small alert box that lets you know it is running.

- Double-click the *Kerio WebSTAR Launcher* application.



Figure 1.2 The Kerio WebSTAR Launcher application

When you double-click the *Launcher*, it displays a window in which it reports on the various events that occur when you launch the server suite. This report usually concludes with a list of the plug-ins that have been successfully launched. This is shown in figure 1.3.

The *Launcher's File* menu lets you open the *Admin Client* application, the *Monitor* application, open the *Mac OS X Console* application, relaunch the *Admin Server* application, or open selected directories in the *KerioWebSTAR* folder. It is shown in figure 1.4.

Use the *Relaunch Admin Server* menu item to reboot the server if it quits unexpectedly.

The *Open Directory* menu contains submenus that provide direct access to important folders within the parent *KerioWebSTAR* folder.

- The *WebServer* folder contains the actual web sever application (WSWebServer) and the folders containing the virtual hosts that you are publishing.
- The *Plug-ins* folder, which is in the *WebServer* folder, is where the plug-ins which are loaded automatically when the server starts are stored.

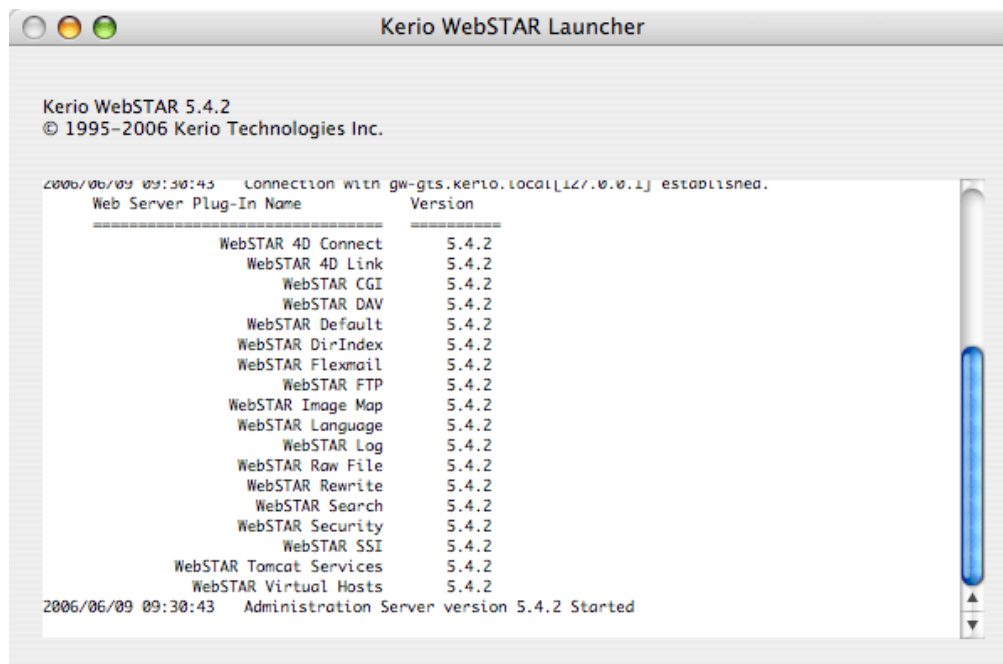


Figure 1.3 The Kerio WebSTAR Launcher Alert box

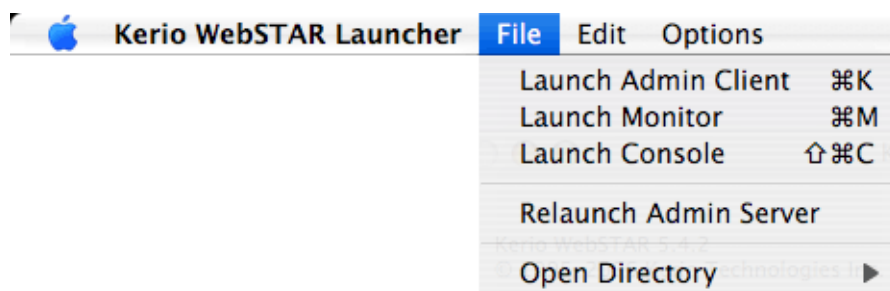


Figure 1.4 The Launcher File menu

- The *Developer Resources* folder contains tools for customizing *Kerio WebSTAR*. You can extend WebSTAR by writing your own plug-ins and provide an interface to your plug-ins by creating panels that are displayed by the *Admin Client* application — just as if your plug-ins were part of the release product.

The *Developer Resources* folder is not installed if you run the *Easy Install* option. It is a custom install option. If you did not run this option, this menu item does nothing.

- The *SSL Tools* folder contains two subfolders, *Keys* and *ServerCerts*. If you are using SSL certificates, you need to store your keys and certificates in these folders.

The *Options* menu contains two options for starting the *Monitor* application and the *Console* application when you start the *Launcher* application.

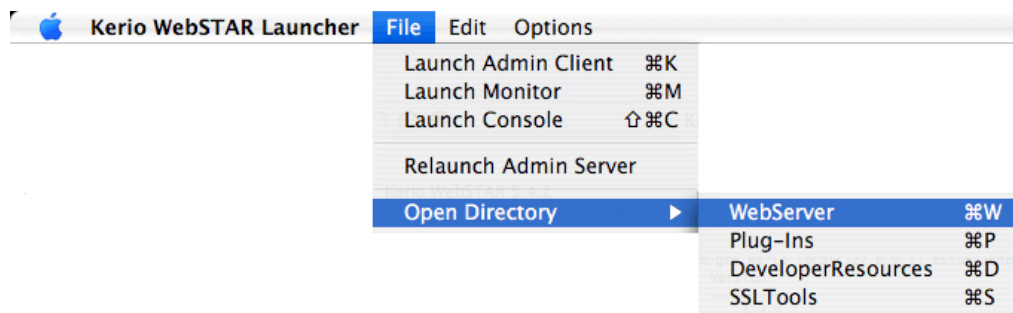


Figure 1.5 The Open Directory submenu

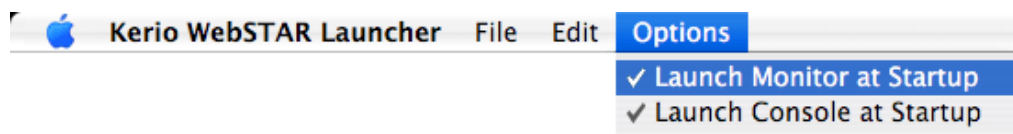


Figure 1.6 The Launcher's Options menu

The *Monitor* application is part of the *Kerio WebSTAR* and allows you to monitor web and admin activity. The *Console* application is shipped with Mac OS X and allows you to monitor the computer's activities at the (Unix) operating system level. It shows the same type of information that is shown in the *WebSTAR Launcher* window (figure 1.3), but for your Macintosh as a whole:

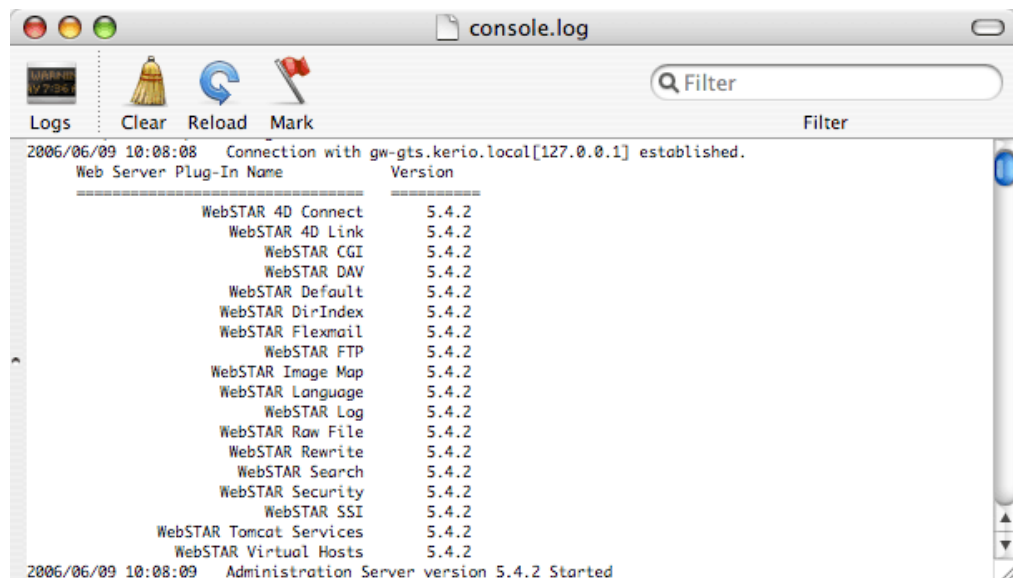


Figure 1.7 The Console application

Figure 1.7, for example, shows that *Kerio WebSTAR 5* successfully launched and loaded the plug-ins shown in the version table.

If you have problems successfully running the *Kerio WebSTAR 5*, use the *Console* application to view the initialization process. Otherwise, there is no particular reason to start the *Console* application. Keep in mind that the *Console* log keeps old information, so watch the time stamps. You may also want to set the *Console* log preferences to about 200 lines to reduce the time it takes to load.

The Process Viewer application

After you start the *Launcher* application, the *Process Viewer* listing should have items for *WSWebServer*, and two items for *WSAdminServer*. This is shown in figure 1.8. The *WSWebServer* application should be running under the special webstar user that you created during installation.

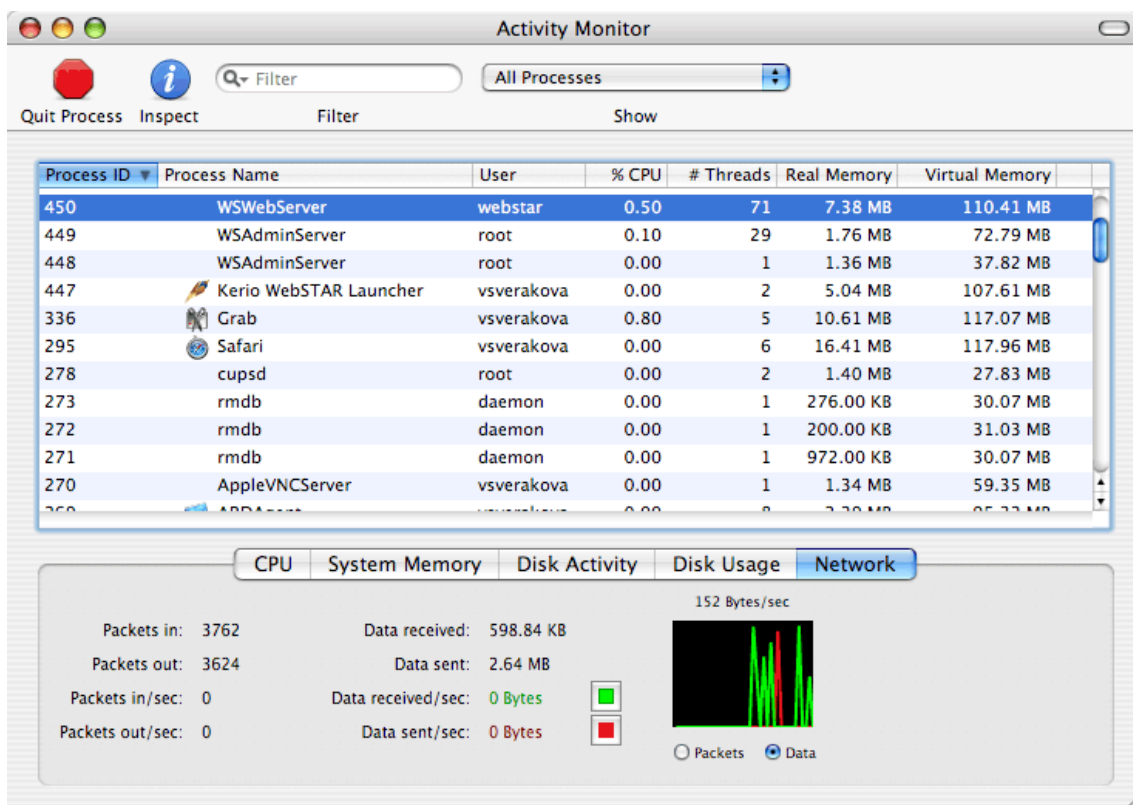


Figure 1.8 The Process Viewer shows that the Web and Mail Servers are operating normally

Note: If the CGI plug-in was installed, the *WSCGIServer* will also be running.

You may wish to reduce the default *Sample every* option within the *Process Listing* to 3 or 4 seconds to ensure that you are viewing the most recent information.

If you don't see these items, go back and verify that the *WebSTAR Launcher's* icon is in the *Dock* and it is running.

You should now be able to use any internet browser that can see the server machine to log onto the server machine and browse the web site in the *DefaultSite* folder.

If you have a domain name, enter it into the browser (for example: `http://www.domain.com`). If you have no domain name, enter the IP address of the Server Machine, e.g., `http://192.168.1.18`.

If you have not modified the default site provided by the *Kerio WebSTAR Installer*, you will see the page shown in figure [1.9](#).



Figure 1.9 The Kerio WebSTAR default site home page

Chapter 2

Configuring the Kerio WebSTAR 5 Servers

As you can see, it is easy to get your web site on the internet without doing any web server configuration whatever. You only need to install the server suite and your web site documents and launch the server.

With the *Admin Client* application, you can do many more tasks:

For example, you can:

- Enter the usernames, passwords, and access levels of the individuals who are authorized to administer *Kerio WebSTAR 5* and grant them access rights.
- Identify your web virtual host (or hosts) to the *WebSTAR* server.
- Set some basic web server parameters: the IP address(es) that the server listens to, the maximum number of simultaneous connections, the timeout, and (optionally) the SSL configuration to use.
- Set up logging and monitor visits to your site.
- Set up secure connections using SSL.
- Configure *Kerio WebSTAR 5* to serve dynamic web pages using a *4th Dimension* database as a back end.
- Set up secure realms within your site.
- Set up WebDAV service to provide secure file sharing among remote users.

2.1 Kerio WebSTAR 5 Configuration Basics

Kerio WebSTAR 5 supports two levels of administration: server-wide and host-specific. Some aspects of *Kerio WebSTAR 5* are configured at a global level and apply to the server as a whole and all web hosts, while other aspects are configured separately for each web host.

Server-Wide Settings

The following aspects of *Kerio WebSTAR 5* are configured at a global level:

- Enter and set access privileges for *Kerio WebSTAR 5* administrators,
- Create and configure web virtual hosts,
- Set the *Default Document* loaded when browsers log onto your sites,
- View installed components (a.k.a., plug-ins),
- Set the locations of the monitored servers, the ports used to communicate with *Admin Client* and the servers, and other server parameters,
- Set various web connections and web caching parameters,
- Configure encrypted connections using SSL,
- Edit the suffix mapping table,
- Set up a search index to support quick searches of your web site,
- Establish connections with *4th Dimension* databases to publish dynamic web sites,
- Set up FTP service.

Host Specific Settings

The following aspects of *Kerio WebSTAR 5* are configured separately for each web virtual host:

- Configure web security. You can create secure realms within your web site and control access via a password system and or prohibit or allow specific hosts/IP addresses to connect.
- Set up shared folders using WebDAV and set permissions.
- Monitor and log web usage
- Configure CGI actions

Levels of Administrative Access

As a result, *Kerio WebSTAR 5* supports two levels of administrative user access, the full administrator level, which grants access to the serverwide *Kerio WebSTAR 5* settings as well as the ability to administer all virtual hosts, and host admin, which grants access only to the virtual host-specific settings for selected web hosts.

The principle is this: If you use *Kerio WebSTAR 5* to serve several (unrelated) web sites, it is possible for you to designate different individuals as administrators for each web site. If you grant host admin privileges to each person and configure *Admin Server* so that each person can only see his web site, you can delegate certain administrative tasks to each person without compromising the security of the other web sites or the server as a whole.

How it Works:

When a person tries to connect to the *Admin Server* using the *Admin Client* application, the *Admin Client* firsts presents a *Connect* dialog box, in which the user must enter a valid username and password.

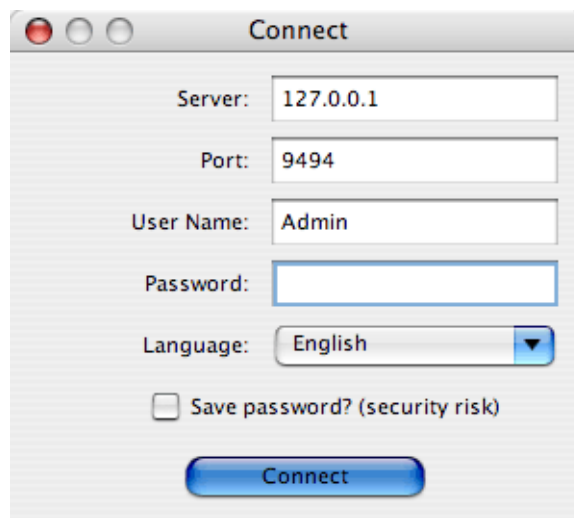


Figure 2.1 The Connect dialog box

Using the *Server* and *Port* areas, the user must specify the server machine and the port that the *Admin Server* is using to listen for requests from *Admin Clients*. By default, the *Admin Server* is listening for requests from *Admin Client* on port 9494. You can change this value once you access *Admin Server*.

He or she must then provide a username and password that has been entered into the system.

If the person enters a valid username and password, the *Admin Client* application presents a list of virtual hosts that the user is authorized to administer.

Here is an example: in figure 2.2, *Kerio WebSTAR 5* has already been configured to serve two virtual hosts. A person with full admin privileges has just entered his username and password. He can choose to access only the server-wide settings or the server-wide settings, plus the settings for either virtual host.

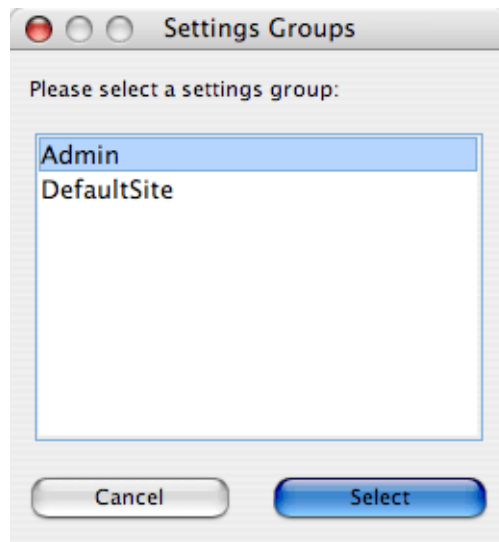


Figure 2.2 A Virtual Host dialog box when a person with Full Admin privileges logs on

In figure 2.3, a person who has been given host admin privileges to the same installation and can only administer the *DefaultSite* site has just given his username and password.

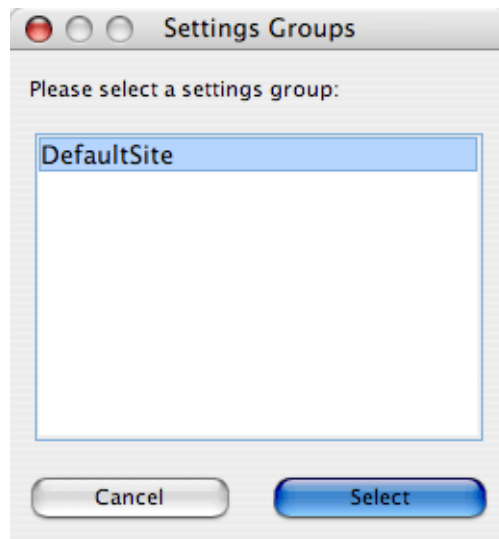


Figure 2.3 A Virtual Hosts dialog box when a person with Host Admin privileges logs on

When the person with host admin privileges logs on in figure 2.3, he will see only the host-specific settings for the *DefaultSite*. The host-specific settings for the other virtual host are unavailable. However, when the person with full admin privileges logs on in figure 2.2 and selects the *DefaultSite*, he/she sees both the server-wide and the host-specific settings for the *DefaultSite*.

The Admin Client Interface

The *Admin Client* screen is divided up into two areas: the browser area and the panel area.

The browser area allows you to view a panel in the panel area by clicking it. Panels are organized into topics and you can collapse or expand a topic by clicking its disclosure triangle. You click on a panel name to display it in the panel area.

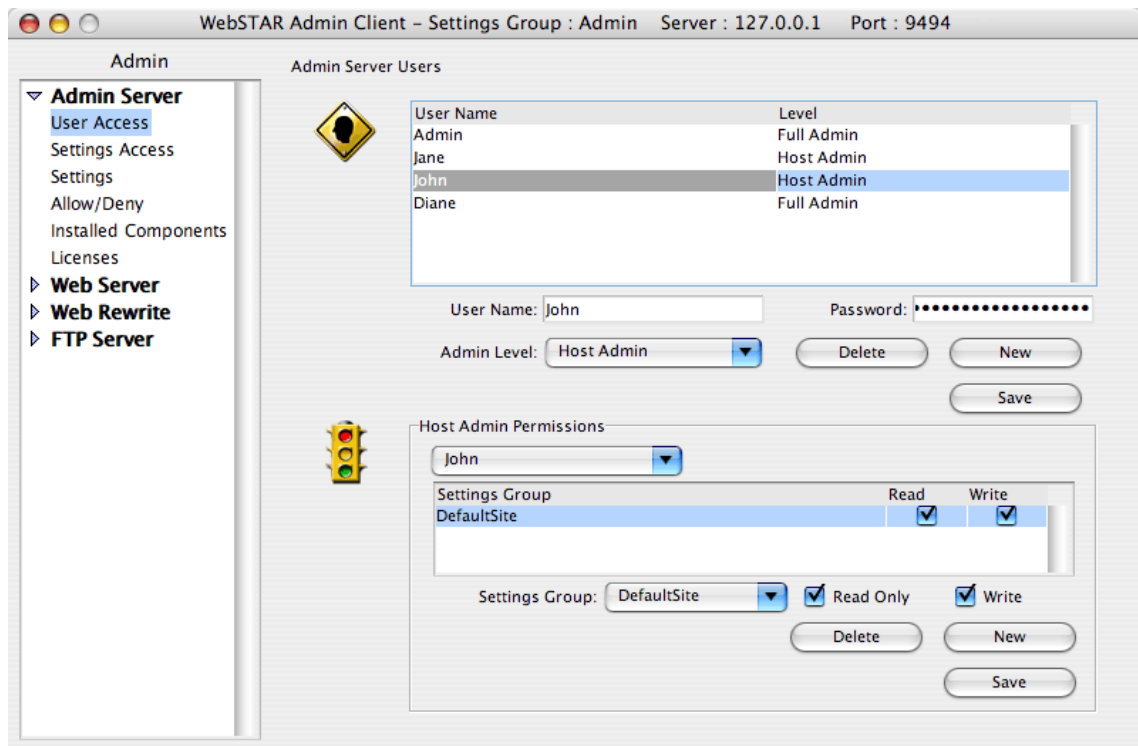


Figure 2.4 The Browser and Panel areas

If you access the *Admin* settings group, only the server-wide topics are listed in the browser area. At this point, only four web server topics are available, *Admin Server*, *Web Server*, *Web Rewrite*, and *FTP Server*.

When you log back in as a full admin to access any virtual host, the browser area will display the additional topics and panels that are virtual-host specific as well as the panels for the server-wide settings.

When you create a new admin user and give that person host admin privileges, the browser area will show only the host-specific panels that person has access rights for — omitting the server-wide panels and hiding the host admin settings for all other virtual hosts served by your copy of *Kerio WebSTAR 5*.

If you create several virtual hosts and log in as yourself (i.e., full admin privileges), you will be asked to choose the virtual host you wish to administer. The browser area will then show the server-wide panels and the virtual-host specific panels for only the virtual host you chose, not other virtual hosts for which you have access rights. You can open another window to gain access to other virtual hosts.

To summarize:

- If you're a full administrator and you want to access only server-wide settings, choose the *Admin* settings group in the *Settings Groups* dialog.
- If you're a full administrator and you want to access settings for a we virtual host, choose that virtual host in the *Settings Groups* dialog.
- If you're a full administrator and you want to access settings for two or more virtual hosts, choose one of the settings groups in the *Settings Groups* dialog and then choose *File* → *New* and choose another virtual host. The second group of settings are then accessible in the new window.

Note: This feature also gives you the ability to open another copy of the same virtual host in a new window. This is discouraged for the following reason. If you make a change to a setting in one copy of the virtual host's settings, those changes aren't reflected in the other copy automatically. You need to remember to choose *Refresh* from the *View* menu when you switch to the second instance of the virtual host to update that display. If you aren't careful, you could easily get confused.

- If you're a host administrator, choose the virtual host you want to access in the *Settings Groups* dialog.

The name of the current settings group appears just above the browser list and in the title bar of the *Admin Client* window.

The browser panels for each of these possibilities are illustrated in figure 2.5.

As you can see, the system allows you to hide all the server-wide settings and irrelevant virtual hosts from host admins.

Chapter 10 provides brief overviews of each of the panels in *Admin Client*. Later chapters in the reference manual provide detailed information.

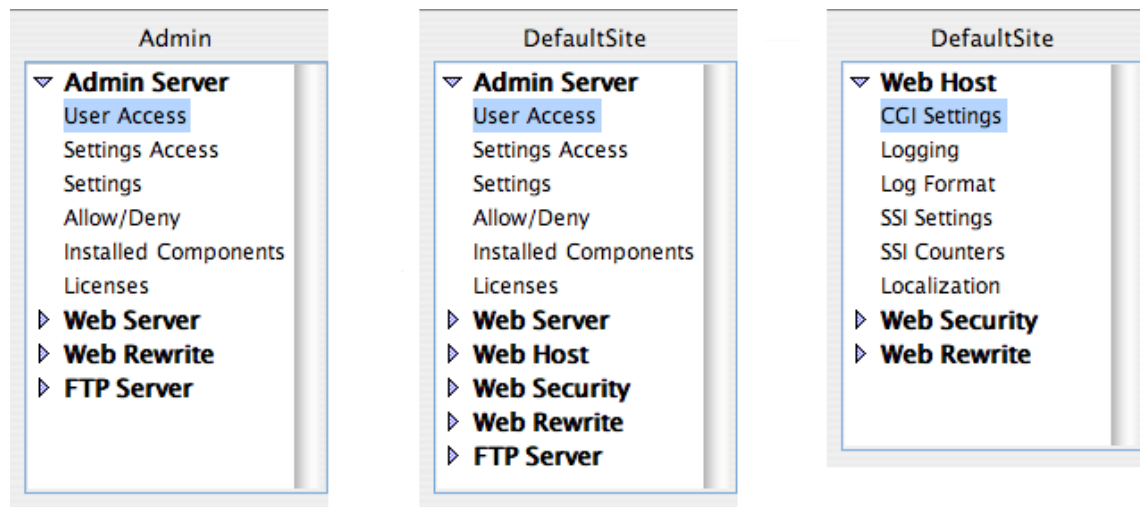


Figure 2.5 Browser panels for full admin logged into Admin, a full admin logged onto a web virtual host settings group, and for a host admin logged into a virtual host settings group

Chapter 3

Kerio WebSTAR 5 Admin Client Basics

This chapter covers the basic operating principles of the *Admin Client* application. It covers:

- Establishing contact with *Admin Server*,
- Using the browser area,
- Entering and modifying settings using the panel area.

3.1 Establishing Contact with Admin Server

To administer the *Kerio WebSTAR* from another computer, you need TCP/IP access from the client to the server machine. The default port used by the *Admin Client* and the *Admin Server* is 9494. If necessary, make sure that a firewall allows access to this port.

Note: For information about the initial contact with *Admin Server*, please see the *Installation Guide*.

To configure your virtual host:

1. Launch the *Admin Client* application.

You can do this by:

- Choosing *File* → *Launch Admin Client* from the *Launcher* application.
- Double-clicking the *Admin Client* application icon in the *Admin Client* folder in your *KerioWebSTAR* folder. In a few seconds, the *Connect* dialog box appears (figure 3.1)

2. Enter the IP address or hostname of the computer that is running the *Kerio WebSTAR* 5 in the *Server* entry area.

3. Enter 9494 in the *Port* area.

This is the default port used by *Admin Client* to communicate with the *Admin Server* application. This port be changed at a later time using *Admin Client*.

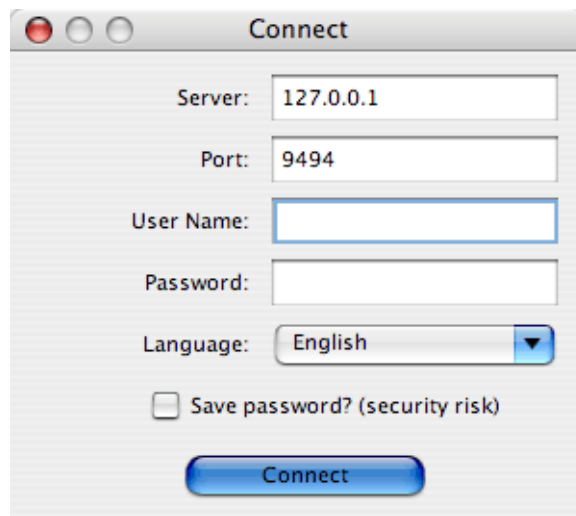


Figure 3.1 The Admin Client Connect dialog box

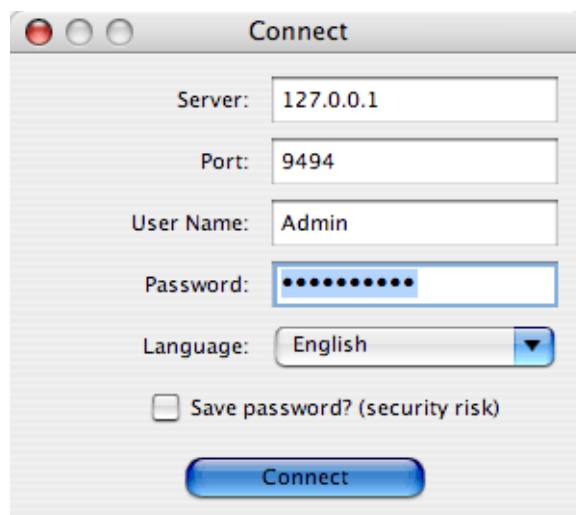


Figure 3.2 A filled-in Connect dialog box

When you first connect to *Admin Server*, the system uses the username and password as the master ID. You should choose a nonsense string as a password so that it is very secure.

4. Choose a *User Name* and *Password* and enter them in the dialog. Click *Connect* (see figure 3.2).

The *Admin Client* attempts to connect to the server machine. In a few seconds, you will see the *Settings Groups* dialog (see figure 3.3).

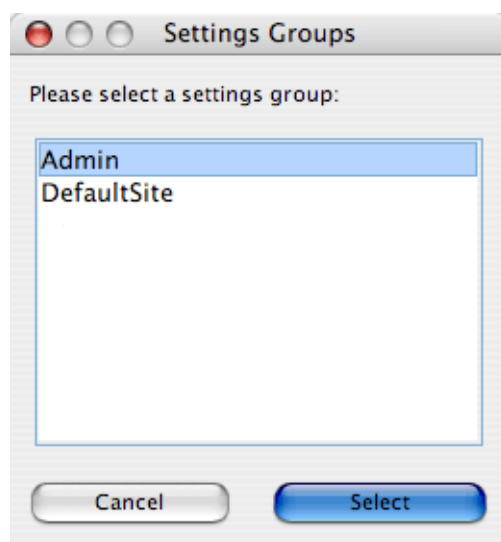


Figure 3.3 The Settings Groups dialog box

When you first log in, the *Admin* and *DefaultSite* settings groups are listed.

The *Admin* setting doesn't grant access to a Web Virtual Host; it gives you access to the server-wide settings only. The *DefaultSite* settings group gives you access to the default web site that the *Kerio WebSTAR 5 Installer* has placed on in the *KerioWebSTAR* folder.

Any additional web virtual hosts you create using the *Admin Client* application will appear in this list in subsequent logins. In future connections, you choose the settings group that you want to administer.

5. Highlight *DefaultSite* and click *Select*.

The *Admin Client* application opens a window that provides access to the *Admin* (Server-wide) panels.

You access a panel by clicking its name.

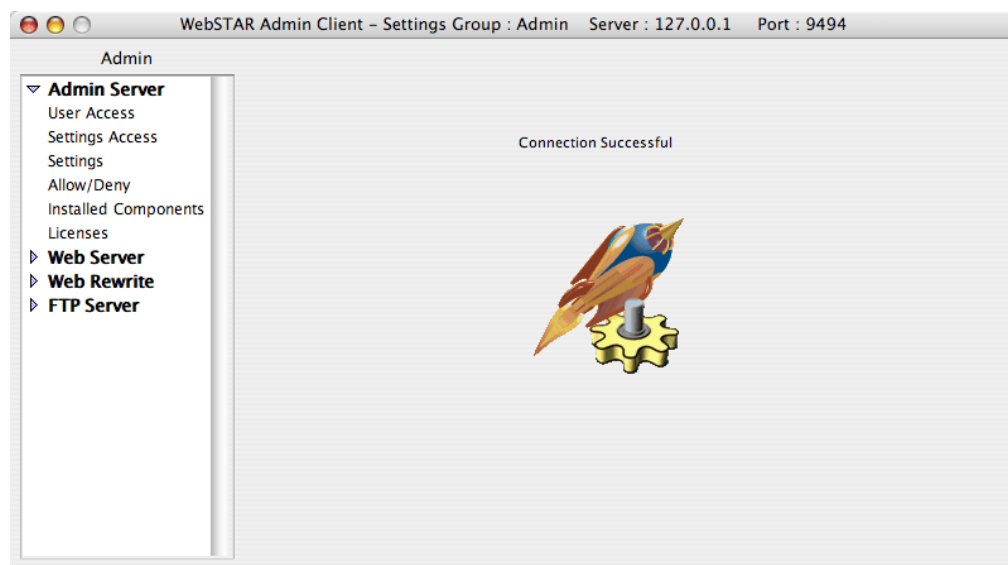


Figure 3.4 The Kerio WebSTAR 5 Admin Client window

3.2 Working with the Admin Client

The *Admin Client* window uses a browser area on the left to allow you to display different panels and a panel area in which you view and configure *Kerio WebSTAR 5*. The title bar shows the settings group to which the settings apply, the IP address of the machine on which the *Admin Server* is running, and the port used to communicate with *Admin Client*. The name of the settings group also appears just above the browser list.

This information is shown in figure 3.5.

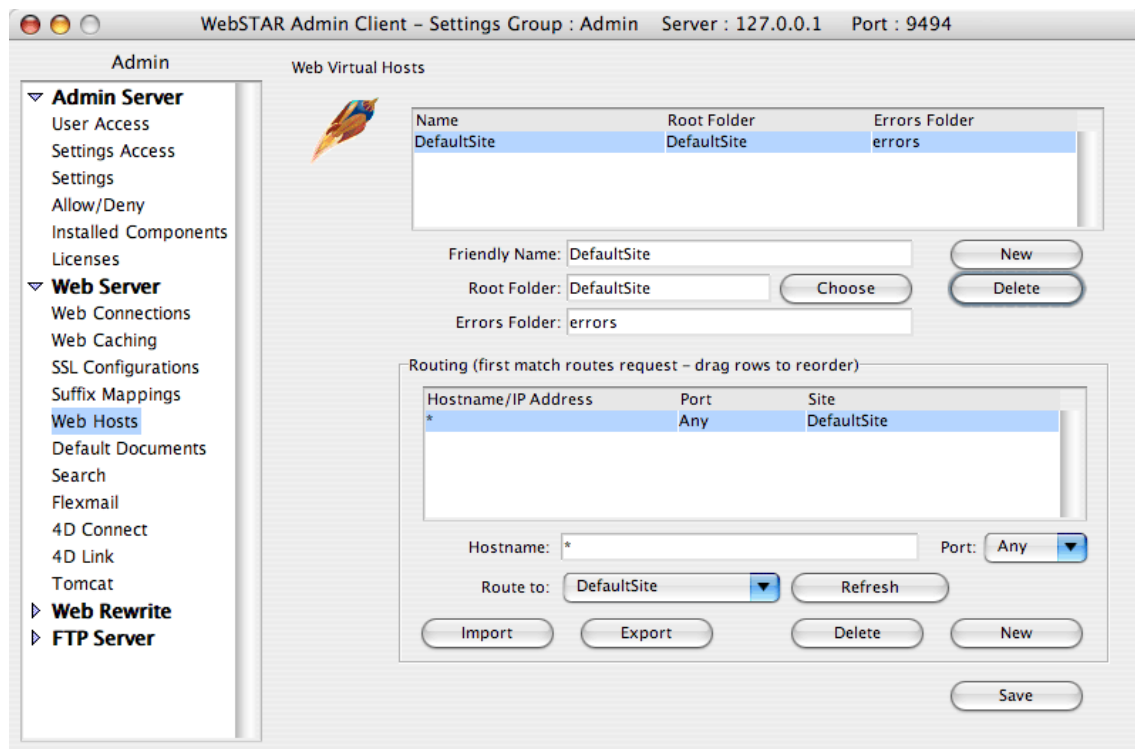


Figure 3.5 The Admin Client window title bar showing the current Settings Group

In figure 3.5, the title bar says that the settings group is *DefaultSite*, since that is how the user logged in (figure 3.3).

Working with the Browser area

The browser area organizes panel names by topic. The *Admin Server* and *Web Server* groups contain the server-wide settings and are shown only if you have full admin privileges. The *Web Host* and *Web Security* groups are web virtual host-specific. The settings in these groups of panels pertain to the web host shown in the title bar and above the browser.

For example, in figure 3.6 the user has logged onto a the *DefaultSite* settings group. The *Web Host* and *Web Security* groups are now shown and the panels pertain to this *Virtual Host* only.

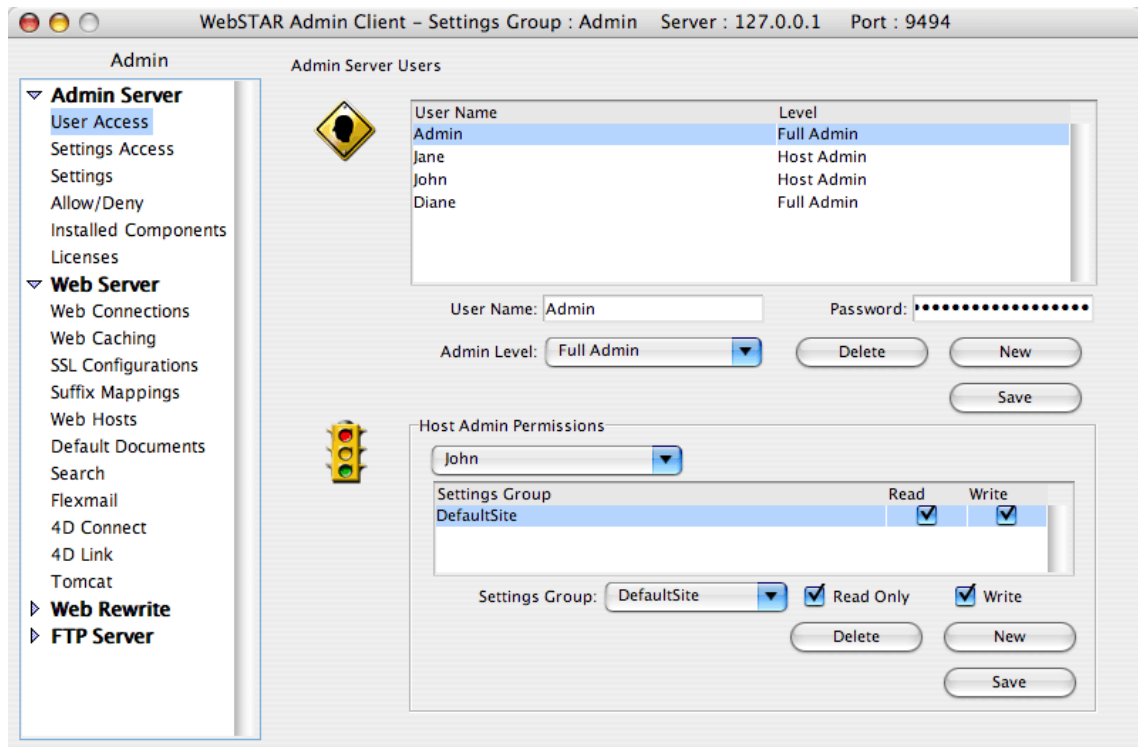


Figure 3.6 The Admin Client window when the user has logged onto the DefaultSite Settings Group

You display a panel by clicking on its name in the browser; you can hide the panel names in a group by clicking the group's disclosure triangle.

Working with the Panel area

Although each panel allows you to manage different settings, they all use the same interface conventions.

Tables

Many panels allow you to create a new entry in a table. For example, the list of web virtual hosts in figure 3.7 appears in a table.

You follow the same basic procedure for entry and editing of all tables.

- *To create a new entry* — Click the *New* button associated with the table. (Some panels contain two tables, each with their own set of buttons.) This creates a new, row in the table. The new row may be blank or filled in with placeholder values that you replace with values for your web site.

Name	Root Folder	Errors Folder
DefaultSite	DefaultSite	errors

Friendly Name:

Root Folder:

Errors Folder:

Figure 3.7 The web virtual hosts table in the Web Hosts panel

You can enter information by typing directly into the new row. Often there are separate entry areas below the table in which you can also enter information. There is one area per column in the table. When you type into one area, your text is mirrored in the other area.

Name	Root Folder	Errors Folder
DefaultSite	DefaultSite	errors
Compass	DefaultSite	errors

Friendly Name:

Root Folder:

Errors Folder:

Figure 3.8 Entry into the table cell is mirrored in the entry area (and vice versa)

Add as many rows as needed by clicking the *New* button to add each new row. Click *Save* to save all your changes to the table. Some tables have their own save button; other panels have only one *Save* button, located at the bottom of the screen.

- *To select a folder* — Some tables require that you select a folder on your hard disk. This is true of the Virtual Hosts table shown in figure 3.8. You can do so in two ways: you can enter the path to the folder by typing directly into the *Root Folder* field or you can click the *Choose* button next to the field and select the folder from a *Select Folder* dialog box.

In figure 3.8 you must select the folder that contains all the files for the web virtual host that you are entering. This can be done by clicking the *Choose* button to display the dialog box shown below:

When you navigate to the desired folder, highlight it and click *Choose*. When you return to the *Admin Client* panel, the path to the folder you chose is automatically filled into the field adjacent to the *Choose* button.

- *To modify an entry* — Double-click in a table cell to get an insertion point. Type

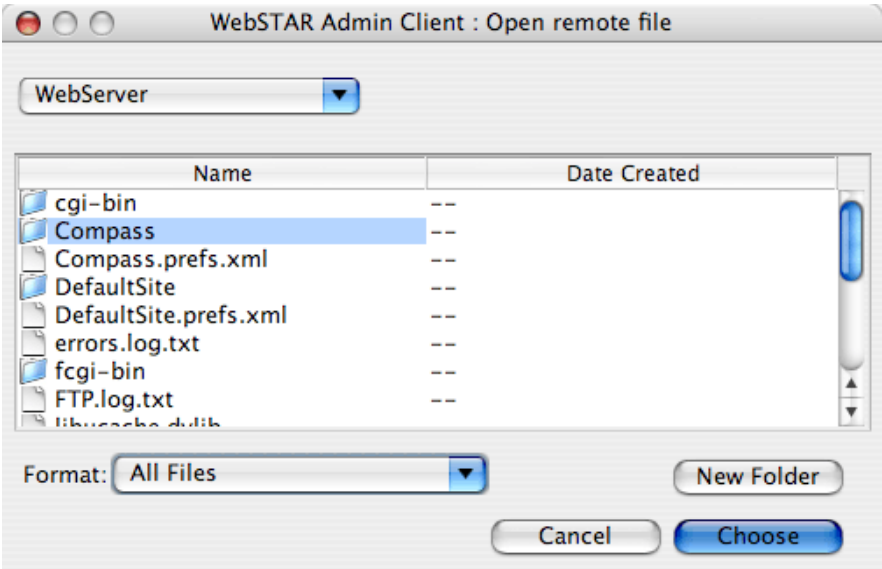


Figure 3.9 A Select File dialog box with a folder in the WebServer folder selected

the new text in the cell or modify the contents of the corresponding entry area below the table. Click the *Save* button to save your modifications to the table.

- *To delete an entry* — Click in a row to select the row. Click the *Delete* button to remove the row. If you change your mind, choose *Refresh* from the *View* menu to revert back to the saved settings. After you click *Save* to commit your deletions to the server, deleting a row is not undoable.

Table Pop-up Menus

Some tables use pop-up menus to control the contents of the table. For example, the contents of the table in figure 3.10 is controlled by the pop-up menu just above it.

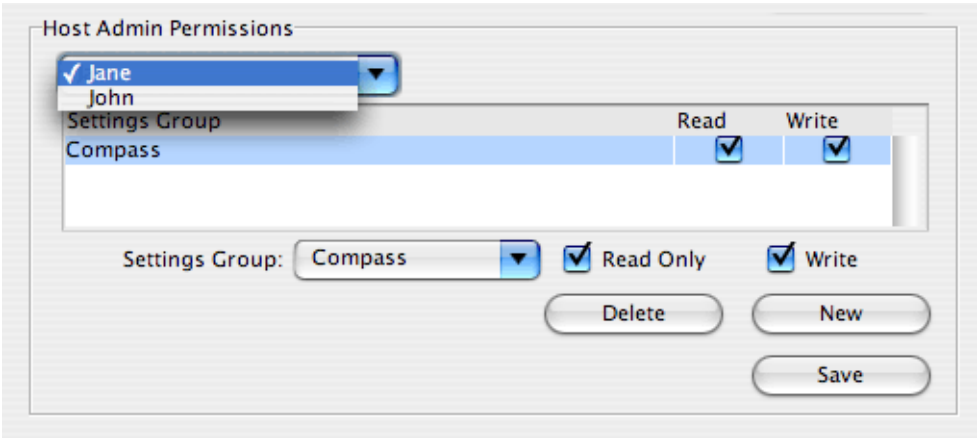


Figure 3.10 Host Admin permissions for user Jane

Figure 3.10 is taken from the *Admin User Access* screen. A user, Juli, has already been given host admin privileges. In figure 3.11, taken from the same panel, another host admin has been selected from the pop-up menu. The table then changes to show her host admin permissions.

Host Admin Permissions

John

Settings Group	Read	Write
DefaultSite	☒	☒

Settings Group: DefaultSite ☒ Read Only ☒ Write

Delete New Save

Figure 3.11 Host Admin permissions for user John

You add, modify, and delete records within the table using the same process described earlier.

Resizing Table Columns

You can resize the columns of a table to make individual columns more readable. Move the pointer into the heading area, just above a column divider. The pointer will turn into a resizing cursor. When it does, drag to the left or right to resize the column.

This is illustrated figure 3.12.

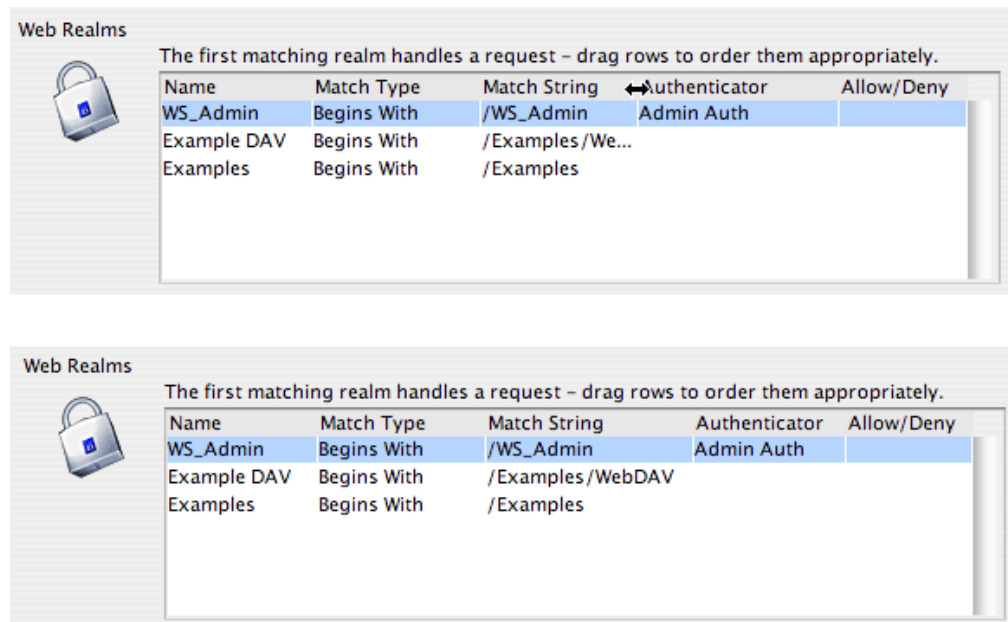


Figure 3.12 Resizing several columns to bring the last columns into view

Sorting Tables

You can sort the rows of a table by a column. Click in the header area to sort by that column. An up or down arrow in the sort column indicates whether sorting is in ascending or descending order.

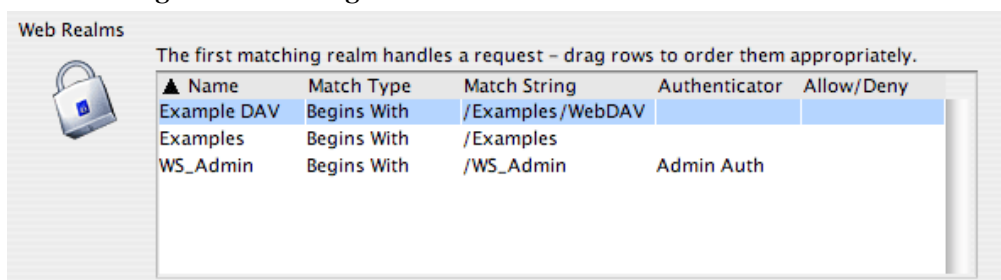


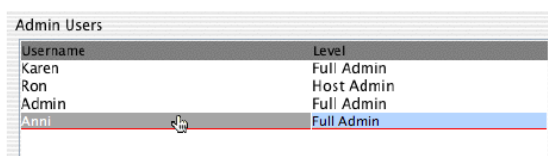
Figure 3.13 A sorted column in a table

Reordering Table Rows

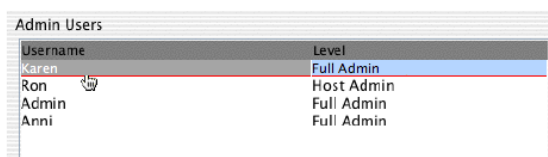
Many tables allow you to drag rows up and down to reorder the rows in the table. Sometimes the order of the rows has no meaning, but in some tables, *WebSTAR* evaluates rows in the order that they appear in the table, from top to bottom.

To reorder rows, click on a row to be moved and drag it up or down. When you begin to drag, the pointer becomes a pointer cursor and a red line between rows indicates where the row would be placed if you choose to drop the row at that moment.

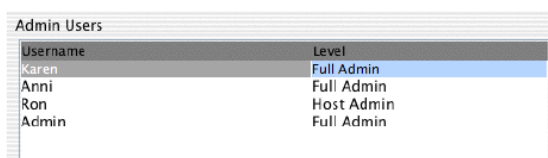
The following illustration shows a row being moved.



The User "Anni" is being moved between Karen and Ron.



The red line indicates that the user has moved the record to the correct position.



The reordered rows after the drop.

Figure 3.14 Reordering rows by dragging

Import and Export

Some tables have import and export functionality. Clicking the *Export* button exports the contents of the table as a tab-delimited text file. The *Import* button imports such a text file into the table. When you import a text file, the imported file replaces any existing rows in the table.

3.3 The Admin Client Menubar

The *Admin Client* application has five menus:

- *File* contains the *Connect*, *New*, and *Disconnect* menu items.
- *View* contains the *Refresh* menu item.
- *Admin Server* contains the *Shutdown* menu item.
- *Windows* contains menu items for the *Admin Client* monitoring windows.
- *Tools* contains the *Generate Keys* and *CSRs* menu item, used to generate SSL certificates. For information on this option, see the section [16.4](#).

The File Menu

The *File* menu allows you to do the following:

- *Connect* — the *Connect* menu item is enabled only when you have an open *Admin Client* window that is not connected to *Admin Server*. Such a window says *Not Connected* in its title bar. Choose *Connect* to display the *Connect* dialog box. Repeat the login process described in the section [3.1](#).
- *New* — Allows you to open another window. When you choose *New*, the *Settings Group* dialog box appears. You can repeat the process described in the section [3.1](#).
- *Switch Settings Group* — Allows you to access a different settings group (provided you have such access privileges). When you choose *Switch Settings Group*, the *Settings Group* dialog appears (figure [3.3](#)). When you choose another settings group, it replaces the current settings in the *Admin Client* window. Another way to access a second settings group is to use the *New* command to open a second *Admin Client* window with the second group.
- *Disconnect* — Allows you to end the connection with *Admin Server*.
- *Most Recent Connections* — Below the fixed menu items is a group of temporary menu items for the most recent connections. Selecting one of these menu items reconnects

to *Admin Server* using the username and password for that menu item. Use these menu items to bypass the process of re-entering a username and password to re-establish a connection.

The View Menu

The *View* menu has one item:

- *Refresh* requests the most recent values of the parameters displayed in the current panel and redraws the current panel.

The Admin Server Menu

The *Admin Server* menu has one item:

- *Shut down* shuts down the server suite.

The Windows Menu

The *Windows* menu contains menu items that open the monitoring windows that are also available from the *Kerio WebSTAR Monitor* application on the server machine.

- *Admin Activity* Number of Admin user connections,
- *Admin Log* the admin log,
- *Web Activity* number of hits, the current, the high, and the number of denials.
- *Web Log* the web usage log,
- *FTP Log* the FTP server usage log,
- *Web Activity* web connection activity.

The Tools Menu

The *Tools* menu has four menu items that open dialog boxes that contain tables that are also available on panels within the *Admin Client* application (*Sites*, *Routes*, and *Built-in User Lists*). The *Generate Keys and CSRs* menu command opens a dialog box that you need to use as part of the process of implementing encrypted communications via SSL.

- *Web Sites* — Opens a dialog box that shows the web virtual hosts table, listing the virtual hosts being served by *WebSTAR*. This table also appears in the *Virtual Hosts*

panel. You can create, edit, or delete virtual hosts from this dialog box. For more information, see the section [5.2](#).

- *Web Routes* — Opens a dialog box that shows the *Routing* table, which shows how incoming requests are routed to your virtual hosts. This table also appears in the *Virtual Hosts* panel. You modify the *Routing* table from this dialog. For more information, see the section [5.2](#).
- *Generate Keys and CSRs* — Used to generate a private key and a certificate to submit to a certificate authority. This is relevant only if you use SSL. For more information about this dialog box, see the section [16.4](#)
- *Web Host Users* — Opens a dialog box that shows the *Built-in User Lists* table. This list is also displayed on the *Built-in User Lists* panel in the *Web Security* group. For more information, see the section [6.1](#).

Chapter 4

Basic Admin Server Settings

After you get your web site up and running, you will want to add your name to the list of persons who are authorized to administer *Kerio WebSTAR 5* via the *Admin Client* application. If other people are also permitted to perform administrative tasks, their names should also be entered.

Within *Admin Client*, you are also able to limit access to *Admin Server* by creating a list of machines that are permitted to connect to *Admin Server*.

4.1 Adding Admin Users

By default, one user has access to *Admin Server* — *Admin*. You should add your name to the list of admin users and give yourself full admin privileges.

Note: For information about the initial contact with *Admin Server*, please see the *Installation Guide*.

To enter an admin user:

1. If it is not already running, start the *Admin Client* application — either by choosing *Launch Admin Client* from the *Launcher's File* menu or by double-clicking the *Admin Client* application (if *Admin Client* is already running, skip to step 6, below.).

In a few seconds, the *Connect* dialog box appears:

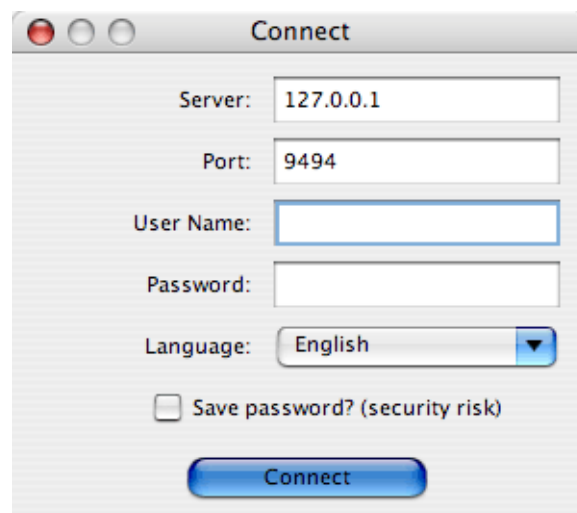


Figure 4.1 The Admin Client Connect dialog box

2. Enter the IP address or hostname of the computer that is running the *Kerio WebSTAR 5* in the *Server* entry area.

3. Enter 9494 in the *Port* area.

This is the default port used by *Admin Client* to communicate with the *Admin Server* application. This port can be changed at a later time using *Admin Client*.

4. Enter the *User Name* and *Password* that you used when you initially connected to *Admin Client* during the installation process.

The *Admin Client* attempts to connect to the server machine. In a few seconds, you will see the *Settings Groups* dialog:

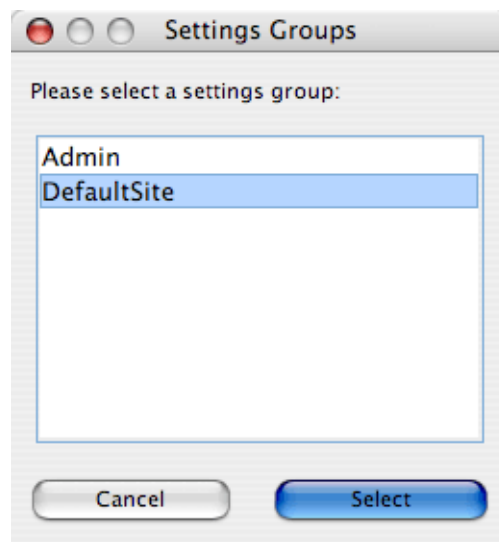


Figure 4.2 The default Settings Groups dialog box

When you first log in, only the *Admin* and *DefaultSite* settings groups are listed. Only the *DefaultSite* group corresponds to a real virtual host in your *WebServer* folder. The *Admin* settings group allows a full admin to administer the server-wide settings only.

The virtual hosts you create using the *Admin Client* application will appear in this list in subsequent logins. In future connections, you choose the virtual host that you want to administer.

5. Highlight *Admin* and click *Select*.

The *Admin Client* application opens a window that provides access to the *Admin* (Server-wide) panels.

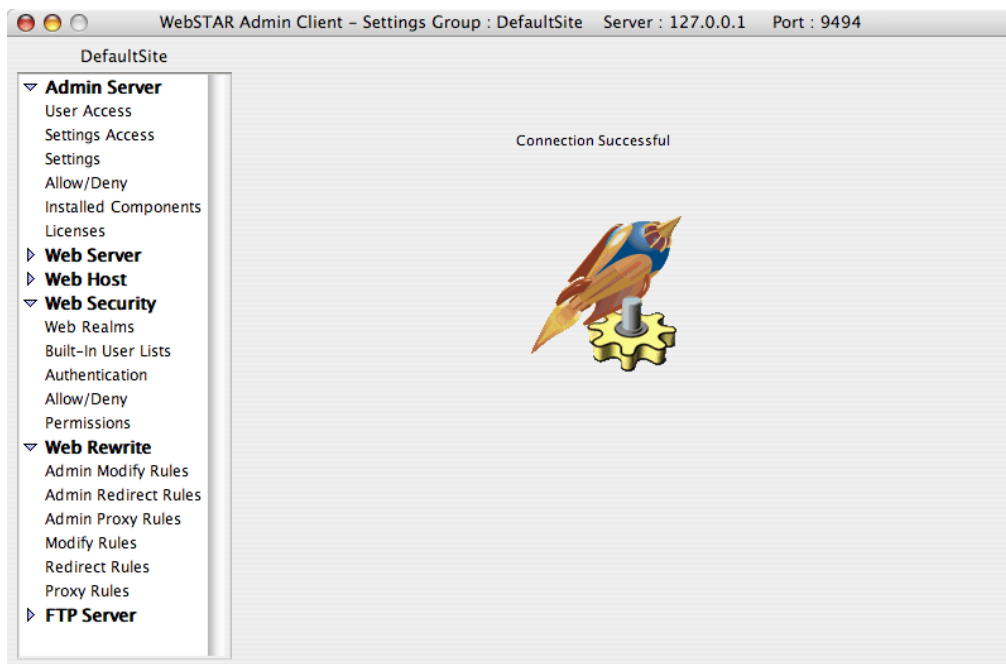


Figure 4.3 Main panel

6. Click *User Access* in the *Admin Server* group in the browser area.

The *User Access* panel appears, showing only the default user, *Admin*.

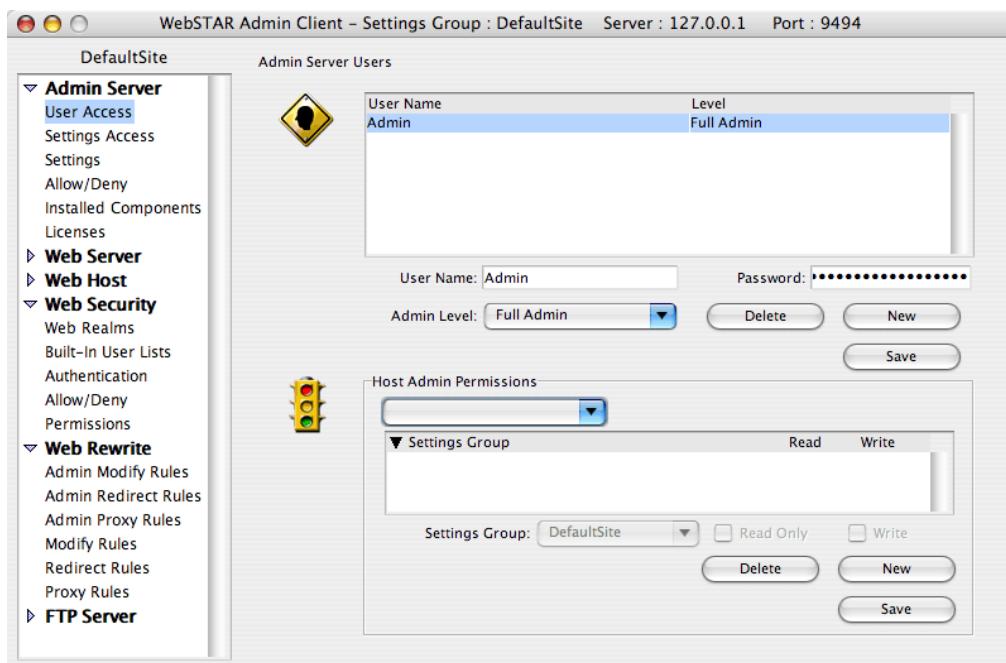


Figure 4.4 Admin Server → User Access

7. Click *New* in the *Admin users* area to add a new user.
A blank row in the *Admin Users* table appears.
8. Enter your username and password either in the new row or in the two entry areas below the table.
9. Use the *Admin Level* pop-up menu to assign yourself full admin access privileges.
This allows you to configure both the server-wide and host-specific *Kerio WebSTAR 5* settings.
10. (Optional) If you want to give other users full admin access, repeat steps 7 to 9.

Adding Host Admin Users

After you create your virtual hosts (see chapter 5, you may need to return to this panel to create users who have only host admin privileges. If you (or the other full admins) plan to administer all the virtual hosts, you don't need to create separate host admin entries; you are automatically granted access to all virtual hosts.

If you do need to add host admin users, follow this process after first adding the virtual hosts to the system:

To add a host admin user:

1. Click *New* in the *Admin users* area to add a new user.
A blank row in the *Admin Users* table appears.
2. Enter the username and password of the host admin in either in the new row or in the two entry areas below the table.
3. Use the *Admin Level* pop-up menu to assign him/her host admin access privileges.
4. Repeat these steps to add all the host admin users.

In the next series of steps, you give each host admin user access rights to selected virtual hosts.

1. In the *Host Admin Permissions* area, choose a *Host Admin User* from the pop-up menu above the table.
2. Click *New* in the *Host Admin Permissions* area.
A new row is added to the *Host Admin Permissions* table.

3. Choose a virtual host from the *Virtual Host* pop-up menu below the table.
4. Click the *Read Only* or *Read/Write* checkboxes, as appropriate.
5. Repeat steps 3 and 4 for each virtual host that this host admin has access rights to.
6. (Optional) If there are more host admin users, repeat this process for each host admin user.
7. Click *Save* to save the changes to the server.

4.2 Controlling Access via an Allow/Deny Table

By default, the admin users you enter in the *Admin User Access* panel can connect to *Admin Server* from any suitable computer that runs Java applications. Security is controlled by the username/password settings and (if applicable) full/host admin access rights.

You can also restrict access by hostname or even IP address of individual machines. You do this by entering an *Allow/Deny* table that specifies the hostnames and/or IP addresses of connections that either are or are not authorized to administer *Kerio WebSTAR 5*.

To control admin access with an *Allow/Deny* table:

1. Click the *Allow/Deny* topic in the *Admin Server* group.

The following panel appears (figure 4.5).

2. Click *New* to add a new row to the table.

The *Allow/Deny* column is filled with a pop-up menu that lets you specify either *Allow* or *Deny* for the entry.

Chapter 4 Basic Admin Server Settings

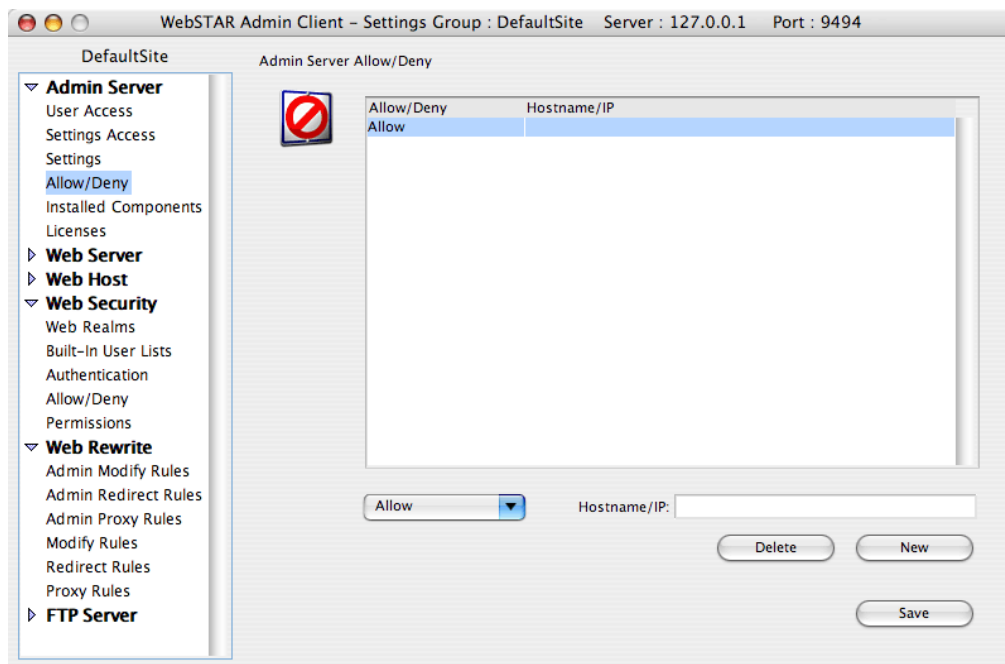


Figure 4.5 The Admin Server Allow/Deny table

3. Choose *Allow* or *Deny* from either the pop-up menu in the row in the table or the pop-up menu below the table.
4. Enter either a hostname or the IP address in the second column.

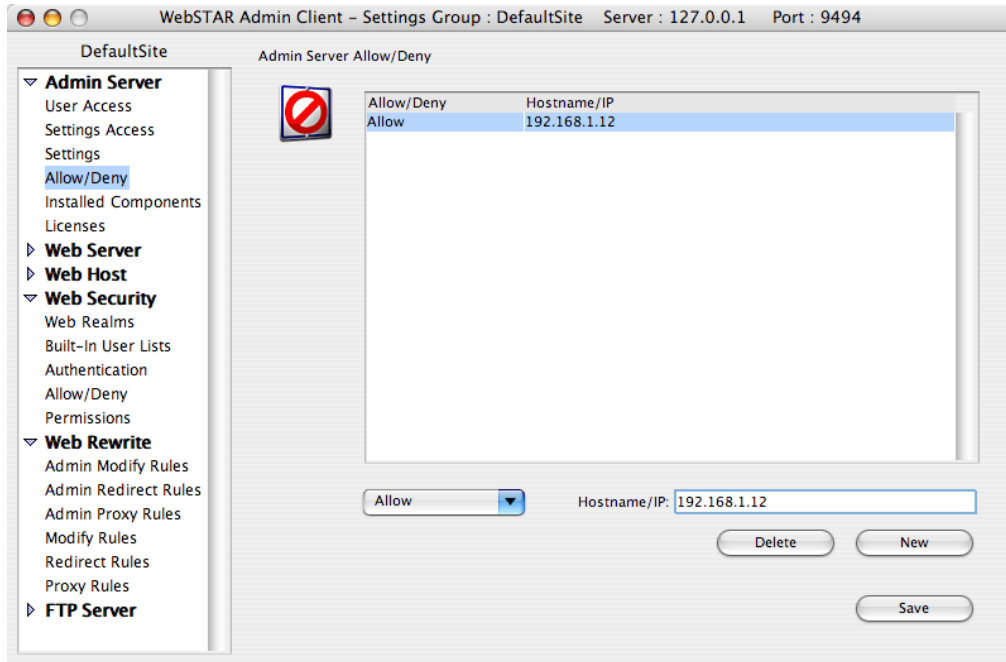


Figure 4.6 An Allow/Deny table that accepts Admin connections from only one computer

If you enter one *Allow* hostname/IP address, all others are automatically denied. If you enter a *Deny* hostname/IP address, all others are allowed.

5. (Optional) Repeat the process to add additional rows.
6. Click *Save* to save the *Allow/Deny* table.

Now, only the *Admin Users* must use the specified hostnames or IP addresses to access *Admin Server*.

4.3 Setting Access Rights for Host Admins

In addition to limiting the ability of host admins to “see” other virtual hosts on your server, you can also limit their ability to administer certain aspects of their own virtual hosts. This is done using the *Settings Access* panel in the *Admin Server* group.

This panel lists features of *Kerio WebSTAR 5* that are implemented via plug-ins. The *Read* and *Write* columns allow a full admin to limit the degree to which host admins can view and/or modify these features.

The following is the *Settings Access* panel:

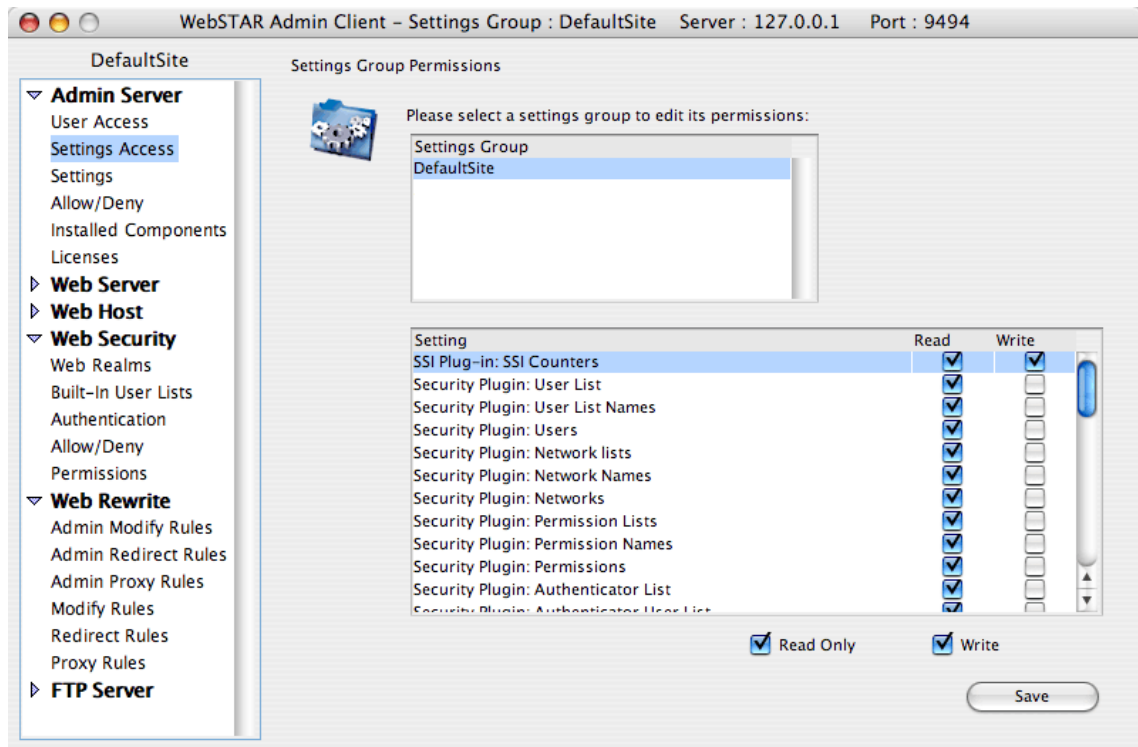


Figure 4.7 The default Settings Access panel

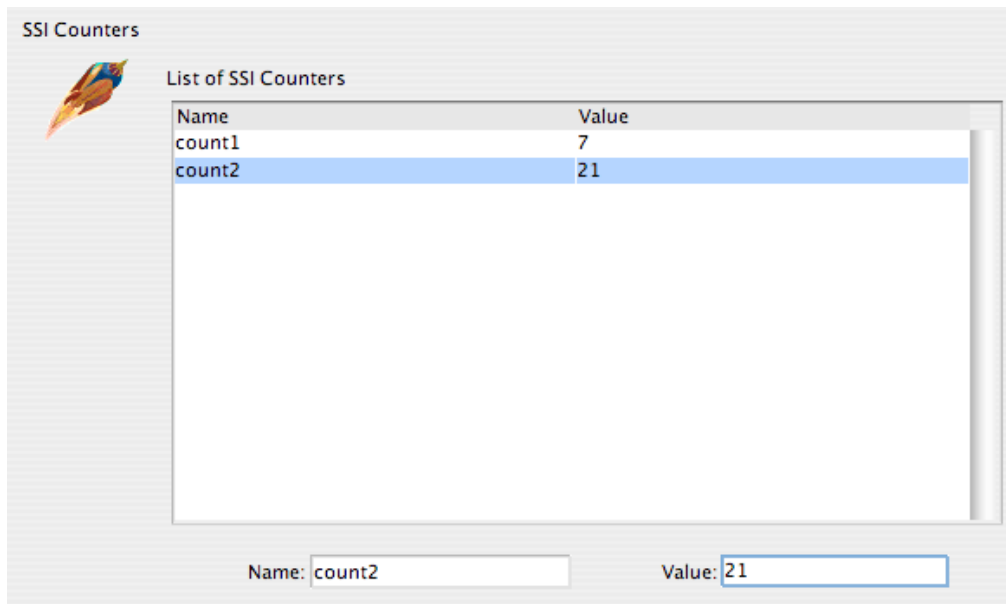
Each *Settings Group* is shown in the list on the top portion of the panel. The lower table shows the settings for the currently selected *Settings Group* (unless you add additional *Virtual Hosts*, there is only one *Settings Group*, *DefaultSite*, so it is automatically selected).

The settings in the lower table apply to all host admins for the selected virtual host, but do not apply to any full admins who, of course, also have access to the selected virtual host.

By default, read and read/write access is permitted for all features. If Read access is permitted but write access denied, the virtual host can see what is entered, but no changes can be made. If both read and read/write access is denied, the panel appears with no entries. Although it is possible to grant read/write permission but not read permission, this combination behaves as if both types of permissions are denied.

The following *Kerio WebSTAR 5* features can be controlled via the *Host Permissions* panel:

- *SSI Counters*: The list of SSI counters in the *SSI Counters* panel in the web host group. By default, this panel contains no counters. You must add the counters that you want to use.



The screenshot shows a window titled "SSI Counters" with a feather icon. It contains a table titled "List of SSI Counters". The table has two columns: "Name" and "Value". The first row is "count1" with a value of "7". The second row is "count2" with a value of "21" and is highlighted in blue. Below the table, there are input fields for "Name" (containing "count2") and "Value" (containing "21").

Name	Value
count1	7
count2	21

Figure 4.8 An SSI Counters table

- *Security — User List, User List Names, Users:* The names of user lists and the names of users within lists.

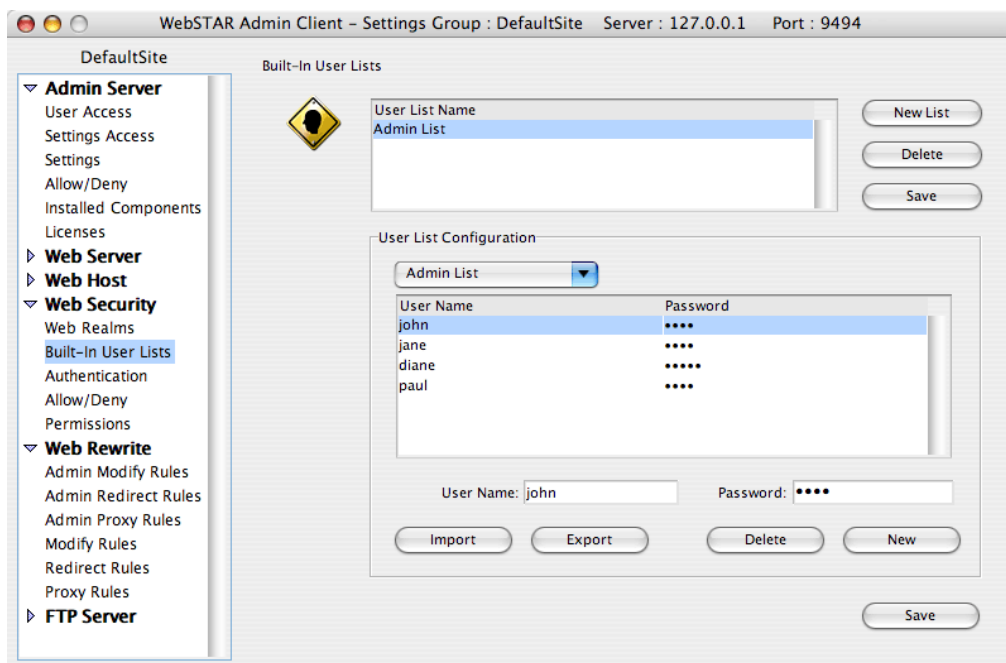


Figure 4.9 Built-in User Lists

This panel is described in the section 6.1.

- *Security — Networks, Network Lists, and Network Names:* This set of permissions refers to the *Web Security Allow/Deny* table, the list of tables and the lists of host-names/IP addresses:

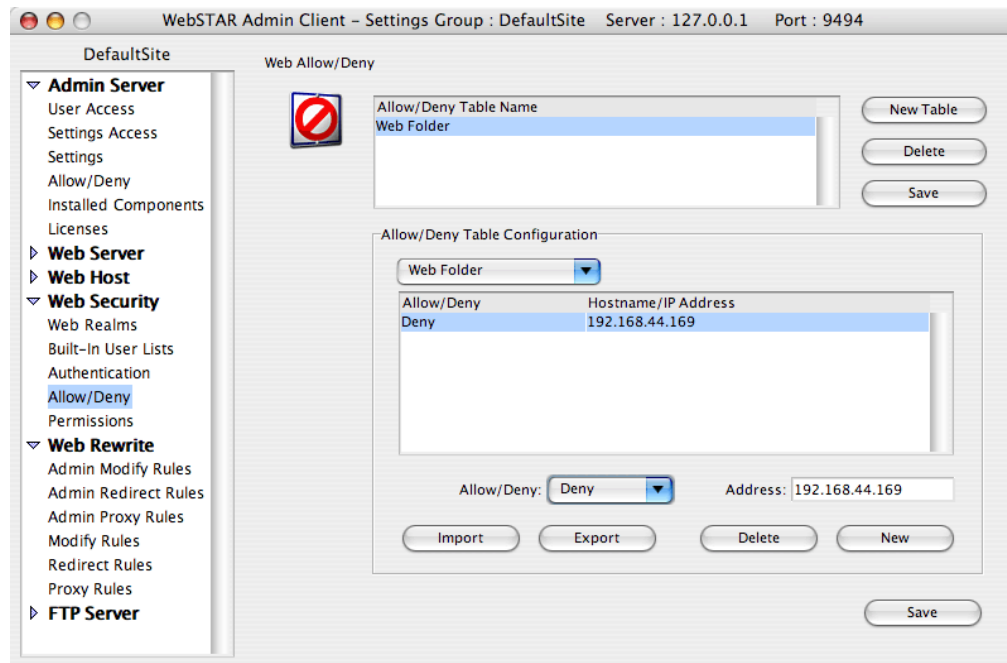


Figure 4.10 The Web Allow/Deny panel

An *Allow/Deny* table can be used in the definition of a realm to limit access to particular parts of your web site. This panel is described in the section 6.1.

- *Security — Permissions, Permissions Lists, Permission Names:* This set of permissions refers to the *Permissions* panel and the associated permissions lists and usernames:

This panel is described

- *Security — Authenticator List and User List:* This set of permissions refers to the *Authenticator List* and associated *User Lists*:

- *Security — Realm List, Authenticators, Networks (a.k.a., Allow/Deny), and Permissions:* This set of permissions refers to the *Realms definitions* panel and allows you to control permissions for the individual elements that define the realm:

4.3 Setting Access Rights for Host Admins

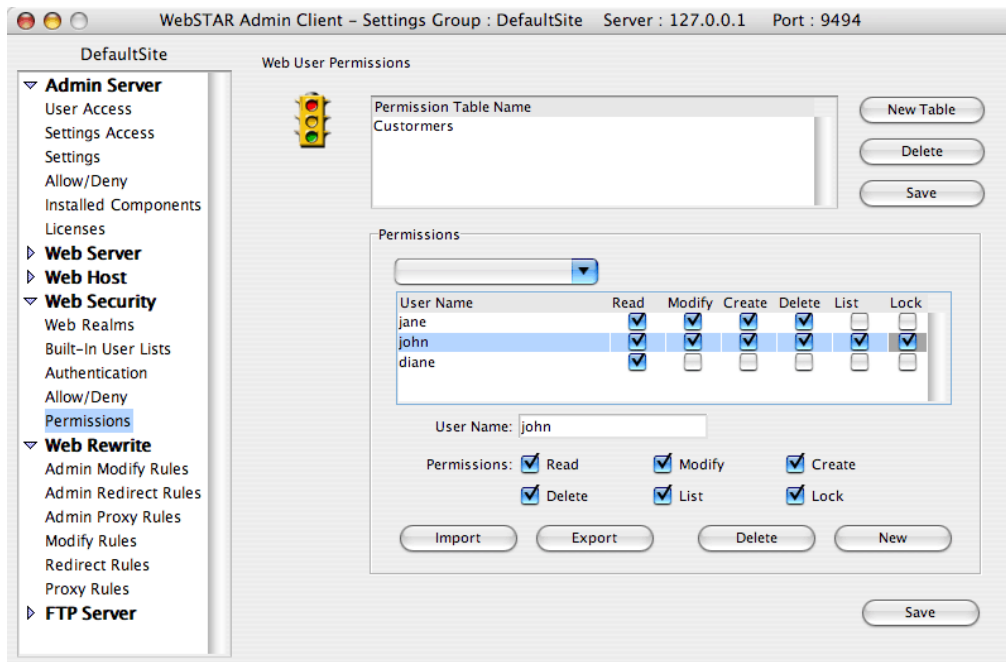


Figure 4.11 The Web User Permissions panel

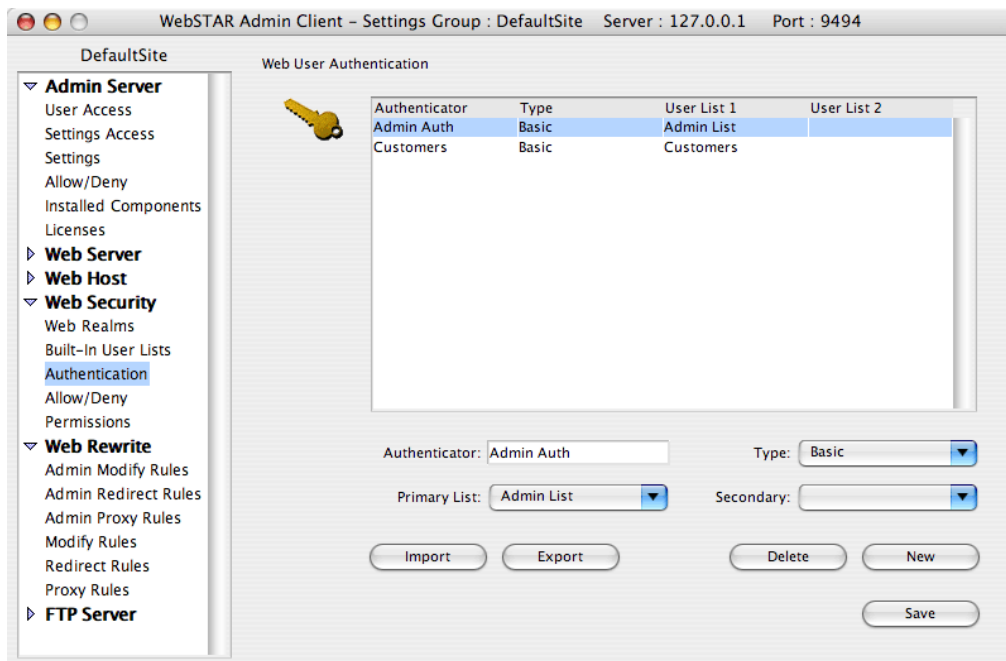


Figure 4.12 A selected Authenticator list and associated User list

- *Logging — Log Archiving (on/off), Log Archiving Path (a.k.a, folder), DNS lookups, Log Buffer & Log Buffer Size, Time to Begin, Month Day (unit of archive every), Unit*

Chapter 4 Basic Admin Server Settings

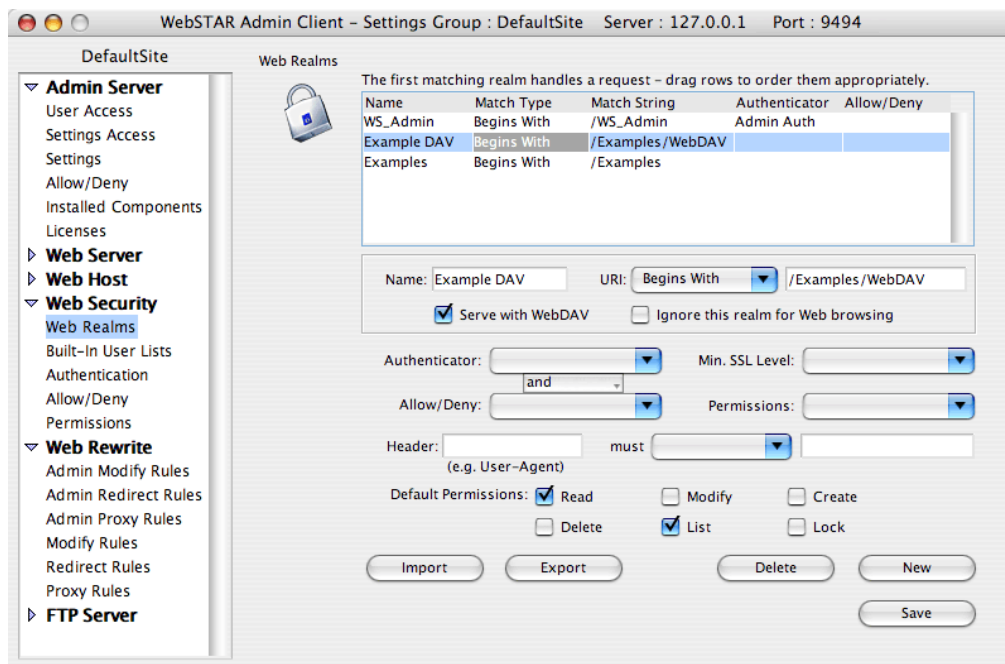


Figure 4.13 A Realm definition panel

Frequency, Virtual Enable, Weekday to Log: This set of permissions refers to the *Web Logging* options and give you detailed control over each parameter:

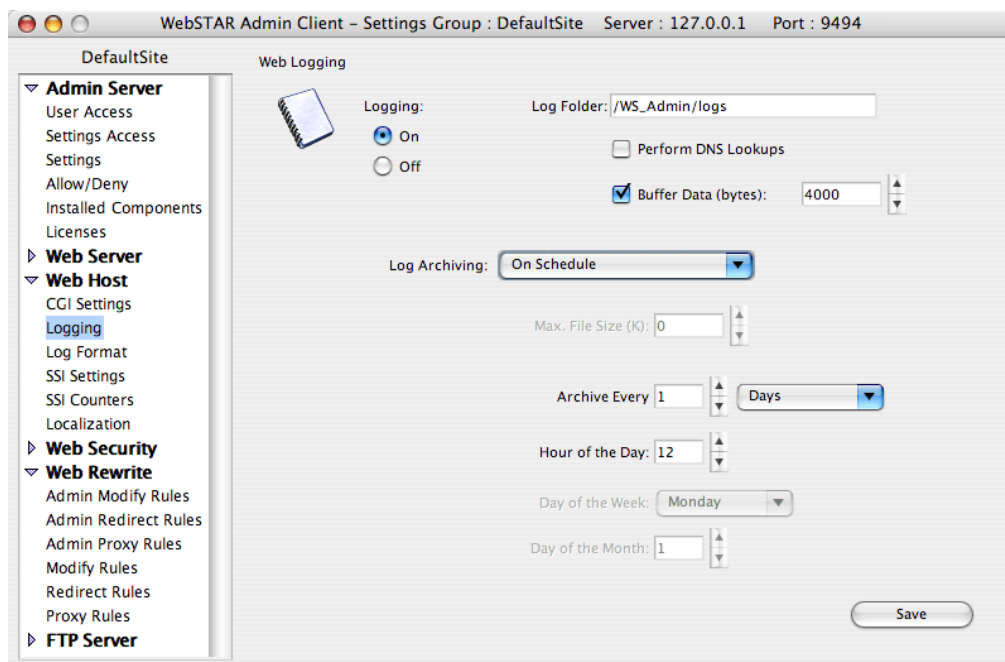


Figure 4.14 The Web Logging panel

- *CGI Action List and CGI permissions:* This set of permissions refers to the list of CGI actions on the *CGI Settings* panel and the five *CGI Permissions* checkboxes above the list:

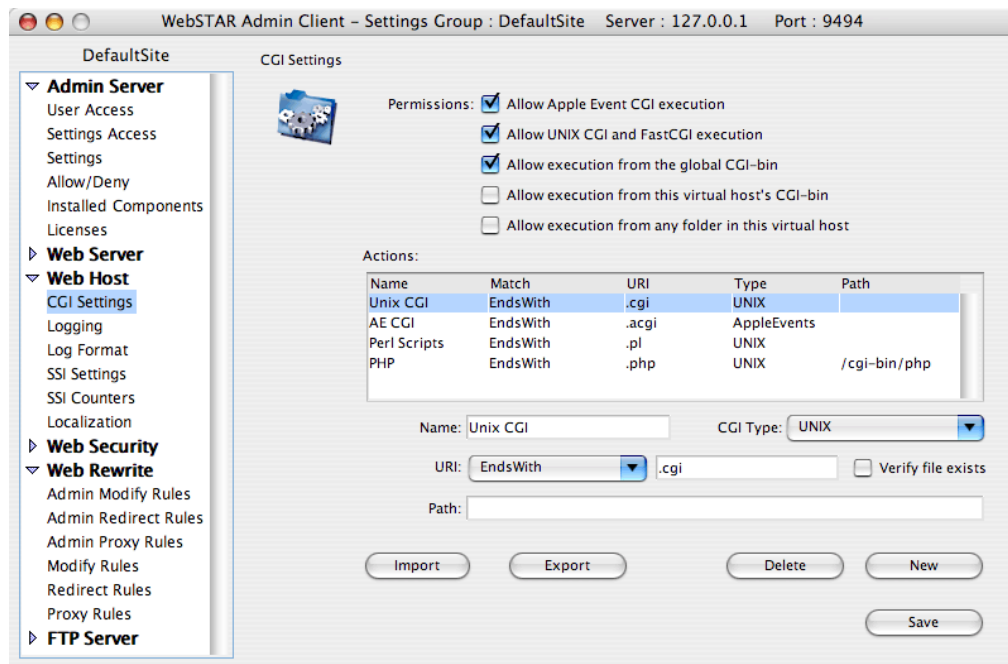


Figure 4.15 The CGI Settings panel

The last two permissions on this list (unchecked in figure 4.15) are less secure than the *Allow execution from the global CGI-bin* folder, so you may want to prohibit host admins from selecting these options. The last permission, in particular, is the least secure since it allows the execution of a CGI from anywhere within the virtual host.

Chapter 5

Creating Web Virtual Hosts

For reasons of security and convenience, we recommend that all documents that are served to browsers be located in folders within the *WebServer* folder. You organize your sites by creating one folder per virtual host and place the folders in the *WebServer* folder. After you organize the documents in this fashion, you can tell *Admin Server* about them. There are two basic tasks:

- Tell *Admin Server* which folders are virtual hosts and enter any necessary routing information,
- (Optional) Establish host admin access privileges for others.

You would need to do the second task only if you plan to delegate any administrative responsibilities to others. If you are going to administer all the virtual hosts yourself, you only need to give yourself full admin privileges.

5.1 Adding the Virtual Host Files to your Server Machine

By convention, all the files belonging to a virtual host are placed in a folder in the *WebServer* folder in the parent *KerioWebSTAR* folder. After you place the folder in the *WebServer* folder, you use *Admin Client* to tell the *Admin Server* about the virtual host and (optionally) grant host admin privileges to a designated virtual host administrator.

Whenever you add files to your hard disk that the web server uses, you should do so by logging in as the special *webstar* user that you created during the installation process. This ensures that the web server will have the appropriate file permissions. If you are running a production web site and are not using your Macintosh for anything else, consider only logging in as the special *webstar* user. In that way, the files you install in a virtual host's folder will always have the correct permissions.

A disadvantage is that the special *webstar* user does not have admin privileges. This is done so that, in the unlikely event that your site is attacked, the attacker will not have admin privileges and, therefore, cannot do much damage.

It does mean, however, that there are other Macintosh-related tasks that the special *webstar* user cannot perform.

If you do not install your files as the “webstar” user, you do have a backup option. You can run the *Kerio WebSTAR Installer* on your CD and execute the *Custom Install* option,

Configure File Permissions. This option executes a script that gives the web server the correct read/write file permissions for all files in the *KerioWebSTAR* folder.

To install the files for a new virtual host:

1. If you are not already logged in as the webstar user, choose *Log Out* from the Mac OS X Apple menu.
2. When the login dialog box appears, enter *webstar* as the user and enter the password you chose when you created this user.
3. Create a new folder in the *WebServer* folder and populate it with the correct files that make up the web site or drag an existing web site folder to the *WebServer* folder.

If you are moving your *WebSTAR 4* web site to *Kerio WebSTAR 5*, this will involve moving the files that are at the same level as the *WebSTAR 4* server, putting them in a folder, and moving the folder to the *WebServer* folder.

4. If you need to do other tasks that require admin privileges, choose *Log Out* from the Mac OS X Apple menu and then log in as a user with admin level privileges.

You are now ready to launch the server suite and *Admin Client*.

Note: Although it is recommended that you place your web virtual hosts in the *WebServer* folder, there is no technical requirement that you do so. You can place a web virtual host folder anywhere on your hard disk, provided you set pathnames and permissions appropriately.

5.2 Entering and Configuring Virtual Hosts

A default virtual host folder, *DefaultSite*, has been placed in the *WebServer* folder by the *Kerio WebSTAR 5 Installer*. If you have an existing site, you can publish that site simply by moving those files into this folder and (if desired) renaming the folder, as appropriate.

Your error *.html* files should be placed in an *errors* folder inside the *DefaultSite* folder. Each file should be named using the error number, e.g., *404.html*, *403.html*, etc.

If you are going to use the web server to serve more than one site, create new folders in the *WebServer* folder and move the relevant documents into those folders.

After you have organized the files that make up your site, you can tell *WebSTAR* about the site.

When you enter pathnames to the folders that make up your virtual hosts, you use Unix-style pathnames. Please see chapter 23 for more information.

To create a virtual host:

1. In the *Admin Client* application, click the *Web Hosts* item in the *Web Server* group.

The following screen appears.

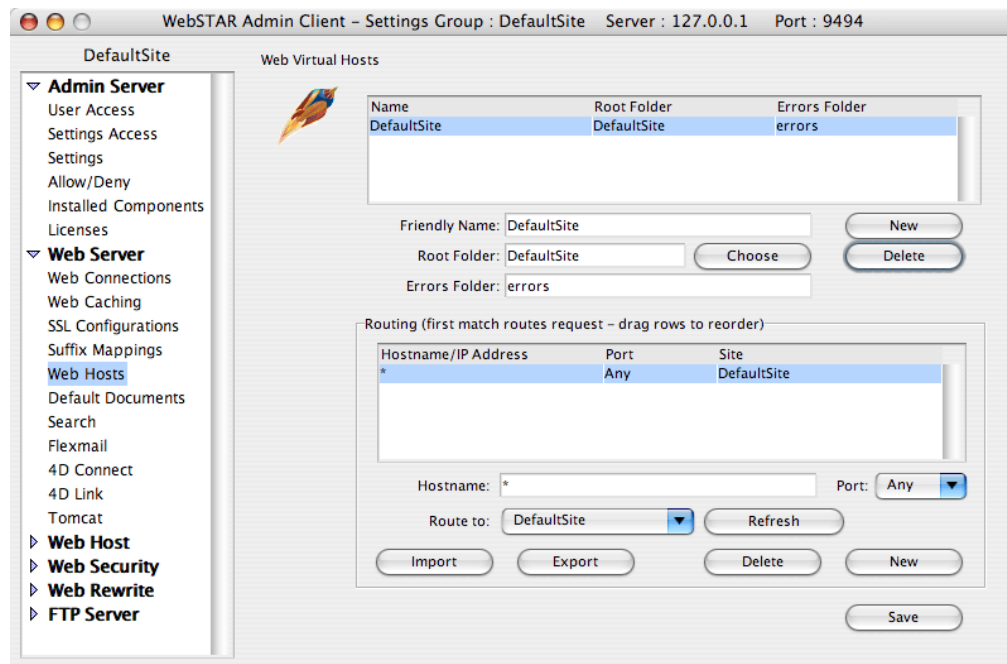


Figure 5.1 The Web Virtual Hosts screen

Use the *Web Virtual Hosts* list at the top to enter information about the each virtual host folder that you have created in your *WebServer* folder.

- The *Friendly Name* is the name of the web virtual host that appears in the *Settings Groups* dialog when you log in to the *Admin Server* application.
- The *Root Folder* is the relative path from the web server application to the root folder for the virtual domain (i.e., the folder where all the documents making up the site are stored).
- The *Errors Folder* is the relative path from the root folder to the folder containing the error messages. You put the *Errors* folder inside the root folder.

If you place your web virtual host folders in the *WebServer* folder, you can just use the folder names as relative paths.

2. Click the *New* button below the *Virtual Hosts* table to add a web virtual host.

A new row in the *Web Virtual Hosts* table appears. Default values are provided. You can enter/edit text directly into each field in the line or enter information into the entry areas below the table.

3. Enter the *Friendly Name* for the web virtual host.

If you are entering directly into the table, press the Tab key to move to the next field.

4. Enter the root folder either by typing the relative path from the web server application or click the *Choose* button to select the folder using a *Select Folder* dialog box.

For example, in figure 5.2, the *DefaultSite*'s root folder is being selected.

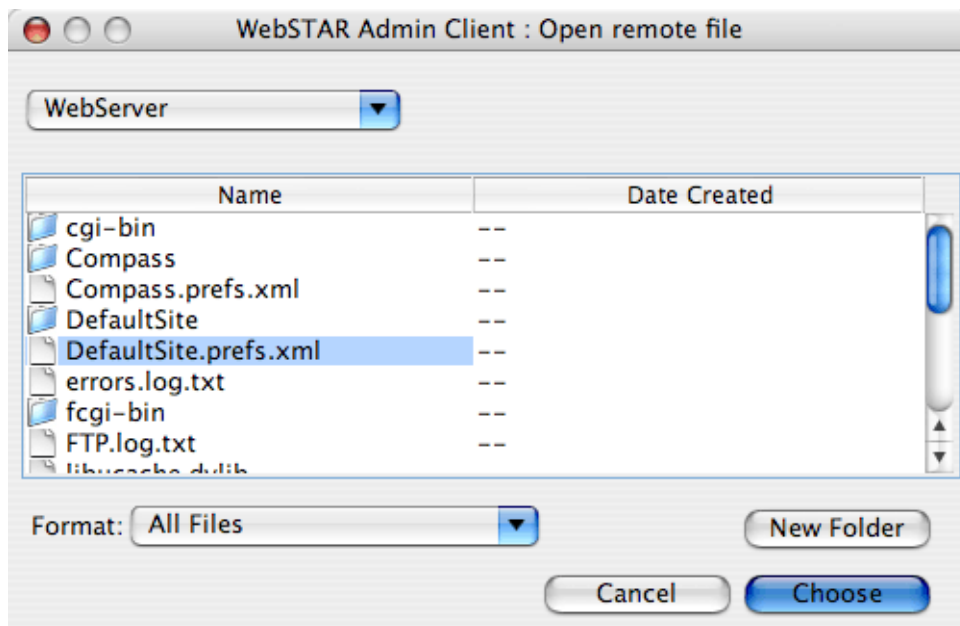


Figure 5.2 Selecting the *DefaultSite*'s root folder in the *Select Folder* dialog

5. (Optional) Repeat these steps for each web virtual host you want to create.

A completed *Virtual Hosts* table looks like this:

In figure 5.3, a second virtual host has been entered. The path to the root folder was specified using the *Choose* button.

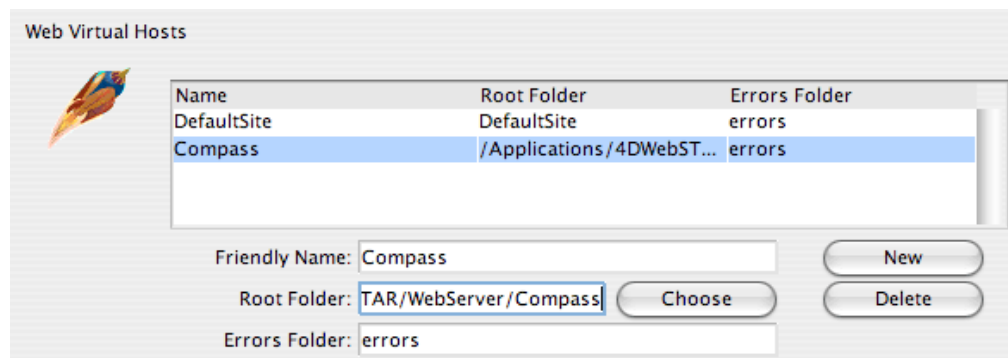


Figure 5.3 A completed Virtual Hosts table

Using the Routing Table

The next task is to specify how incoming requests are to be routed to your virtual hosts. If you are only serving one virtual host, the issue is simple: Route all incoming requests to that host. This is the meaning of the default entry: All requests from any port are routed to the *DefaultSite* virtual host. If this is the case for you, you don't have to edit the *Routing* table.

For example, if you replaced the contents of the *DefaultSite* folder with your site, simply route all connections to the server machine to that virtual host. Figure 5.4 shows an example of routing by IP address and by domain name:

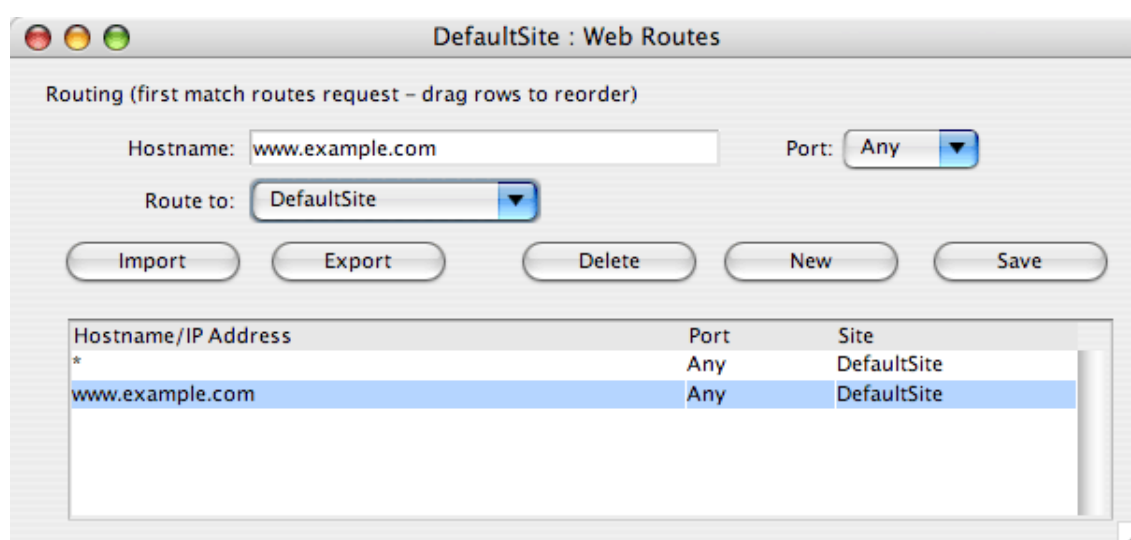


Figure 5.4 Routing connections to the DefaultSite virtual host

The *Routing* table also allows you to specify two or more IP addresses or domain names and route them to a designated virtual host. Figure 5.5, for example, shows two domain names that are all routed automatically to the same site.

You can map more than one variation of your hostname to the same virtual host. For example, `mydomain.com`, `www.mydomain.com`, and `dev.mydomain.com` can be routed to a single virtual host site, allowing you to have only one settings file for all the variations.

If you use the *Routing* table to select a particular port, be sure that the port or ports you specify are also specified in the *Web Listeners* table on the *Web Connections* panel.

The *Routing* table is order dependent. *WebSTAR* will use the first row that matches the incoming request and route according to that mapping. You can reorder the rows in a table by dragging rows. See the section 3.2 for more information.

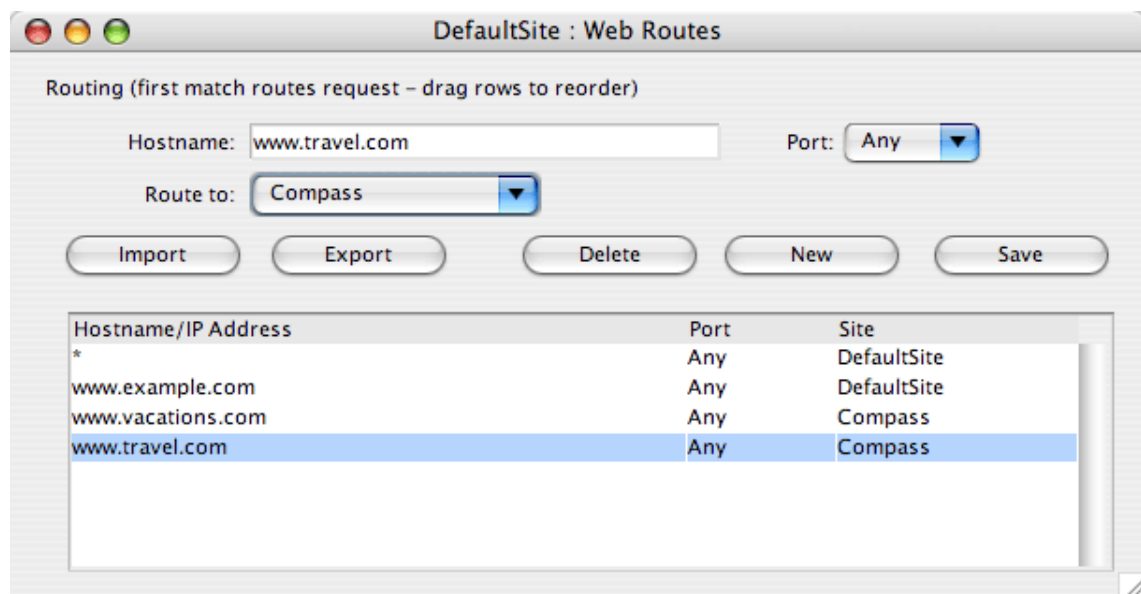


Figure 5.5 Using Routing to direct two domain names to one virtual host

5.3 Granting Access Rights to your Virtual Hosts

If you are the only user with access rights to *Kerio WebSTAR 5*, then you, as a full admin, automatically have access rights to all web virtual hosts that you create. The next time you log in (either by choosing *File → New* or when you next launch the *Admin Client* application), the *Settings Groups* dialog box will now list all the virtual hosts that you have just created. You can pick one in the list to configure that virtual host. If you want to configure another virtual host within that session, choose *File → New* to open a new window or choose *File → Switch Settings Group* to open the other virtual host in the same window.

If you want to create host admin users (admins who can only administer certain virtual hosts), you need to return to the *Admin User Access* panel to add these users and set their access rights. Please see chapter 4 for more information.

Chapter 6

Web Security

The security model in *Kerio WebSTAR 5* is much more flexible than Mac OS 9 versions of *WebSTAR*. All settings are configured separately for each web virtual host. User lists, allow/deny tables, and permissions are now reusable, and can be assigned to multiple realms within a virtual host.

Security is based on the concept of a realm. A realm is usually a folder or file within a web virtual host's root folder. The realm is actually defined by a string that is contained within a request to the server. A request that contains that the string that defines the realm is intercepted by the security system and the requesting browser's credentials are challenged. For example, the browser may be asked to supply a username and password. If the browser is unable to provide the username and password, access is denied.

Kerio WebSTAR 5 does not require that the realm actually be defined by a physical folder or file. A realm is defined by any special word or phrase in the request. You define that string as the realm.

When a request contains the string that defines a realm, *Kerio WebSTAR 5* can “challenge” the requestor's credentials in two ways:

- You can set up a user list that contains usernames and passwords for only those people who are authorized to access a realm.

Moreover, you can set up default permissions for the authorized users (e.g., read only, create, modify, etc.) and, if desired, set up separate permissions for each user. A user can have different permissions for different realms.

- You can set up an *Allow/Deny* table that lists domains or IP addresses that are either permitted or not permitted access.

In the *Kerio WebSTAR 5* security model, *User Lists*, *Permissions Lists*, and *Allow/Deny* tables are separate objects. They are stored separately from *Realms*. This means that they can be reused in different realms.

You can, of course, use a *User List* in combination with an *Allow/Deny* table. The *Allow/Deny* table is used to block access from unauthorized host names, the *User Lists* is used to limit access to particular users, and the *Permissions* list is used to control level of access for authorized users.

The group of screens in the *Web Security* topic allow you to specify realms and you can use several ways to challenge a potential browser's rights to access the protected material.

6.1 Setting up Access Challenges

You create *User Lists*, *Permissions Lists*, and *Allow/Deny* tables on separate panels. You then reference them on the *Web Realms* panel when you create the realm definition.

Setting up User Lists

With the *User Lists* panel in the *Web Security* group, you can set up any number of lists and enter the usernames and passwords for each authorized user. The administrators list appears by default, but contains no names.

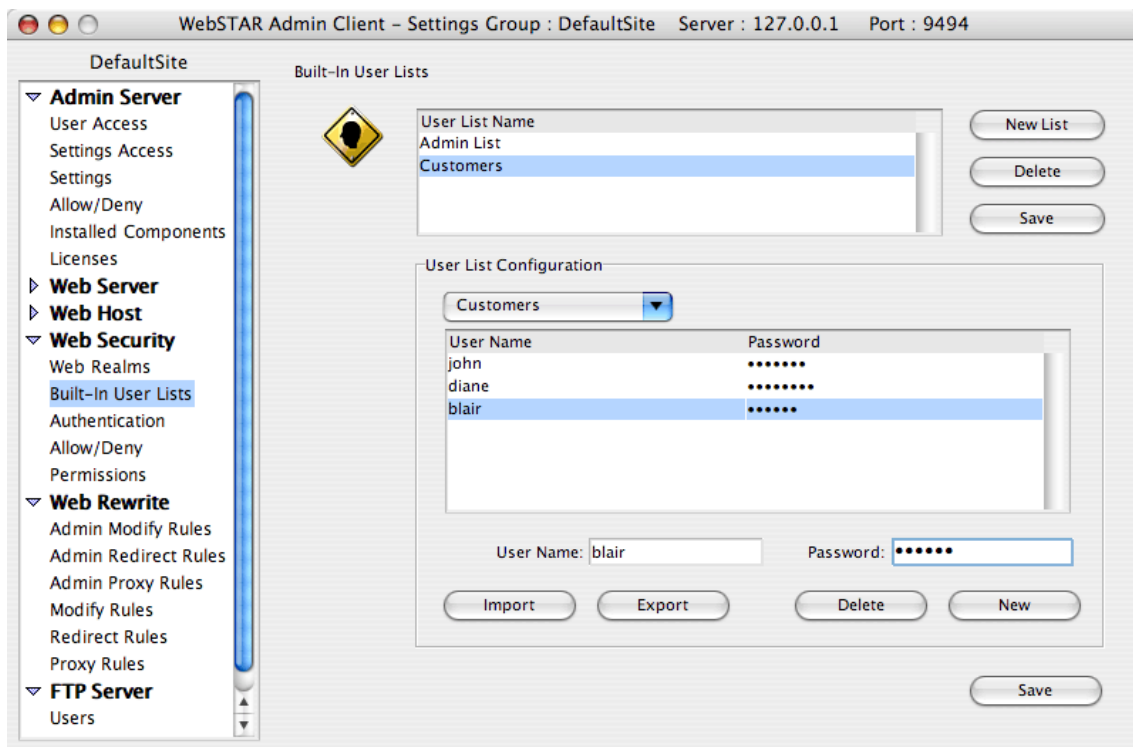


Figure 6.1 The Built-In User Lists panel

In figure 6.1, a second user list has been added and three names have been added to it in the *User List Configuration* area.

When you use this form of security, the user is confronted with a dialog box that requests his/her name and password whenever he/she requests access to the protected realm. The WebDAV login dialog used by Mac OS X is shown in figure 6.2.



Figure 6.2 The User Authentication dialog box

If the user supplies the correct information, his/her usage of the realm is controlled by the assigned access privileges. The access privileges may either be the *Default Permissions* (set on the *Realms* panel) or set via a *Permissions List*, which is set in *Permissions* panel and linked to the realm.

To add a new user list:

1. Click the *New List* button.
A new row appears in the *Built-In User Lists* table.
2. Type the new list name directly into the table.
3. Click *Save* in the *Built-in User Lists* area to save the list.

When you save the list name, it is added to the pop-up menu in the *User List Configuration* area.

To add users to a user list:

1. Choose the name of the list from the *User List Configuration* pop-up menu.
2. Click *New* in the *User List Configuration* area.
A new enterable area appears in the *User List Configuration* table.
3. Enter the *Username* and *Password* either into directly into the table or in the two entry areas below the table.
4. Repeat this process for each user.
5. Click *Save* to save your changes to the user lists.

Using a List as an Authenticator

Once you have created a user list, you need to indicate that you wish to use the list as an authenticator. An authenticator is the object that challenges a browser's access rights to a realm. An authenticator can be made up of more than one list.

The *Authentication* panel allows you to define up to two user lists as an authenticator. Later, using the *Web Realms* panel, you link the authenticator — not the user lists themselves — to the realm definition.

In figure 6.3, an authenticator named *Customers* has been defined as the list *Customers*.

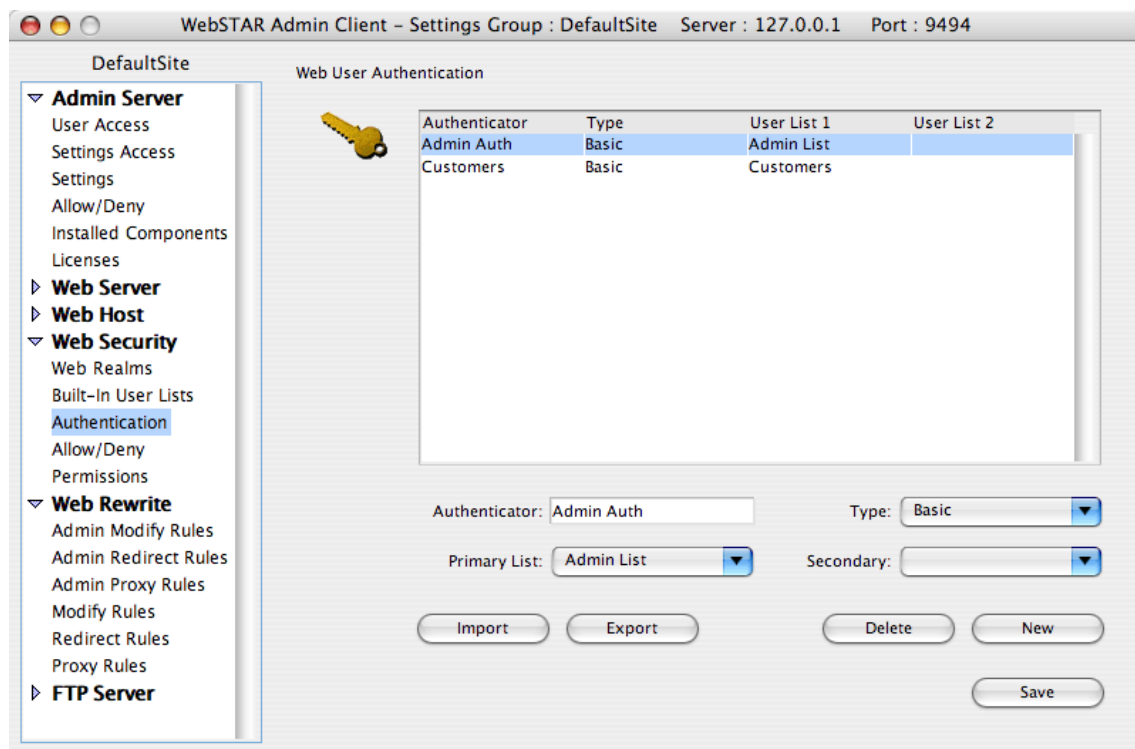


Figure 6.3 The Web User Authentication panel

To create a user list authenticator:

1. Click the *Authentication* item in the *Web Security* group.
2. Click *New* to create a new item in the *Web User Authentication* table.

A new blank row appears in the table.

3. Enter the name of the authenticator directly into the table or into the authenticator entry area below the table.

4. Choose the user list from the *Primary User list* pop-up menu.
5. (Optional) Choose another list from the *Secondary List* pop-up.
6. (Optional) Repeat the process to create additional authenticators.
7. Click *Save* to save the authenticator.

Setting Permissions for Individual Users

When you use a user list as an authenticator to limit access to a realm, the *Realms* panel allows you to set permissions for the users as a group. You can set *Read*, *Modify*, *Create*, *Delete*, *List*, and *Lock* permissions. When permissions are set on the *Web Realms* panel, these permissions apply to all the people referenced by the authenticator.

If you wish, you can also set separate permissions for individuals. You do this by creating a *Permissions* list using the *Permissions* panel and then linking that list to the realm.

You only use the *Permissions* panel when you need to set permissions on a per-individual basis.

Under this scenario, the *Authenticator* list controls who has access to the realm, while the *Permissions* list controls level of access for those individuals.

In figure 6.4, the users in the *Customers* list entered in figure 6.1 are granted permissions on an individual basis. Although the *Permissions* list bears the same name as a user list, please remember that it is a logically separate object. This naming convention simply makes it easier to identify the correct *Permissions* list when using the *Customers* user list.

Although you can create a one *Permissions* list for each user list (consisting, for example, of all the people on each user list), there is no requirement for you to do so. A *Permissions* list is a logically separate object from a user list. For example, you can only add the usernames of people for whom the *Default Permissions* are not appropriate.

A *Permissions* list can also mix people from two or more user lists.

If a person should have different levels of access to different realms, then you create several *Permissions* lists in which the person has different levels of access on each list. You then link the different *Permissions* lists to the different realms.

In other words, you use the *Permissions* panel only when you need to “fine tune” access rights.

To set access permissions for individuals:

1. Click the *Permissions* tab in the *Web Security* screen.

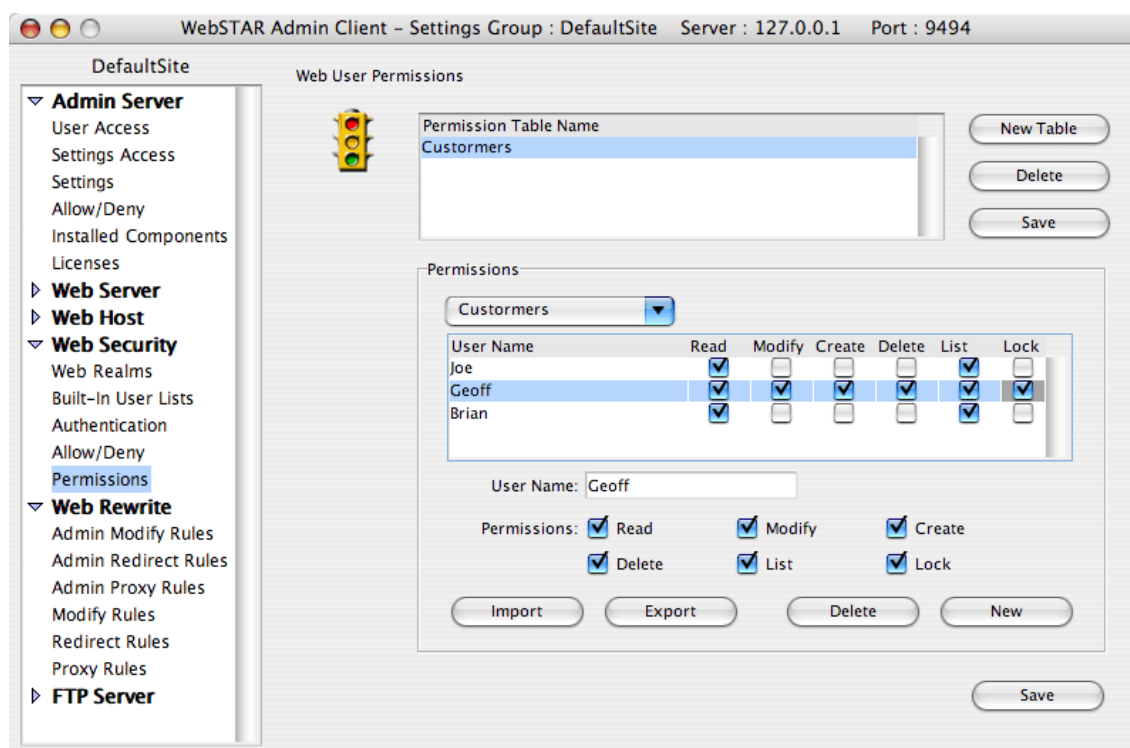


Figure 6.4 A Permissions table configured to grant full permissions to Geoff and restricted permissions to Brian and Joe

By default, no *Permissions* tables are entered.

- In the list of permissions tables area, click *New Table* to add a new table.
A new blank row in the *Permissions* table appears.
- Double-click in the blank row to get an insertion point and enter the name of the permissions table.
If necessary, create additional table names.
- Click *Save* in the permissions tables area to save the list names.
- Chose the a list from the *Permission Table Configuration* pop-up menu.

Next, you add web users to the *Permissions* table. These users should have already been entered into user lists.

- Click the *New* button in the *Permissions* table area to add a person to the *Permissions Table* list.

User Name	Read	Modify	Create	Delete	List	Lock
Joe	☒	☐	☐	☐	☒	☐
Geoff	☒	☒	☒	☒	☒	☒
Brian	☒	☐	☐	☐	☒	☐

User Name:

Permissions: ☒ Read ☒ Modify ☒ Create
☒ Delete ☒ List ☒ Lock

Figure 6.5 A Permissions panel with one Permissions list name entered

User Name	Read	Modify	Create	Delete	List	Lock
Joe	☒	☐	☐	☐	☒	☐
Geoff	☒	☒	☒	☒	☒	☒
Brian	☒	☐	☐	☐	☒	☐
	☒	☐	☐	☐	☐	☐

User Name:

Permissions: ☒ Read ☐ Modify ☐ Create
☐ Delete ☐ List ☐ Lock

Figure 6.6 Adding a person to the Customers Permissions list

A new blank row is added to the end of the list.

2. Enter the name of a user who has previously been added to a *Built-in User List*.
3. Use the checkboxes to set permissions for each individual.

Add as many people as needed to the *Permissions* list.

A *Permissions* list can contain individuals from several *Built-in User Lists*.

In fact, it is possible for a *Permissions* list to contain the names of individuals who are not on any user list, but such people would have no *Username* or *Passwords*!

4. Click *Save* to save the *Permissions* list to the server.

Using an Allow/Deny Table to Restrict Access

You can also limit access to realms by entering a list of authorized or unauthorized IP addresses or hostnames. You do this by setting up an *Allow/Deny* table using the *Allow/Deny* panel in the *Web Security* group.

In figure 6.7, the IP addresses of three computers on the network have been entered into the *Allow/Deny* table named *Web Folder*. When this table is linked to a realm, only those three computers will be able to access the realm.

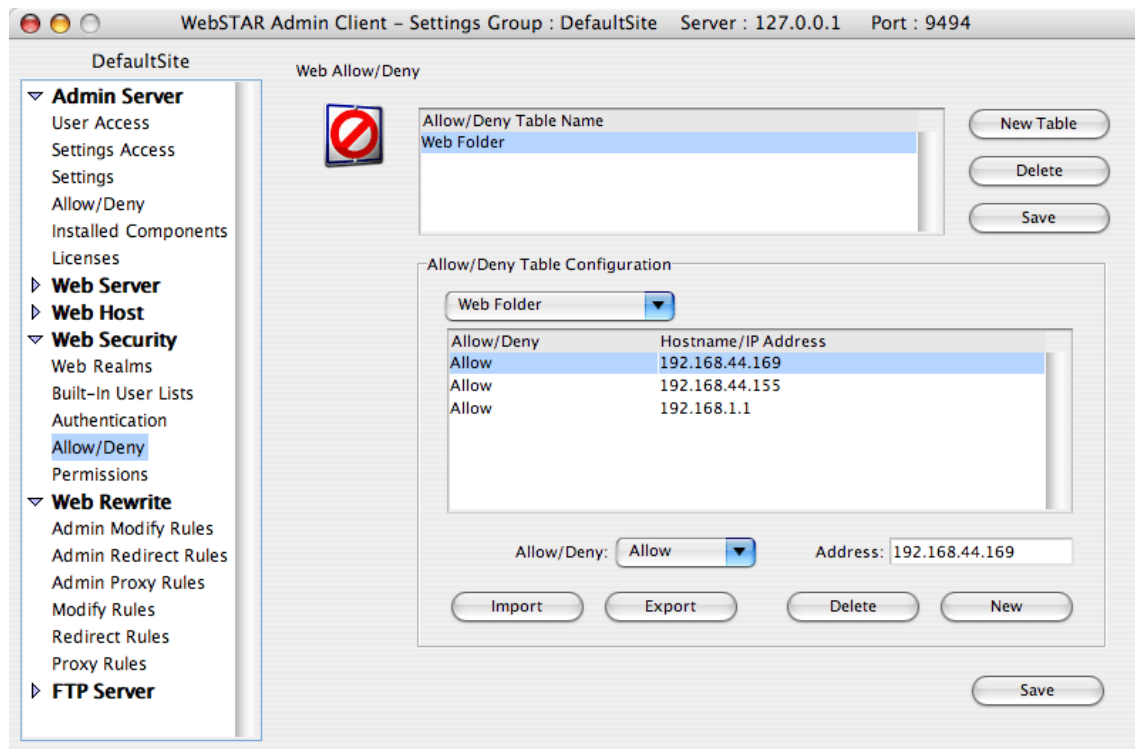


Figure 6.7 The Web Allow/Deny panel configured to allow access for three computers on the network

If you enter one *Deny* machine, all others are allowed.

When you set up a realm, you need to link the *Allow/Deny* table to the realm in order for its specifications to take effect.

6.2 Setting up a Realm

After you have set up the desired authentication and permissions objects, you can proceed to define a realm. The realm consists of:

- The realm definition — the string that is found in the request which will indicate to *Kerio WebSTAR 5* that the security system must intercept the request and challenge the requestor's credentials.
- An Authenticator and/or *Allow/Deny* table. The authenticator will present a *Username and Password* dialog box and require that the browser enter valid credentials before serving the requested page. The *Allow/Deny* table will check to see that the request has come from an authorized computer. Either or both can be used.
- (Optional) If you want to control level of access at the level of individual users, link a *Permissions* table to the realm. Please bear in mind that *Kerio WebSTAR 5* doesn't check to verify that the persons on the *Permissions* list are also on the *User List*. The *Permissions* list doesn't have to have an entry for each person, but if a person on the *Permissions* list is not entered in the user list, he has no username and password and can not access the realm.
- (Optional) If you have configured your server to handle encrypted communications using SSL (see chapter 16), you can set up a realm that requires a *Minimum SSL Level*. Your choices are *None*, 40-bit, and 128-bit. The default, of course, is *None* (no encryption); the latter two choices permit only encrypted communications using a cipher described in the section 16.1.

For example, if you need to serve sensitive information such as patient records or financial data, you can organize those documents in a realm and require that browsers requesting such documents support a specified encryption level.

Here are several example of realms:

- *Authenticator* in figure 6.8, a realm is defined by the string */Customers*. When this string is found in a URL, the security system intercepts the request and displays a dialog box in which a user can enter his/her username and password.

Since the realm uses the *Customers* authenticator, only the usernames defined in figure 6.1 will be granted access to the realm. Their access privileges are controlled by the *Default Permissions* checks at the bottom of the *Web Realms* panel.

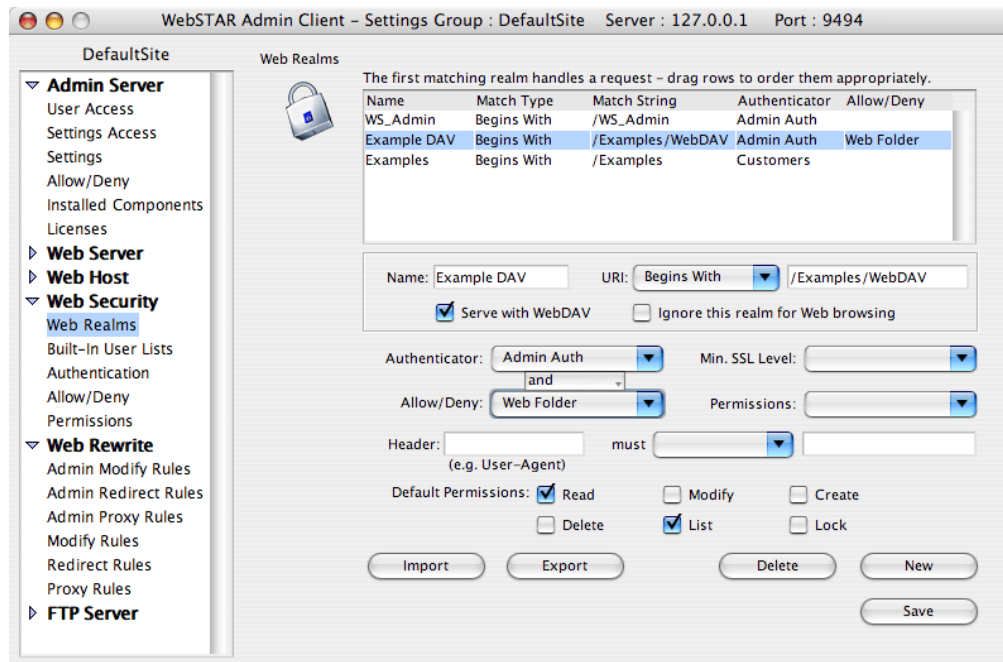


Figure 6.8 An Authenticator that references the User list defined in figure is used in a realm definition

- Authenticator and *Allow/Deny* table in figure 6.9, the *Allow/Deny* table created in figure 6.7 has been added to the realm definition. Therefore, only requests from these three machines are granted access. Since the *Customers* authenticator is also used, users attempting to log in from these machines must also enter their usernames and passwords. Although permissions for those users has been set up using a *Permissions* list (figure 6.5), it is not used. This means that the default permissions shown in this screen will apply.
- User list and *Permissions* list in figure 6.10, the *Permissions* table has been added to the realm definition and the *Allow/Deny* table has been removed. Under this scenario, the three users entered into the user list (figure 6.1) can access the realm from any computer; after successful login, their browsing privileges are controlled by the *Permissions* table of figure 6.4.

To define a realm:

1. Click the *Web Realms* item in the *Web Security* topic.
The *Realms* panel appears.
2. Click the *New* button to create a new realm.
A new row appears in the *Web Realms* table at the top of the screen.

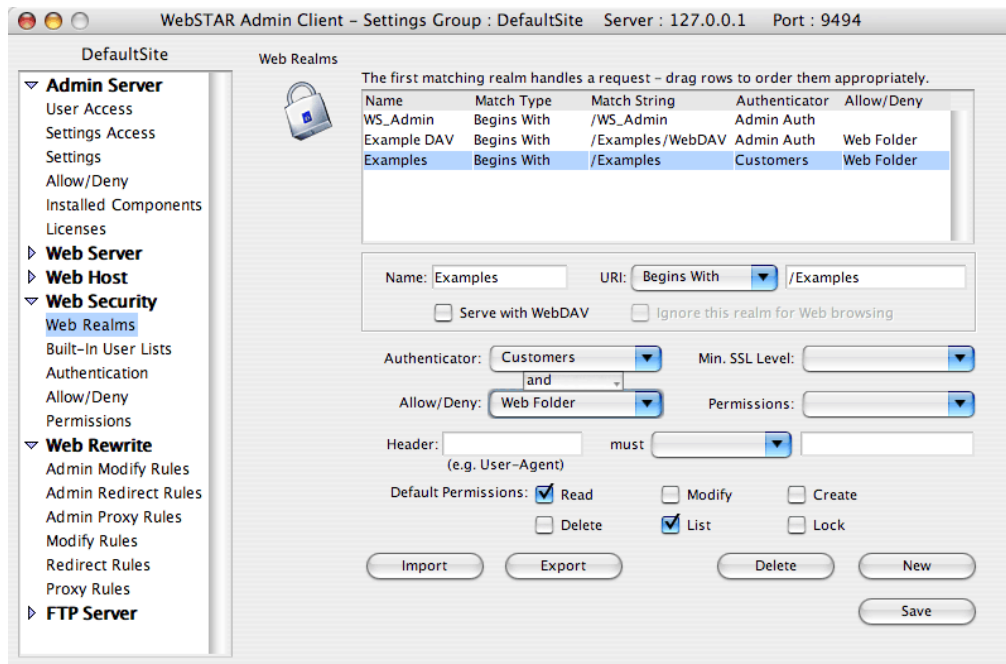


Figure 6.9 The Allow/Deny table shown in figure limits access to the realm to the three machines. The Authenticator (User lists) is also used

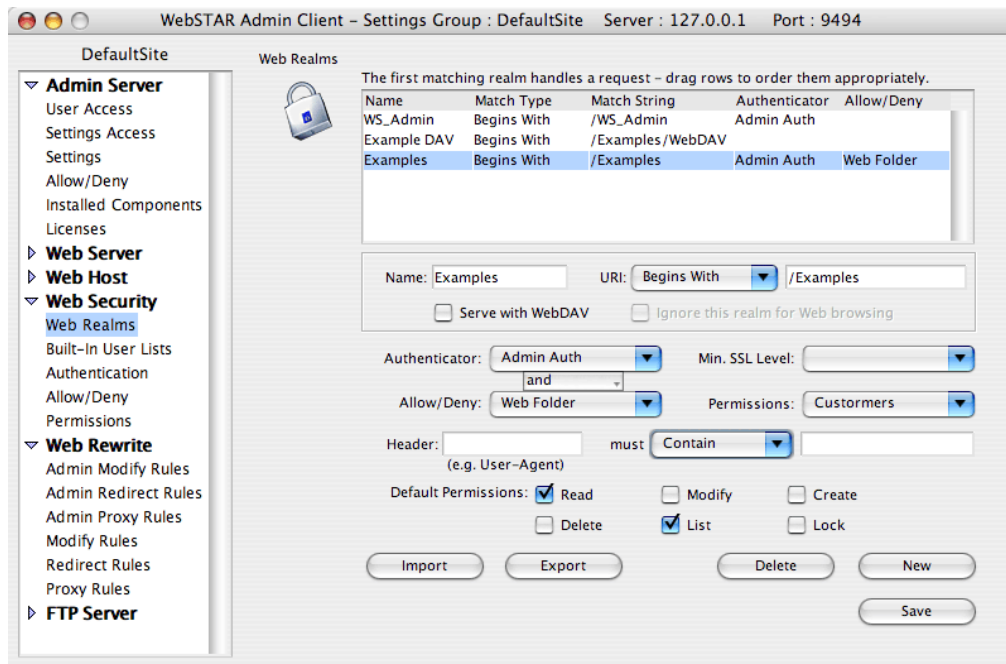


Figure 6.10 The Permissions table shown in Figure 50 has been added to the realm definition

3. Enter the realm name either in the new row in the table or in the *Name* area below

the table.

4. Use the URI pop-up menu to select the search criterion.
Your choices are *Contains*, *Begins with*, *Ends with*, or *Matches Regular Expression*. The latter allows you to use standard regular expressions to define the search string. For information about regular expressions, see chapter 9.
5. Enter the string to search for in the *Match String* column or entry area.
Next, enter the “challenge” to a browser’s access rights to the realm. You can use an Authenticator, an *Allow/Deny* table or both.
6. Choose an Authenticator and/or an *Allow/Deny* table from the popup menus.
Next, specify the permissions granted to authorized users.
7. If you want one set of permissions to apply to some or all authorized users, click the appropriate *Default Permissions*.
8. If any user’s permissions are set by a *Permissions* list, choose the list from the *Permissions* pop-up menu.
9. If you need to create additional realms, repeat steps 2 to 8.
10. Click *Save* to save the realm and its settings.

Chapter 7

Creating Web Folders with WebDAV

WebDAV (Web Distributed Authoring and Versioning) is a network protocol that extends the http protocol to provide file sharing over the internet. WebDAV-enabled applications allow geographically separated people to work together by sharing files using the internet.

Kerio WebSTAR 5 is a WebDAV server, enabling you to publish folders over the internet. With WebDAV, you can:

- Mount a shared folder as a volume on your desktop and browse its contents in a Finder-like fashion,
- Upload files to the shared folder by dragging from a local volume to the shared folder,
- Create new folders on the shared folder,
- Delete files and folders on the shared folder by dragging to the Trash,
- Renames files and folders on the shared folder,
- Download files from the shared folder to your computer by dragging from the folder to a local volume.

As a WebDAV administrator, you use *Kerio WebSTAR 5*'s web security features to authenticate users and regulate permissions. You can block access to unauthorized users entirely and selectively grant permissions on an individual basis. Although all the capabilities listed above are available, you don't need to make them available to all your WebDAV users. That is, all the security features described in the previous chapter are available to regulate WebDAV folder access.

7.1 WebDAV Client Applications

A WebDAV client application or WebDAV aware operating system provides access to a WebDAV folder. Under Mac OS X, users need no separate client application. Using the Finder's **Go . Connect to Server** menu, a Mac OS X computer can mount a WebDAV volume. Mac OS 8-9 users use a WebDAV client utility such as *Goliath*, and Windows users can use *Internet Explorer* or *Microsoft Office* applications. A Mac OS X version of

Goliath is also under development and can be used instead of the WebDAV client built into Mac OS X.

7.2 Setting up WebDAV Service

To set up WebDAV service, you need to create a virtual host that references the folder to be published on the web. After you've created this folder, the process is very similar to creating an ordinary realm. The realm is the *Web Folder*. The *WebDAV* folder can simply be a folder within the root *Virtual Host* folder.

You use a user list to limit access to the realm and define the *Web Folder* as a realm. Using *Default Permissions* (on the *Web Realms* panel) or a *Permissions* list, you can restrict access rights to the *Web Folder*. For more information, see [6.1](#) and [6.2](#).

To publish a folder on the web using WebDAV:

1. Create a folder within your *WebServer* folder that you wish to share over the internet and place the files you want to share in the folder.

It can either be a separate virtual host or a folder within an existing virtual host. Follow the steps in chapter 5.

2. (Optional) If the WebDAV folder is to be a separate virtual host, log in with full admin privileges and select the virtual host that is or includes the web folder.

If the *WebDAV* folder is within an existing virtual host, you don't need to do this. You instead define this folder as a realm in the *Web Realms* panel.

The next group of steps creates the list of users who are permitted access to the *Web Folder* and set up their permissions.

1. Click the *Built-In User Lists* item in the *Web Security* topic or choose *Built-in User Lists* from the *Admin Client Tools* menu.

The latter opens the *User Lists* panel as a dialog box.

2. Create the list of users who are authorized to access the *WebDAV* folder.

You can either add the users to an existing list or create a new list.

If you want to create a new list, click the *New List* button, enter the name of the list in the table, and click *Save* in the *Built-in User Lists* area. Next, choose the new list name from the *User List Configuration* pop-up and click *New* to add each username and password. Click the *Save* button at the bottom of the screen to save the new user list.

If you want to add the usernames to an existing list, simply select it from the *User List Configuration* pop-up menu and add the names to that list. Click *Save* at the bottom of the screen to save the new usernames.

3. Click the *Authentication* item in the *Web Security* topic.

4. Click *New* to create a new authenticator.

Give the new authenticator a name that indicates it controls access to the *WebDAV* folder.

5. Choose the name of the list you created or modified in the *Built-in User Lists* screen in the *Primary List* pop-up menu.

This will mean that only those users will be able to access the *Web Folder*. They will be prompted to supply their username and password.

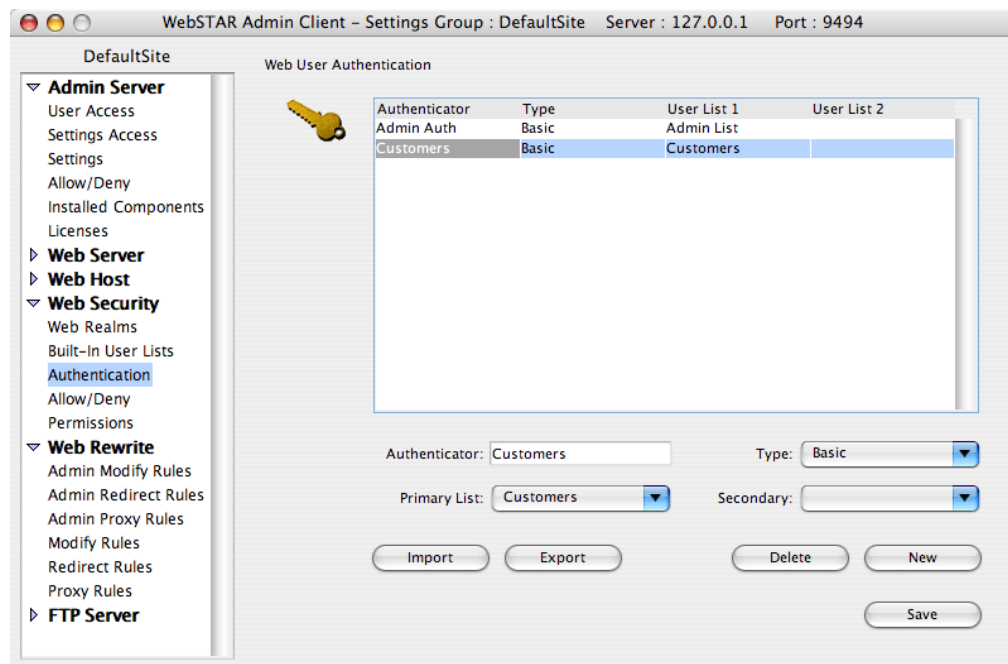


Figure 7.1 A User Authentication screen uses the Customers list

6. Click *Save* to save the new authenticator.

7. Click the *Web Realms* item in the *Web Security* topic.

Next, you will define a realm that tests whether the browser is trying to access the WebDAV folder and link this realm to the authenticator.

1. Click *New* to create a new realm entry that is ready to serve for WebDav.

Give it a name that indicates that it checks for access to the *Web Folder*.

2. In the *Match String* field, enter a string that is unique to requests that access the *Web Folder*, such as the folder's name.

The comparison pop-up menu gives you the choices of *contains*, *begins with*, *ends with* and, *matches a regular expression*. For information about regular expressions, see chapter 9.

3. Use the *Authenticator* pop-up to choose the authenticator that you created earlier.
4. Click the *Serve with WebDAV* checkbox.
5. In the *Default Permissions* area, check the desired permissions.

Use Read and List for *read only* privileges and add *Create* and *Modify* to grant *read/write* privileges. These permissions will be applied to all users in the user list linked to this realm. If you want to modify permissions for specific individuals in this list, you can do so using the *Permissions* item in the *Web Security* topic. For more information, see section 6.1.

6. Click *Save* to save the realm settings.

Figure 7.2 shows that the folder *Pictures* within the current virtual host is set up as a *WebDAV* folder.

All the users on the *Customers* authenticator have read/write privileges (*Read* and *List*, plus *Modify* and *Create*), but they cannot delete or lock files.

Figure 7.3, the realm is configured so that the entire web site is published as a web folder. This is because any request that includes a slash is directed to the *WebDAV* folder. This allows you to use WebDAV to develop your site remotely via WebDAV.

Chapter 7 Creating Web Folders with WebDAV

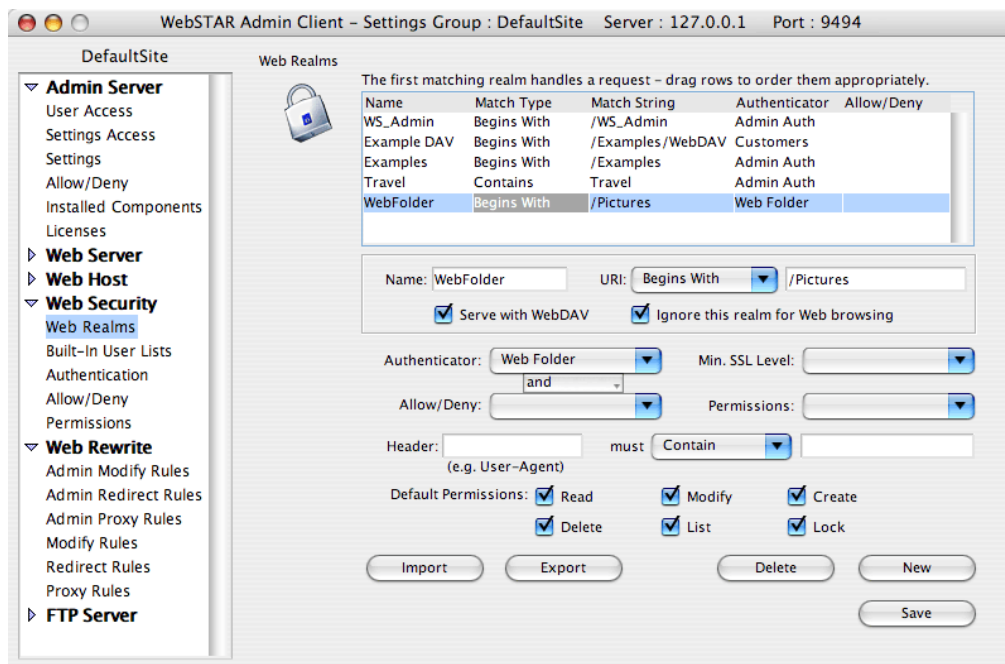


Figure 7.2 The Pictures folder within a Virtual Host folder as a WebDAV folder

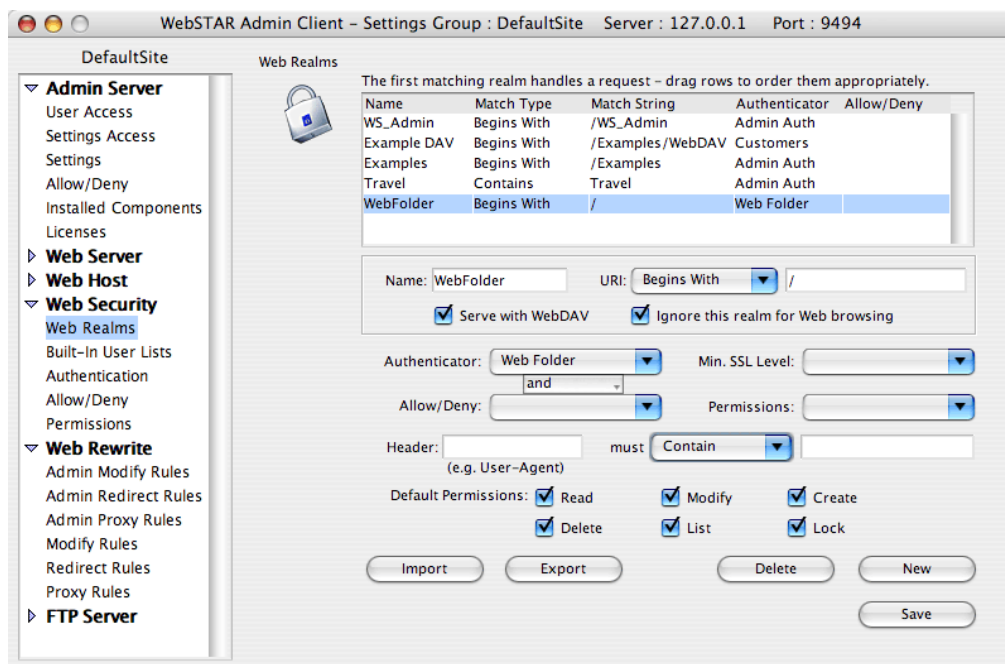


Figure 7.3 A realm configured to serve the Web Folder folder using WebDAV

7.3 Accessing the WebDAV Folder via the Internet

This section covers the process of accessing the *WebDAV* folder from Macintosh and Windows computers.

Macintosh OS X

With *Macs OS X*, you can access the *WebDAV* folder directly from the *Finder*. Also, a version of the *Goliath* application is being developed for *Mac OS X*. Use the instructions for *Mac OS "classic"* for the *OS X* version of *Goliath*.

To access a *WebDAV* folder from the *Mac OS X Finder*:

1. Choose *Go → Connect to Server*.

The connect dialog box appears.

2. Enter a request that includes the contents of the *Match String*.

You will connect to your server through WebDAV.

If all is well, the following dialog box will appear:

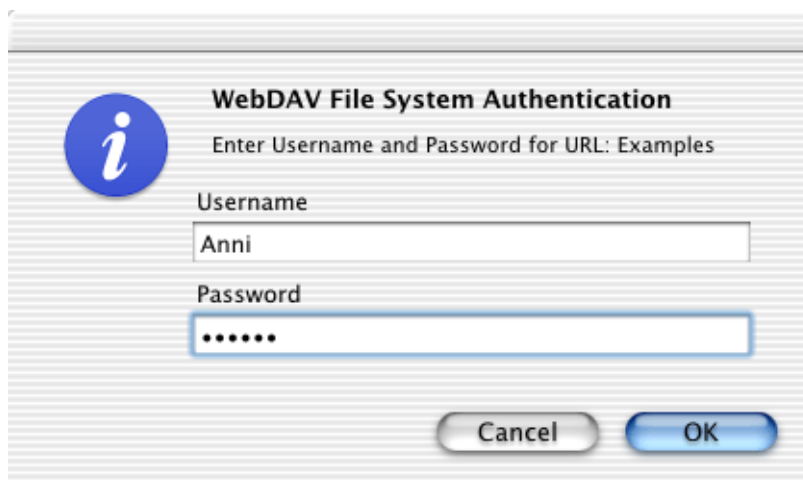


Figure 7.4 The Mac OS X WebDAV Authentication dialog box

3. Enter a username and password from the *User List*, as shown in figure 7.4, and click *OK*.

The *Macintosh OS* will then mount the *Web Folder* as a volume on your desktop. You can access the files according to the permissions set for that user.

Macintosh Classic

If a Macintosh client uses a previous version of the Mac OS, you must use a WebDAV client application such as Goliath (“DAV and Goliath” — get it?). You can download a copy of *Goliath* from the WebDAV site: www.webdav.org/goliath/

Both *Mac OS “classic”* and *Mac OS X* versions of *Goliath* are available.

To access a WebDAV folder using *Goliath*:

1. Double-click the *Goliath* application.

A dialog box appears in which you enter both the URL and your username and password.



Figure 7.5 The Goliath WebDAV connection dialog box

2. Enter the domain or IP address of the web folder, the port (if not 80), and your username and password.
3. Click *OK*.

If all goes well, *Goliath* will open a window on your desktop for the *Web Folder*.

Windows OS

A number of Windows applications can serve as WebDAV clients, notably *Microsoft Internet Explorer 5.0* (and above), *Word 2000*, *Excel 2000*, and *PowerPoint 2000*. *Internet Explorer*, for example, can open the *Web Folder* in a desktop window.

To access a *Web Folder* using *Internet Explorer*:

1. From *Internet Explorer*, choose *File* → *Open*.

A dialog box appears in which you can enter a URL.

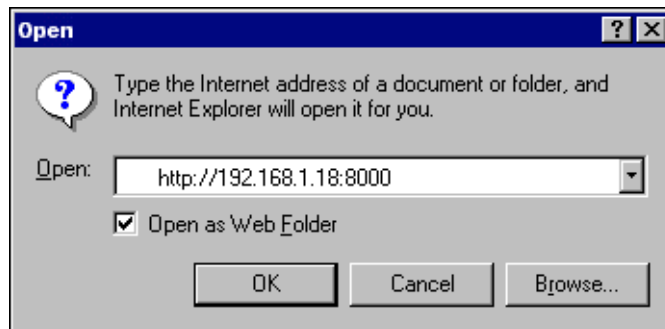


Figure 7.6 The Internet Explorer Open dialog box

2. Enter the domain or IP address and (optionally) port (e.g., `http://123.123.123.123:8000`), select the Open as *Web Folder* option, and click *OK*.

If all goes well, *Internet Explorer* will display a version of the *Authentication* dialog shown in figure 7.4.

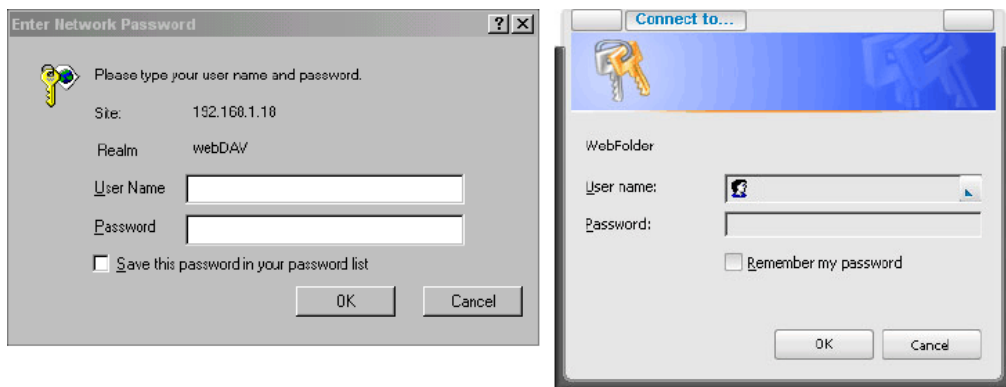


Figure 7.7 Windows and Windows XP WebDAV Authentication dialog boxes

3. Enter a username and password from the user list and click *OK*.

Internet Explorer will then open a new window showing the contents of the *WebDAV* folder.

Chapter 8

Kerio WebSTAR Rewrite

As a web server, *Kerio WebSTAR* receives requests from clients, evaluates them, and then serves the requested information. To make this process more flexible and adapt it to the special needs of webmasters and users, the *WebSTAR Rewrite* plug-in makes it possible to alter a request before it is passed on to the server.

WebSTAR Rewrite does its work on incoming requests. Simplified, you can see think of the process like this:



Figure 8.1 WebSTAR Rewrite allows you to do sophisticated search and replace operations on requests before the web server sees them

In this discussion, a request refers to an HTTP header that was sent from a client to *Kerio WebSTAR*, including many of its connectionrelated-related parameters.

8.1 Basic Uses

WebSTAR Rewrite supports three different ways of altering a request. They are:

- *Modify* — The client sends a request but *Kerio WebSTAR* actually receives a different request.
- *Redirect* — A *Redirect* simply redirects the client to a different location. It does this by sending a “302 Moved” message back to the client.
- *Proxy* — *WebSTAR Rewrite* constructs a new request that is sent out to the final destination. The result of that request will be returned as the result for the original request.

Technically, only the modify mode alters the request. The redirect and proxy modes actually alter the request so that later *WebSTAR Rewrite* gets complete control over the request and can process it completely. That way, *WebSTAR Rewrite* is able to read the request completely (i.e., POST arguments) and also send back data to the client — which is not directly possible while in the altering process:

When *WebSTAR Rewrite* is not involved in the process, it works like this:



Figure 8.2 Scheme

The action is determined by the suffix found in the filename and the entry in the *Suffix Mapping* table.

When *WebSTAR Rewrite* handles the modified request, it works like this:

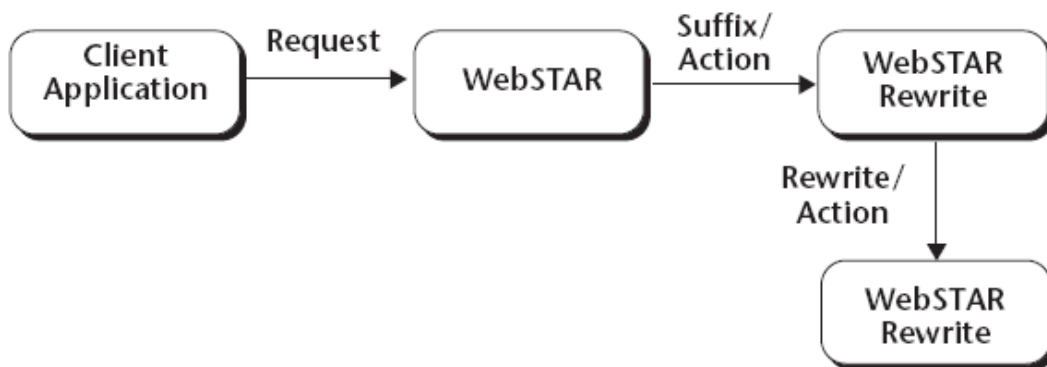


Figure 8.3 Scheme

8.2 WebSTAR Rewrite Modes

WebSTAR Rewrite offers three different ways of altering a request. Here are descriptions of these three modes.

Modify

Modify is a “silent” mode. The client (usually a browser) sends a request but the web server actually receives a different request. This is not directly visible to the client except in the case of relative links.

Since clients typically maintain their own view of the current directory and since requests to servers are always absolute (in the form of GET /foo/bar...) the client has to calculate the absolute path using the relative path together with their view of the current directory. The current directory is the directory just before the last slash in the address field in the browser — just before the name of the requested document.

When the request is changed by *WebSTAR Rewrite* in a way that changes the base directory of the requested file, then all relative links in that file may fail because the client will request them not relative to the file but relative to the location of where the client thinks the file is.

Example

Suppose the browser sends the following request:

`http://www.host.com/foo/index.html`

The original request is GET `/foo/index.html`. This means that the browser thinks all links are relative to the `foo` directory.

Suppose *WebSTAR Rewrite* changes the request to GET `/bar`.

Kerio WebSTAR returns the page `/bar/index.html`.

This page contains a relative link. The Browser requests:

`http://www.host.com/foo/hello.gif`

WebSTAR doesn't find `/foo/hello.gif`.

There are two ways of handling this:

Code links in affected pages in an absolute manner.

or

Alter not only the HTML page but also all affected files (i.e. images or other pages).

Typical uses

A modify can shield your directory structure from the public. If the modification is based on specific parameters of the request (browser name, client IP, etc.) then this can be used as a kind of protection without actually denying the request. For example, it is possible to have a simple hijacking protection by altering image requests of any form if their Referrer value does not match the current site. Altering would then point the request to a global gif (or jpg) that shows a "Hijacked image"! content (or advertisement) instead of a broken image link.

Modify can also patch broken links or directory changes. For example, it is possible to change `.htm` requests into `.html` requests.

Another possible use of modify is for multi-language sites. If, for every `index.html`, page there are also `index-de.html` and `index-en.html`, then a modify could change the name of the index file depending on the language settings of the browser. Alternatively, the whole site could be organized into separate subfolders for each language. For example, `/site/en` and `/site/de`.

Redirect

A Redirect simply redirects the client to a different location. It does this by sending a “302 Moved” message back to the client. The big advantage of the redirect mode is that it is dynamic and does not require you to modify the contents of the *Virtual Host* folder. The URL that is to be redirected does not need to exist in order to use redirect.

This means that not only specific URLs (requests) can be redirected but also parts of the site or even the whole site. The redirect destination can be dynamic and therefore different for each case, thus allowing you to keep certain parts of the URL in the redirected path.

Example

A redirect from folder /foo to folder /bar without losing the remaining path:

The Browser requests:

`http://www.host.com/foo/some/thing?here`

Redirect to:

`http://www.host.com/bar/some/thing?here`

See the section [9.7](#) for information on how to do this.

Proxy

The name proxy refers to the common concept of a man-in-the-middle. For a proxied request, *WebSTAR Rewrite* constructs a new request that is sent out to the final destination. The result of that request will be returned as the result for the original request. The client that requested the original request will not notice that the result is not from WebSTAR except for any links that either are absolute (containing the name of the final destination) or don’t make sense in combination with the original request.

Currently, *WebSTAR Rewrite* does not alter the result of the remote proxy call. The content is passed unaltered except for any connectionbased parameters like chunk-encoding or content length — those are recalculated by *WebSTAR* when the result is sent back to the client.

Proxy requests can, however, change certain parameters of the request before it is forwarded to the destination. First, the host name is adapted to match the destination and it is possible to specify a different port number and authentication parameters for the new request. This comes very handy when doing a local proxy call to a server (for example, Kerio) that runs on a different port than the original request. Also very handy is the ability to alter the authorization username/password by removing it from the request, altering it, or, adding it to the request.

The last option (adding) is very useful to make a certain part of a password protected area publicly available without revealing the authentication parameters.

In general, the Proxy method allows integration of external resources without revealing that fact. But since the result cannot be altered, some degree of control of the destination is necessary to make it work smoothly. In the future, more sophisticated result filtering will be available to reformat links and extract only certain parts of the content (in combination with SSI).

8.3 Implementation Details

Kerio WebSTAR 5 is administered at both the server-wide (Admin) and virtual host-specific levels. Rewrite actions that are implemented at the admin level apply to all virtual hosts. This means that in addition to the master administrator, there can be a number of virtual hostbased sub-administrators.

Since rewriting a request can be very powerful and can potentially mess up your web service, *WebSTAR Rewrite* uses a multi-level rewriting system.

Order of Execution

For each request, all defined actions (called rules) are run in the following order:

1. Original Request
2. Administrator Rules (Preprocess)
3. Virtual Host Modify Rules
4. Administrator Rules (Postprocess)
5. Final Request

Administrator rules can be set to run either before or after virtual host level rules (or both).

Administrator rules have the highest priority because they can alter a request both before and after the virtual host rules. There is only one set of administrator rules for each server and there is only one set of virtual host rules for each virtual host (determined by the current request). However, some administrator rules can run prior to the virtual host's rules, while other administrator rules can run after the virtual host's rules.

The set of virtual host rules can be administrated by logging into the admin client and editing the virtual host with the corresponding name. The "master" set of administrator rules can be edited by logging into the admin client as administrator and choosing either admin or any virtual host.

WebSTAR Rewrite Interface in Admin Client

In the admin client application, the *WebSTAR Rewrite* panels are organized into their own group in the browser, between the *Web Server* and *FTP Server* groups. There are separate panels for the modify, redirect, and proxy modes. And, there are separate sets of panels for the admin level rules and the virtual host-specific rules.

If you connect to a virtual host with full admin privileges, you see all panels — six panels. If you log on to admin, you see only the three admin panels. And, if you log in with host admin privileges, you see only the three panels for that virtual host.

Figure 8.4 shows each of these possibilities in the admin client browser.

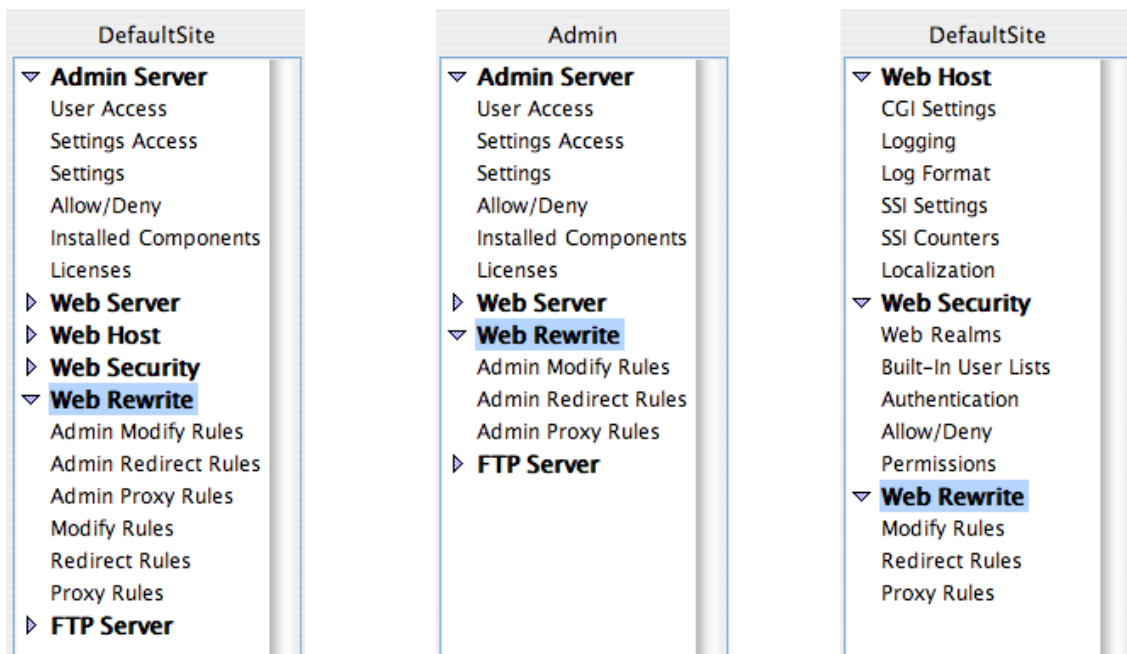


Figure 8.4 Browser panels for Full Admin and Host Admin administrators

Within each set of rules, the three different kinds of rules are executed in the following order:

1. All Modify Rules
2. All Redirect Rules
3. All Proxy Rules

The rule types that actually send something back to the client (Redirect and Proxy) will stop rule execution unconditionally when they are run. In contrast, modify rules change the current path only so that any rules that follow will see and manipulate the changed path. In the case of modify, both admin and virtual host rules can alter the same incoming request — in the order shown in *Order of Execution* (section 8.3).

8.4 Conditional Rules

With so many rules and rule types, *WebSTAR Rewrite* allows control of when a particular rule is executed. Each rule should define a condition that controls whether this rule is executed for a given request. That is, each rule is executed only if an If statement evaluates to true. In a *WebSTAR Rewrite* panel, you enter the If statement in the *Condition* block below the *Rules* table. You can enter a simple If statement or a compound If statement, with up to three clauses connected by and or or.

In the admin client interface, conditions are represented as three identical rows containing pop-up menus and a text field. Each row defines a single condition and they can be combined into a compound condition by using the and and or operators at the end of each row.

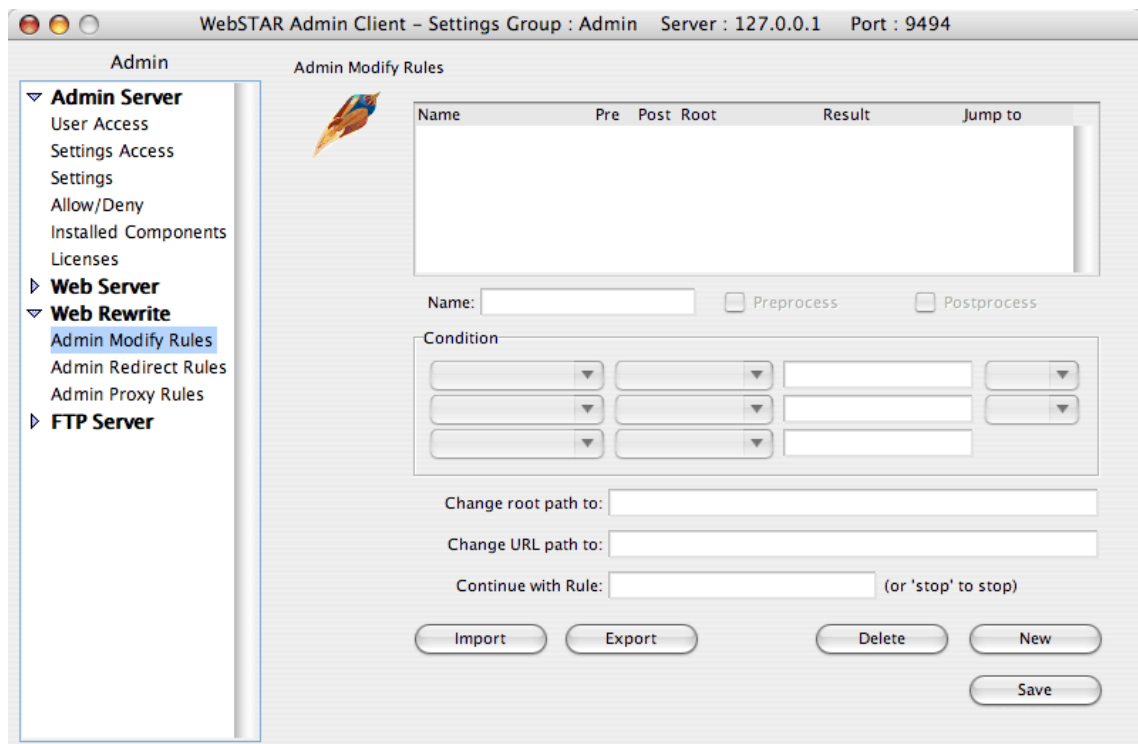


Figure 8.5 The Admin Modify Rules panel

Each row in the *Condition* block contains a *Subject* field (the first popup), a *Comparison Type* (second pop-up) and a *Match* value.

Condition		
Path	contains	/Examples/

Figure 8.6 A Condition block showing one condition

A simple condition looks like figure 8.6; it is one If statement. This condition intercepts requests that contain the string `/Examples/`.

Choosing the Subject pop-up makes *WebSTAR Rewrite* fetch the chosen parameter from the current request and uses its value in the comparison. The match value will be compared against that value and the comparison popup defines the way the comparison is evaluated.

If the condition matches, the string you enter in the *Change URL path to* will be used.

For example, figure 8.6 instructs rewrite to examine the Path and determine whether the string `/Examples/` is within it.

If you need to add another condition, choose a conjunction from the pop-up to the right of the *Match Value* area and enter the second condition in the second row in the *Condition* block.

Condition		
Path	contains	/Examples/

and

Figure 8.7 Adding another condition to the Condition block

WebSTAR Rewrite tests all rules in the order given above and, if a rule condition matches, it executes the rule. If you write a compound condition, the entire expression must evaluate to true in order for the rule to be executed.

Subject Parameters

Here's a list of all subject parameters and some example values. The example values are given assuming original request is:

`http://host.com/foo/bar.html?args#p3`

Note that the part after the pound sign in a URL is never sent to the server. It is used only by the browser to jump to the part of the page that has the local label with the corresponding name.

- *Path* — This reflects the path portion in the current request. This is the string from the first / after the host name up to the end or the first ?, \$, or # in the URL (whatever comes first). In this example, it is:

`/foo/bar.html`

- *Original Path* — Same format as *Path* but it refers to the unaltered value. In this example, it is:

`/foo/bar.html`

- *Search Args* — Anything after the ? in the URL. If there is no ? then it is empty. In this example, it is:

`args`

- *Host* — The host name that was used to access the current site. In this example, it is:
`host.com`

- *Site* — The name of the virtual host (User interface name that you choose in the virtual hosts panel for the site being served). For example, it could be *DefaultSite*. This is mainly used in admin rules so the administrator can constrain rules to certain sites without knowing all possible host names that can access this site.

- *Browser* — The user-agent string that the client sends. For example, *Mozilla 4.72*.

- *Language* — Typically a list of language tokens that the user defined in his browser. Usually a comma+space separated list of 2 to 4 character tokens, An example list would be `de, en-US, sv`.

- *Referrer* — The referring page for the current request. Please note that this value can be faked so it is not safe to build security related rules on this.

- *Full Request* — The full HTTP header. It starts with, for example:

`GET /foo/bar.html?args HTTP/1.1\r\n...`

and usually contains several lines.

- *Client IP* — The IP address of the client, in dot notation. For example:
192.168.72.18
- *Server IP* — The IP of the server (if the server has more than one IP, then it's the IP that the current request arrived on). In dot notation.
- *Server Port* — The port number on that the current request arrived on. Normally this is port 80.
- *Username* — If the browser is sending authorization data, then this is the username that was used to authenticate the user. It cannot contain a :.
- *Password* — If the browser is sending authorization data, then this is the password that was used to authenticate the user. It cannot contain a :.
- *SSL Bits* — The number of bits that describe the encryption level of the current request. It is 0 for ordinary http requests and a positive number (i.e. 40 or 128) for https requests.
- *URL* — The combination of the path and the search arguments. Technically the string that is between the GET and the HTTP/1.x in the HTTP header. In this example, it is:
/bar.html?args

Comparison Types

The *Comparison Types* pop-up contains several simple comparisons and one rather complex comparison type — the matches condition.

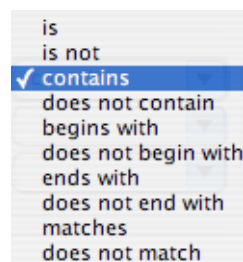


Figure 8.8 The Comparison Types pop-up menu

The conditions are as follows.

- *Is* — the specified string is exactly the same as the expression being searched.
- *Is not* — the specified string is not exactly the same as the expression being searched.

- *Contains* — the specified string is somewhere within the expression being searched.
- *Does not contain* — the specified string is not anywhere within the expression being searched.
- *Begins with* — the expression being searched begins with the specified string.
- *Does not begin with* — the expression being searched does not begin with the specified string.
- *Ends with* — the expression being searched ends with the specified string.
- *Does not end with* — the expression being searched does not end with the specified string.
- *Matches* — The matches option indicates that the match value is a regular expression. This is a very powerful option that allows you to perform very sophisticated search-and-replace operations. See chapter 9 for information on how to use regular expressions.
- *Does not Match* — The specified string is a regular expression, but the expression being searched does not match it. See the above reference about using regular expressions.

The Rule Result

You use the field below the *Condition* block to specify what the rule will do if the conditions are met — the If statements in the *Condition* block evaluate to True. It is labelled differently for each rule type. For modify rules, it is called *Change URL path to:*. For redirect rules, it is called *Redirect to:*, and for proxy rules, it is called *Proxy to:*.

Its function, however, is the same across rules — only the format changes a bit. Keep in mind that this field can be empty for any rule type. This will make the rule not do anything. This is useful in combination with the *Continue with Rule* field, which causes the rule you specify to execute next.

- *Modify* — For modify rules, the field will change the path portion of the current request. The path is anchored to the current virtual host's root directory and you cannot leave this directory regardless what the result of the rule will be. The path will be stripped from any */..* or */.* substrings and it should (must) start with a */* or else the server will reject the request as a malformed request.
- *Redirect* — For redirect rules, the field will determine what will be sent back to the client after the *Location:* tag of the response. Valid strings are absolute URLs

like `http://www.foo.com/bar.html` or relative URLs like `/myfolder/index.html`. Keep in mind that some browsers and spiders only accept absolute URLs.

- *Proxy* — For proxy rules, the field must contain a fully qualified URI. That is, it must start with `http://` and continue with a host name or IP number followed by at least a `/` or longer path containing optional search arguments. The host name can be prepended with an optional `username:password` string (like in `http://user:pass@hostname.com`) and the string before the `@` can be empty to strip the authorization parameters from the current request.

The Change Root Path To Field

On the *Admin Modify Rules* panel only, there is an additional field located directly below the *Condition Block* labelled *Change Root Path to*.

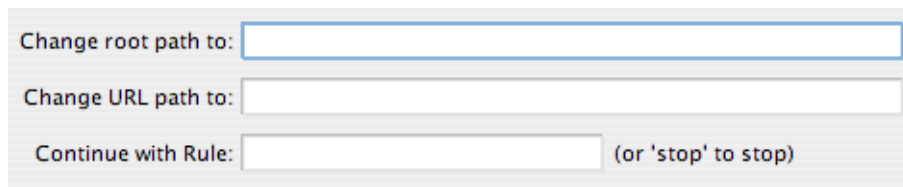


Figure 8.9 The Change Root Path field

It is an optional field that allows you to designate a different root folder, i.e., a different virtual host folder. The easiest way to do so is enter the relative path from the web server — which is the name of a virtual host folder in the *WebServer* folder, such as *DefaultSite*. You can also use an absolute Unix path (starting with a slash and the volume name and ending with the folder name).

The purpose of this field is to make it easy do virtual host routing with *WebSTAR Rewrite* rather than from the *Routing* table in the *Web Virtual Hosts* panel. See the section 5.2 for information on the concept of routing requests to different virtual hosts.

Flow of Rule Testing

Each panel allows you to name and save more than one rule. If you create more than one rule, you have several ways of controlling the order in which the rules are executed. By default, they are executed one after each other, in the order that they appear in the *Rules* table. To change the order, drag a rule up or down in the *Rules* table.

To stop the execution of rules before *WebSTAR Rewrite* reaches the end of the list, enter *stop* in the *Continue with Rule* area.

Sometimes it is useful to skip over some rules or jump back to a previous rule to create a loop in the rule execution. This is also done with the *Continue with Rule* field. If you

want to jump to another rule, enter its name in this field. Execution will continue until the last rule is executed or a stop is encountered.

A jump is executed after the rule has done its job (redirect and proxy rules stop execution anyway, so this applies only to modify rules). The jump, like the rule function, is tied to the condition and you can leave the rule action empty to have it only jump. Three possible values are allowed:

- A name of a rule within the same list as the current rule. This makes this rule jump to the named rule if the current condition is true. If no rule with the given name exists, no jump occurs and normal rule execution continues.
- The string *end*. This will stop execution of the current rule list and continue with rules of the next type of whatever comes next. The sequence is *Admin Preprocess* rules, virtual host modify rules, and *Admin PostProcess* rules.
- The string *stop*. This will force rule execution to stop but only for non-admin rules. Within virtual host rules, this only works as an *end*. This allows any administrator-level rules marked for postprocessing to run.

8.5 Creating a Rule

To create a rule, first determine whether the rule should be a modify, redirect, or proxy rule and whether it should apply to all virtual hosts or only a particular virtual host. If it is the former, you need to connect with full admin privileges and use one of the Rewrite's Admin panels; if it is the latter, you need to connect to that virtual host and use one of Rewrite's non-Admin panels (see figure 8.4).

To create a rule:

1. In the *Admin Client* application, click on the appropriate panel's name in the *Web Rewrite* group.

The panel you selected appears.

2. Click *New* to create the new rule.

A new blank row appears in the *Rules* table.

3. Enter a name in the *Name* area or click twice in the *Name* column in the new row to get an insertion point and enter the name.

If you later want to jump to this rule, you would use this name in the *Continue with Rule* field to execute this rule next in a sequence.

4. Use the *Condition block* to specify the condition under which this rule is to be executed.

See the section [8.4](#) for details on how to specify a condition.

5. (Optional) If you are creating a compound condition, choose *and* or *or* from the conjunction pop-up menu (figure [8.7](#)) and enter the next condition in the next row.

The entire (compound) expression you enter must evaluate to `true` in order for this rule to be executed.

6. Enter the URL or regular expression that evaluates to a URL in the *Rule Result* area (labelled differently depending on the mode you are using) or, if you want the rule to only select another rule to jump to, enter the name of the rule to execute next in the *Continue with Rule* field.

See the section [8.4](#) for more information on the requirements for this field.

7. (Optional) If you are creating an admin modify rule and want to route the request to another virtual host, enter the path to the virtual host folder in the *Change root path to* field.

If the virtual host folder is in the *WebServer* folder, you can enter its name.

8. (Optional) If you entered an expression in the *Rule Result* area and want to jump to another rule, enter it in the *Continue with Rule* field, enter `stop` to halt execution of the current rule list, or enter `end`.

To add additional rules of the same type (modify, redirect, or proxy) click the *New* button and repeat this process.

To add additional rules of another type or at a different level (*Admin* or *Virtual Host*), click the appropriate panel name in the browser area of the *Admin Client* screen and repeat this process.

Chapter 9

Regular Expression Reference

Much of the power and flexibility of *WebSTAR Rewrite* is due to the fact that it supports search and replace operations using regular expressions. Regular expressions can also be used in a few other places in *Kerio WebSTAR*, including realm definitions and CGI settings.

In order to handle more complicated search conditions, *Kerio WebSTAR* uses PCRE (Perl Compatible Regular Expressions). They emerged as a standard when the Perl computer language became popular. Regular expressions take some getting used to, but the effort is well worth it.

Regular expressions use a cryptic but sophisticated syntax that allows you to do much more complex and useful types of search and replace operations than you can do via the sorts of find and replace features of an ordinary word processor.

9.1 Regular Expressions Defined

Regular expressions extend normal search and replace with pattern matching. In search and replace within most word processors, you can match only the literal (exact) string that you specify.

Regular expressions also allow you to break up the results of your search into pre-defined “pieces” so that you can replace only a portion of the matched string and retain the remaining portion of the search result in your replace string.

First of all, any literal string of characters matches itself. For example, if you are looking for the specific string `.com`, you can enter it as a regular expression just as in any *Find* dialog box in a word processor.

Some special characters are used to match a class of characters. A class is a group of characters that share a characteristic, such as numbers, characters that end a line (such as a return or linefeed), or alphabetic characters.

9.2 Wildcards

Wildcard	Matches
.	Any single character except a line break (i.e., a Return), including a space.

Table 9.1 Wildcard for character matches

If you use the

.

 as the search pattern, you will select the first character in the target string. You can use the wildcard character with repetition characters (see 9.4) to find more than one character.

For example, if the text being searched is the text `WebSTAR`, the . will initially match the `W`. If you repeat the search, it will match the `e`, and so forth.

The following wildcards match by position in a line:

Wildcard	Matches	Example
^	Beginning of a line (unless used in a character class; see the next section).	<code>^Phone</code> Finds lines that begin with <code>Phone</code>
\$	End of a line (unless used in a character class)	<code>\$</code> Finds the last character in the current line.

Table 9.2 Wildcards for matching by position

9.3 Character Classes

A character class allows you to specify a set or range of characters based on their characteristics. You can choose to either match or ignore the character class. The set of characters that define the class is enclosed in brackets. If you want to ignore the character class instead of match it, precede it by a caret (^). Here are some examples:

Character Class	Matches
[aeiou]	Any one of the characters a, e, i, o, u.
[^/]	Any character except /.
[a-e]	Any character in the range a-e, inclusive.
[a-zA-Z0-9]	Any alphanumeric character.
[[]	Finds a [.
[]]	Finds a]. To find a closing bracket, place it immediately after the opening bracket.
[a-e^]	Finds a character in the range a-e or the caret character. To find the caret character, place it anywhere except as the first character after the opening bracket.
[a-c-]	Finds a character in the range a-c or the - sign. To match a -, place it at the beginning or end of the group of characters in the brackets.

Table 9.3 Example character classes

Non-printing Characters

You can use the following patterns to find non-printing characters:

Special Character	Matches
\r	Line break (Return).
\n	Newline (line feed).
\t	Tab.
\f	Formfeed (page break).
\a	Alarm (Hex 07).
\cX	The control character X (e.g., \cV matches Control-V).
\b	Backspace character (Hex 08).
\e	Escape character (Hex 18).

Table 9.4 Non-printing Characters

Other Special Characters

The following patterns are patterns for the following special characters:

Special Character	Matches
\\	A backslash.
\.	A period. A . that is not preceded by the backslash matches any one character, as explained in the <i>Wildcards</i> section.
\s	Any whitespace character (Space, Tab, Return, Linefeed, Form feed)
\S	Any non-whitespace character.
\w	Any word character (a-z, A-Z, 0-9, and _)
\W	Any non-word character (All characters not included by \w).
\d	Any digit character (0-9).
\D	Any non-digit character.
\xNN	Hex code NN.
\x{NNNN}	Any number of Hex characters, each is NN.

Table 9.5 Miscellaneous Special Characters

9.4 Repetition Characters

Repetition characters are modifiers that allow you to repeat a specified pattern.

Repetition Character	Matches	Examples
*	Zero or more characters.	<code>w*</code> finds no characters, or one or more consecutive <code>w</code>'s. <code>.*</code> finds an entire line of text, up to but not including the Return character. If the target string is <code>www.webstar.com</code>, then <code>w*</code> finds <code>www</code>. The next search finds the <code>w</code> in <code>webstar</code>, and subsequent searches find zero instances of <code>w</code>.
+	One or more characters.	<code>w+</code> finds one or more consecutive <code>w</code>'s. If the target string is <code>www.webstar.com</code>, then <code>w+</code> finds <code>www</code>. The next search finds the <code>w</code> in <code>webstar</code> and subsequent searches fail to find any characters. The pattern <code>[0-9]+</code> finds a string of one or more consecutive numbers, such as 90404, 1938, the 32 in Win32, etc.
?	Zero or one characters.	<code>w?</code> finds no characters or one <code>w</code>. The search does not fail if there are no <code>w</code>'s in the target string. Use it to match a character that is optional. For example, the pattern <code>\\$?[0-9,]+\.\d{2}?</code> matches amounts with or without the Dollar sign, the period, and cents. For example, it matches both 12,500 and \$12,500.50.

Table 9.6 Repetition characters

As shown in the examples, the pattern can be a literal string (such as `webstar`), a wild-card character (such as the `.`), a character class (such as `[0-9]`), or a special character.

For example, if the target string is `Windows2000`, the following patterns have these effects:

Repetition Character	Matches	Examples
{N}	Exactly N occurrences of the pattern.	w{3} matches www.
{Min,}	At least Min occurrences of the pattern.	w{2,} requires two or most consecutive w's for a match.
{Min,Max}	At least Min occurrences but no more than Max.	w{2,4} matches 2, 3 or 4 consecutive w's.

Table 9.7 Repetition characters — continuation

Pattern	Matches
[0-9]{4}	2000
\d{4}	2000
\w+	Windows2000
[a-zA-Z]+	Windows

Table 9.8 Examples of Repetition Characters

If the target string is `www.Kerio.com`, then the following patterns have these effects:

Pattern	Matches
[0-9]+ 4	
\d*	the 4 or returns no characters.
\w+	The www.
[a-z]+	The www.

Table 9.9 Examples of Repetition Characters

Matching Optional Characters

Note that the `*` and `?` have the ability to match zero instances of the specified pattern. Matching zero instances is different from failing to match anything.

Although it sounds silly to match zero characters, this feature does have an important use. You use this feature in a search pattern to match optional characters in the pattern. To take a simple example, suppose you need to match both positive and negative numbers. The negative sign may or may not be present. To handle this, you can use the pattern:

`-?[0-9]+`

Since the pattern `-?` matches zero occurrences of the minus sign, the entire pattern matches both positive and negative numbers. And the `+` sign ensures that at least one digit is present.

Greediness

The `*` operator is “greedy” (using the terminology of regular expressions) in the sense that it will try to match as many characters as possible, when there is any ambiguity.

Here is an example. Suppose the string being searched is:

AAA(BBB)CCC(DDD)EEE

and the pattern is:

`\(.*\)`

In English, the pattern specifies an open parenthesis, followed by any number of characters, followed by a closed parenthesis. Since there are two closing parentheses following the open parenthesis, the greedy option is to match as many characters as possible:

(BBB)CCC(DDD)

If you want to match only the (BBB), use the question mark after the `*` operator to turn off greediness. The pattern:

`\(..*?\)`

matches the (BBB) only.

9.5 Combining Patterns

Of course, much of the power of regular expressions comes from combining elementary patterns to make up compound patterns. Here are some examples.

Pattern	Matches
<code>\d+\sB\.C\.</code>	One or more digits followed by a space, followed by B.C.
<code>\(?:\d{3}\)?\s\d{3}-\d{4}</code>	US telephone number, with optional parentheses for the Area code (Optional open parenthesis, three digits, optional closed parenthesis, a space character, three digits, a dash, and four digits.)

Table 9.10 Examples of combining patterns to match complex strings

9.6 The Alternation Operator

The alternation operator `|` allows you to match any of a number of patterns using the logical or operator. Place it between two existing patterns to match either pattern. You can use more than one alternation operator in a pattern.

The last example uses parentheses to group complex patterns. See the section 9.7 for more information on how parentheses can be used.

Pattern	Matches
<code>\she\s \sshe\s</code>	he or she (spaces used to ensure whole word matches)
<code>cat dog possum</code>	cat, dog, or possum
<code>([0-9,]+\sB\.C\.) ([0-9,]+\sA\.D\.) or [0-9,]+\s((B\.C\.) (A\.D\.))</code>	Years of the form yearNum B.C. or A.D. e.g., 2,175 B.C. or 215 A.D.

Table 9.11 Examples of the use of the Alternation Operator

9.7 Working with the Matched Pattern

After you have found the desired string using a regular expression, you will want to replace it with a modified string. You might want to keep a part of the matched string and replace only a substring within that matched string. In other cases, you might want to append or prepend other text to the string. There are special regular expression patterns that you use only in the replacement pattern that make this easy.

The symbol `$'`, when used in the replacement pattern, represents the entire matched string. Use it, for example, to append or prepend text to the matched string.

If your search pattern defines subpatterns — see the following section, Subpatterns and Submatches for a description of subpatterns — you can use the special variables `$1`, `$2`, `$3`, and so forth, in the replacement pattern to refer to subpatterns within the matched string.

In *Rewrite*, this standard notation is extended so that you must indicate the row in the *Condition block* that contains the desired subpattern. The format is `$a.b` for the *a*th row and the *b*th subpattern from the left in that row.

This is a very powerful feature of regular expressions. Within *Rewrite*, this allows you to patch an incoming request and map it to the correct directory or document.

Table 9.12 summarizes the special replacement patterns.

Pattern	Description	Example
<code>\$a.b</code>	Refers to the <i>b</i> th subpattern in the <i>a</i> th row in the <i>Condition block</i> (<i>a</i> is 1, 2, or 3). Contains the matched subpatterns, defined by use of parentheses in the search pattern. See the following section.	The search pattern <code>(\d+)\s(B\.C\. A\.D\. BC AD)</code> looks for any number of digits followed by a space character, followed by either B.C., BC, A.D., or AD. The <code>\$1.1</code> variable contains the match to the <code>\d+</code> portion of the expression and <code>\$1.2</code> contains the match to the <code>B\.C\. A\.D\. BC AD</code> portion (assuming the first row in the condition block).
<code>\$a.0</code>	Refers to the entire matched value for the particular line (<i>a</i> is 1, 2, or 3). It returns a match only if the specified row uses the “matches” operator and the expression results in a match.	If <code>\d{4}\sB\.C\.</code> matches 1541 B.C. then the replacement pattern <code>the year \$1.0</code> results in the year 1541 B.C. (assuming the first row was used), as the <code>\$1.0</code> contains the entire match for that row.
<code>\$a</code>	A wildcard that matches full input string of the specified line (<i>a</i> is 1, 2, or 3). The line does not have to use the matches operator; it only needs to specify a valid field type such as Path, Host, or Referrer and <code>\$a</code> will contain the contents of that field type. The condition and match pattern can be left unspecified. That is, <code>\$a</code> will contain the value of the specified field unconditionally	If the first line specifies the host, then <code>\$1</code> contains the full text of the path; this makes it easy to substitute hosts: Line 1: Path matches <code>^/foo/(.*)</code> Line 2: Host (nothing else specified) The replacement pattern is: <code>/\$2/\$1.1</code> That is, <code>\$2</code> will contain the host and <code>\$1.1</code> will contain the text in the path after the last <code>/</code> . If the host is <code>www.webstar.com</code> , then it changes <code>/foo/index.html</code> to <code>/www.webstar.com/index.htm</code>

Table 9.12 Patterns used in Replacement Operations Only

Subpatterns and Submatches

You can use parentheses within your search patterns to isolate portions of the matched string. You do this when you need to refer to subsections of the matched pattern in your replacement string. For example, use this feature when you need to replace only a portion of the matched string and keep the remainder of the matched string as is.

Here is a simple example. Suppose the target string is `www.homefries.com`. The pattern:

```
([a-z]{3,}\.)(\w+\.)(\w+)
```

matches the string in the following way:

Pattern	Description	Matches
<code>[a-z]{3,}\.</code>	Three or more alphabetic characters, followed by a period.	<code>www.</code>
<code>\w+\.</code>	One or more word characters, followed by a period.	<code>homefries.</code>
<code>\w+</code>	One or more word characters (alphanumeric characters).	<code>com</code>

Table 9.13 Example of subpatterns and Submatches

You could also match the entire string without the parentheses, e.g.:

```
[a-z]{3,}\.\w+\.\w+(2)
```

The crucial difference is that the expression (2) matches the entire string — `www.homefries.com` — and stores it in the variable `$'`, while expression (1) also breaks this string up into three submatches. When doing a replace, you can use the variables `$1.1`, `$1.2`, and `$1.3` to refer to each matched substring. After the search, the variable `$1.1` contains `www.`, `$1.2` contains `homefries.`, and `$1.3` contains `com` (assuming that the pattern is in the first row of the *Comparison* block).

When you do your replace, you can retain parts of the matched string in the replace by using the variable names in the replacement pattern. and append or prepend other strings, or replace one substring with another. Without the parentheses, you can only append or prepend text to the entire matched string, or replace the entire matched string with another string.

Example

Suppose you want to change the first subdirectory of the URL path to `/foo` for a particular virtual host. You want to retain everything in the request after the first subdirectory.

You can use a modify rule with the matches option and use the following search pattern:

```
/[^\/]+(.*)
```

It's very cryptic, but the regular expression matches it piece by piece in the following way. Suppose the incoming request is `GET /bar/hello.gif`

Since the last term in table 9.14 is in parentheses, the variable `$1.1` contains the matched pattern, regardless of what the incoming request is.

Therefore, you can use the replacement pattern:

```
/foo$1.1
```

Subexpression	Description	Matches
/	Matches a literal /	/
[^/]+	The [^/] matches any single character that is not a /. The plus sign indicates that you want to match the string up to and not including the next /.	bar
(.*)	Zero or more characters, up to the end of the request. Matches the remainder of the request, including the leading /.	/hello.gif

Table 9.14 Matching the requested path using a subpattern

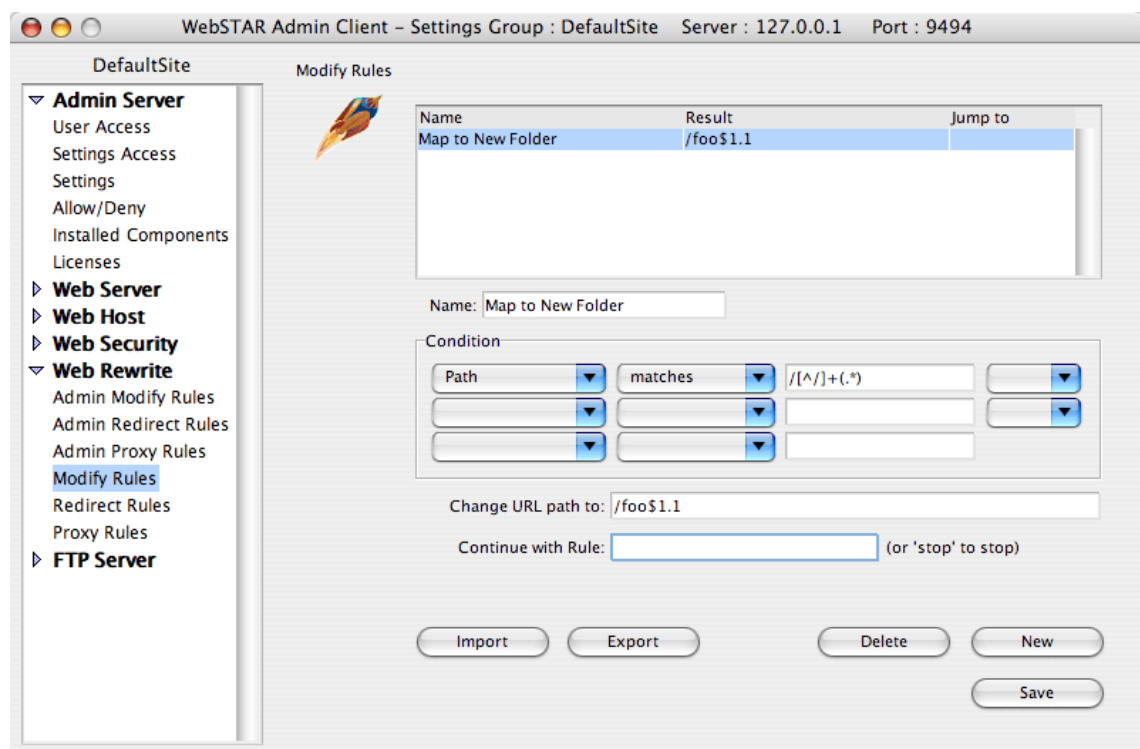


Figure 9.1 Using regular expressions to patch incoming URLs for a Virtual Host

This specification is shown in figure 9.1.

The power of regular expressions is obvious from this example. Since the variable \$1.1 contains the substring in the request after the name of the directory, this one rule can be used to handle all requests for any document in this virtual host folder. You don't have to hardcode rules for each specific document.

Chapter 10

Server-Wide and Virtual Host-Specific Settings

This chapter provides an overview of the panels in *Kerio WebSTAR 5 Admin Client* that can be set at the server-wide and virtual host levels.

10.1 Server-Wide Settings

The following aspects of *Kerio WebSTAR 5* are configured at a global level:

- Enter and set access privileges for *Kerio WebSTAR 5* administrators
- Create and configure virtual hosts
- Set the default document loaded when browsers log onto your sites
- View installed components (a.k.a., plug-ins)
- Set the locations of the monitored servers, the ports used to communicate with *Admin Client* and the servers, and other server parameters.
- Set various web connections and web caching parameters
- Configure encrypted connections using SSL
- Edit the suffix mapping table
- Set up a search index to support quick searches of your web site
- Set up FTP service
- Establish connections with *4th Dimension* databases to publish dynamic web sites

Admin User Access

Use to enter the names of persons who have access to the *Admin Server*. You assign user names, passwords and access permissions. For persons with host admin privileges, you indicate which virtual hosts the person can access and whether access is limited to read/only (view settings without the ability to modify settings) or read and write.

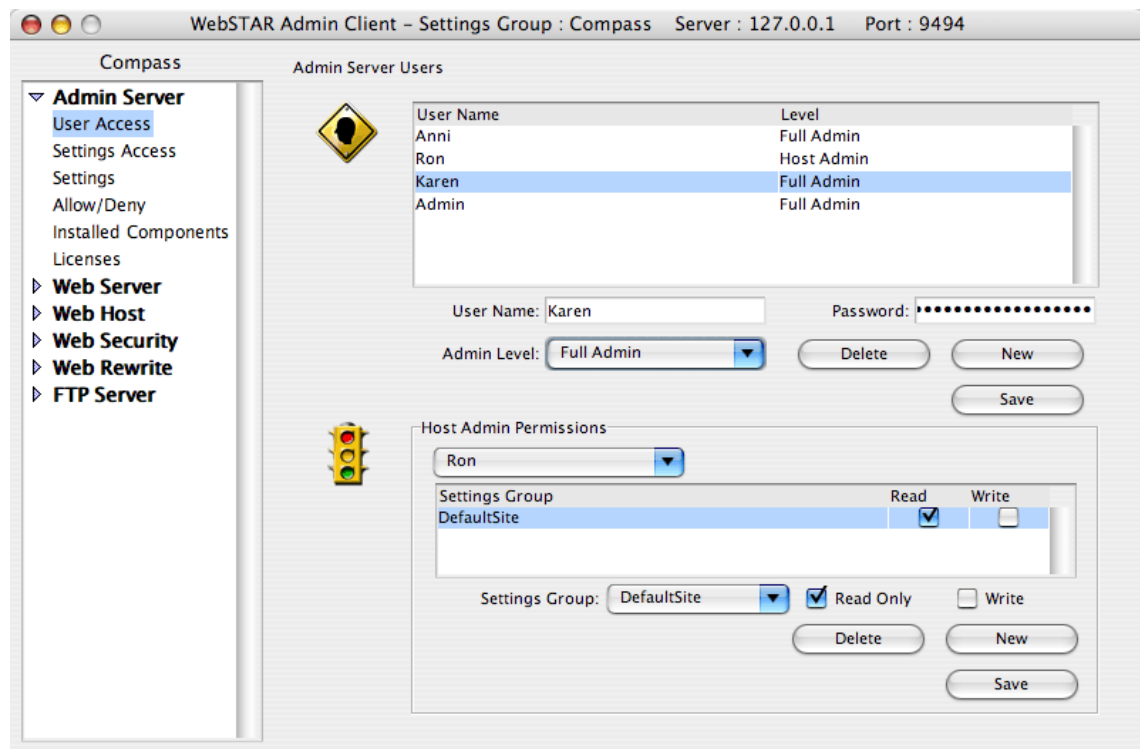


Figure 10.1 Admin User Access for one full admin and two host admins

In figure 10.1, the names of the two host admins appear in the *Host Admin Permissions* pop-up menu. Figure 10.1 shows that host admin Anni has read/write access only for the Compass site. Choosing Julie's name from this pop-up shows her access privileges in the table.

If *Kerio WebSTAR 5* will be administered by more than one person, you will need to modify this panel in order to grant each person access.

Settings Access

Use to specify permissions to the *CGI*, *Log*, and *Security* plug-ins for each virtual host.

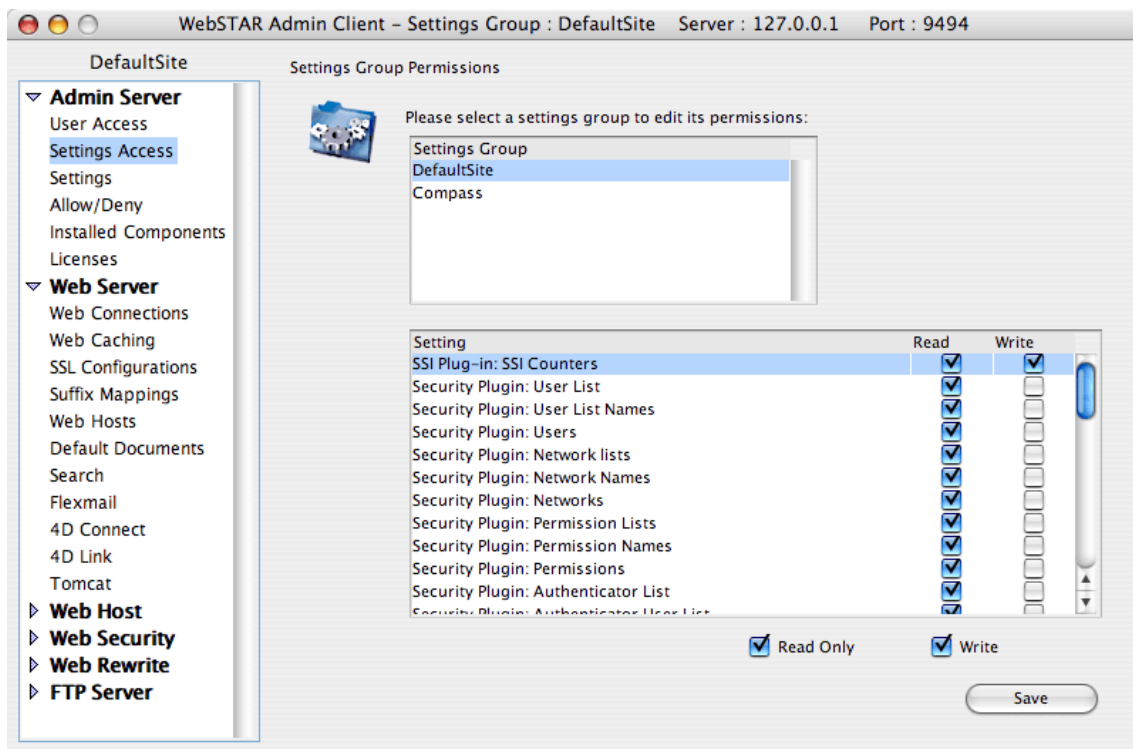


Figure 10.2 The Settings Access panel

Admin Server Settings

Use to enter the servers monitored by *Admin Client*, the port used by *Admin Client* to communicate with *Admin Server*, the port used by *Admin Server* to communicate with the servers, the maximum number of simultaneous *Admin Client* connections, and the maximum number of login failures (i.e., incorrect username/passwords). By default, the web server is entered in this list and default value for the other parameters are provided.

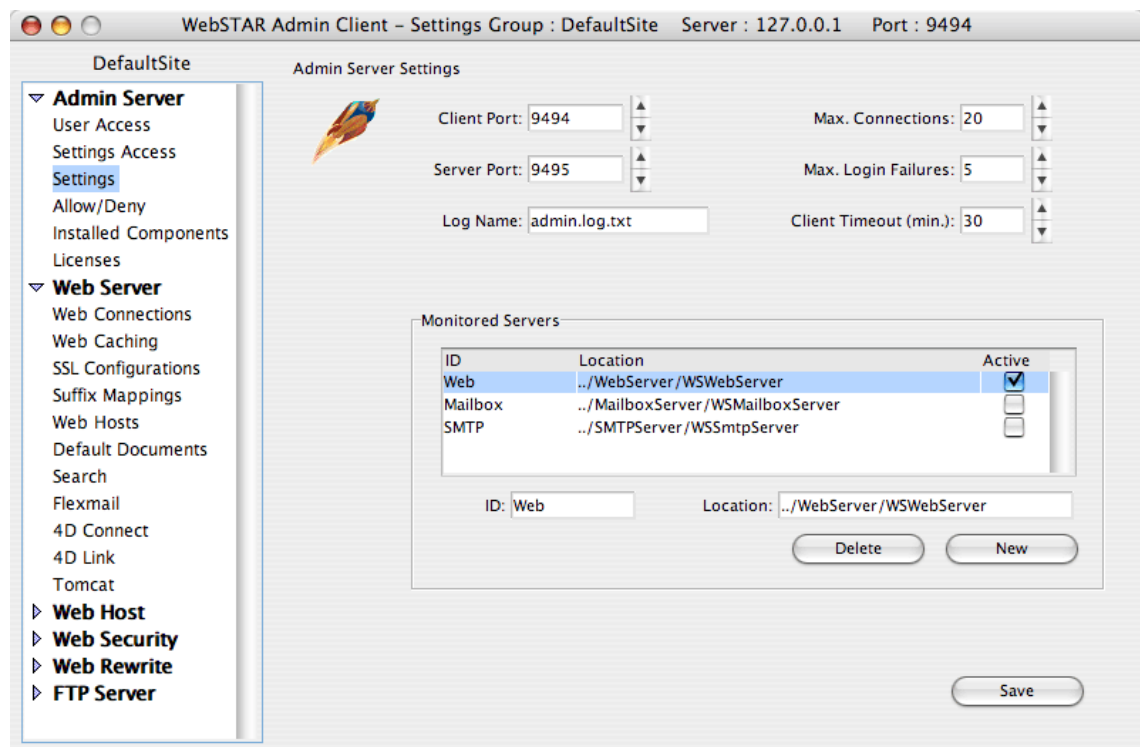


Figure 10.3 Admin Server Settings

Admin Network Access (a.k.a., Allow/Deny)

Use to enter the hostnames or IP addresses of client machines that are permitted (or not permitted) to access settings via *Admin Client*. By default, there are no entries, meaning that connections from any machine are permitted. If you specify client machines that are permitted access, all others are denied.

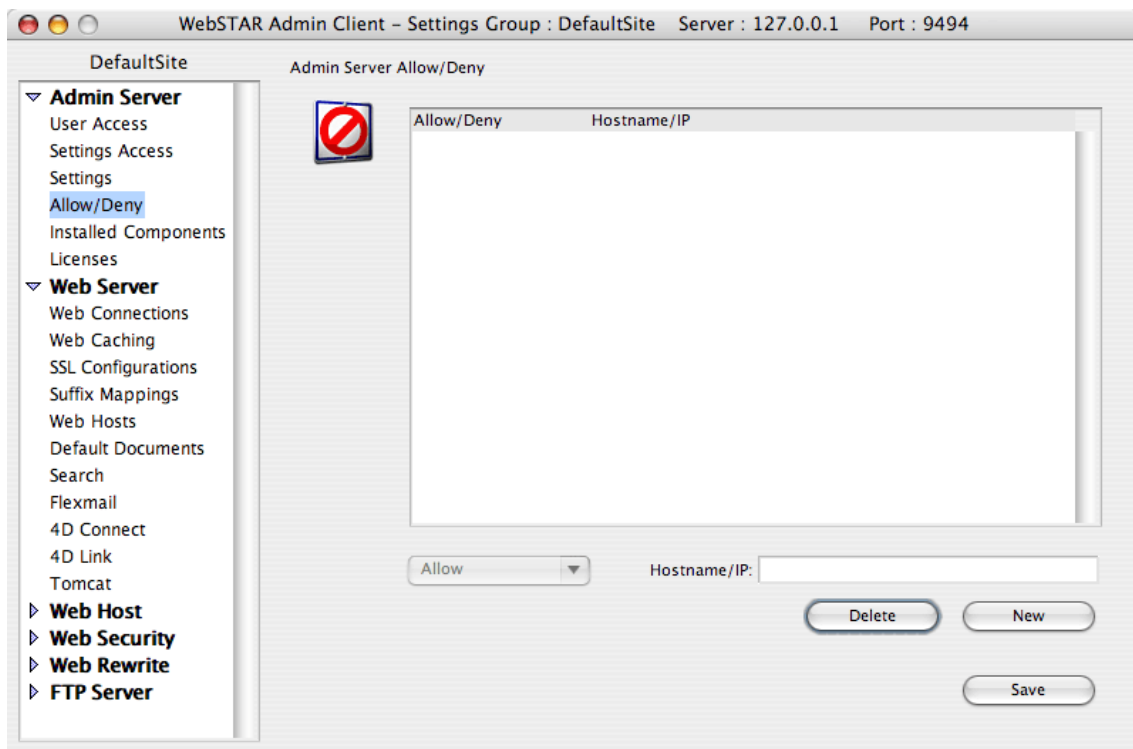


Figure 10.4 The Admin Network Access panel

Installed Components

Use to view the plug-ins that are currently installed and running on the web server. This list can be changed only by moving plug-ins in or out of the *Plug-ins* folder in the *WebServer* folder. You do not install or remove plug-ins via this panel.

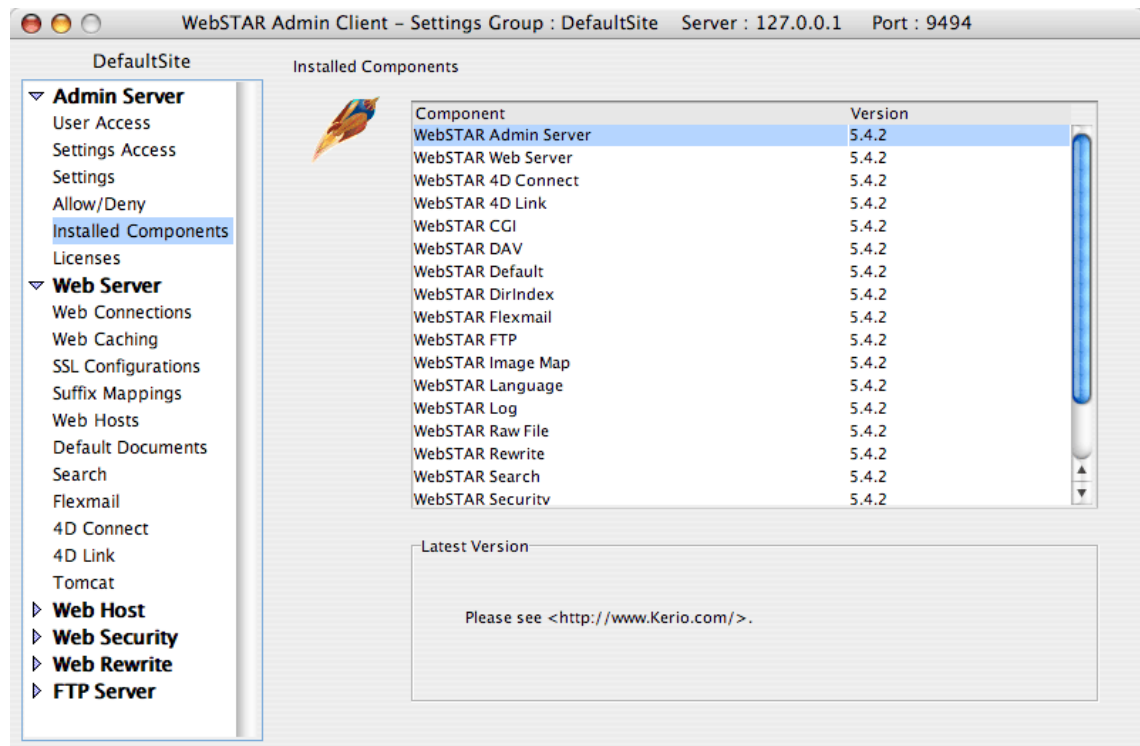


Figure 10.5 The Installed Components panel

Web Connections

Use to set the IP addresses that the web server is listening to, the maximum number of simultaneous connections, the timeout, the buffer size, and whether to use persistent connections. This page makes it easy to specify, for example, one IP address for normal connections and another IP address for secure connections.

By default, the *Web Listeners* table is configured to listen on ports 80 and 8000. Port 80 is the default port for http; if you listen on another port, the request must include the port number as part of the request. For example, to use port 8000, the request should be of the form:

`http://www.domain.com:8000/special.html`

If you configure your server for SSL secure connections (https), you need to add a row to this table which specifies the SSL configuration to use for encrypted requests. When

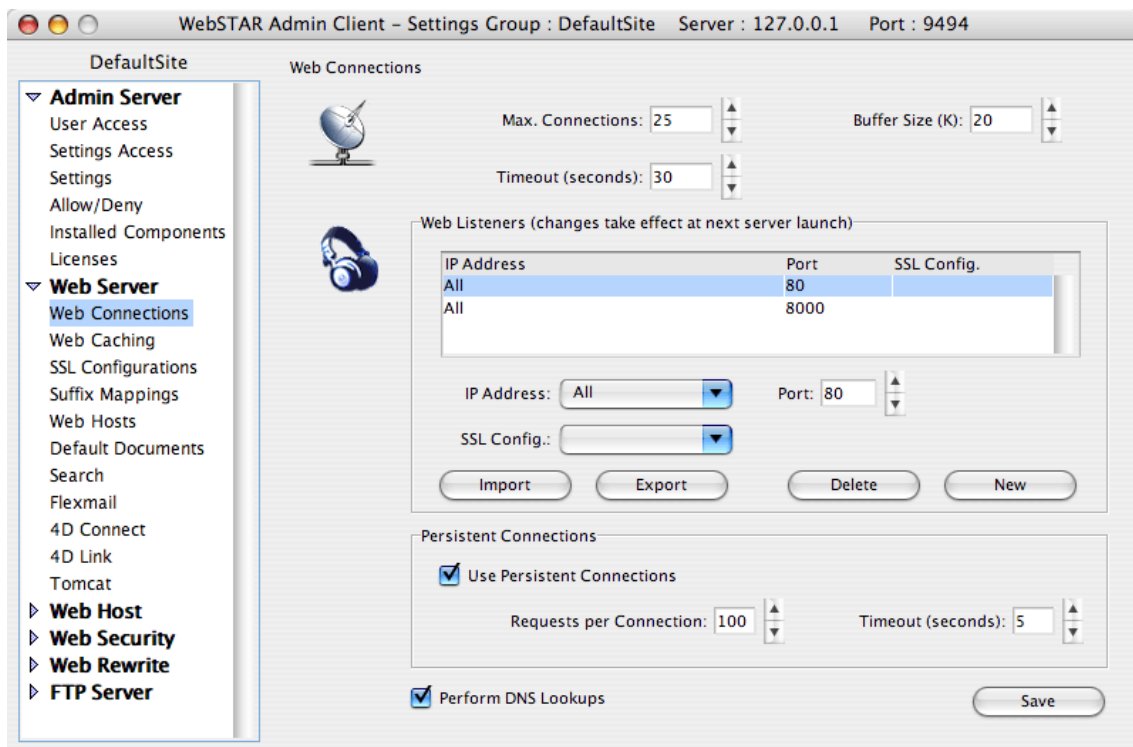


Figure 10.6 The Web Connections panel

you do so, you need to choose the *SSL Configuration* (set on the *SSL Connections* panel; see the section 16.7). Normally, you use port 443 for SSL connections.

The options in the *Web Connections* panel are as follows:

Max Connections

This parameter specifies the maximum number of concurrent HTTP client connections to the web server. Many browsers request more than one element of a page, such as images, at the same time, to speed up their display. By default, *Netscape* browsers and *Microsoft Internet Explorer* are set to four simultaneous connections for each request.

As soon as each file is entirely served, however, the browser disconnects itself. Therefore, the number of connections is not describing just the number of people trying to view your web site, but also the size and complexity of the web pages.

If a browser tries to access your site, and there are already too many connections, *Kerio WebSTAR* will send back a message indicating that the browser should try again in a few seconds. Many browsers automatically try again without displaying an error, but eventually, they may timeout and display a message indicating that the web site is busy. The *Web Activity Monitor* window shows you a *Web Connections* graph, which can help you track whether you have enough connections allocated. If people complain that your sites are always busy, you should probably increase the

number of Max Connections, if you have enough bandwidth.

The faster your server machine and greater your bandwidth, the more browsers you can serve without degrading performance. Conversely, the server can easily get swamped with connections if you have a limited bandwidth. Requests can come in faster than the server can transmit data to the clients.

The *WebSTAR V* web server is fast enough so that a request can be served nearly instantaneously; thus, the default setting of 25 is more than adequate for most users.

The basic rule of thumb is to watch the *Web Activity* window and adjust the Max Connections up or down, based on your server's load and performance under that load. Don't be afraid to restrict the number of connections if your server cannot handle the load (because either the computer or you have limited bandwidth). Doing so may actually help speed transfer rates, as the first requestors will finish and allow others to come in.

The default for Max Connections is 25. The valid range for the Max Connections value is 1 to 9999. After you change the number of connections, you must quit the *WebSTAR* application and restart the web server. The change will not take effect until you do so, although the monitor windows will reflect the new settings.

Persistent Connections

The HTTP 1.1 standard allows a server to send several files to a browser during a single connection. This allows *Kerio WebSTAR* to serve an HTML file and several small images in one connection, instead of requiring the browser to set up a separate connection for each file being requested, and can be much faster. This is a feature of HTTP/1.1, and supports "HTTP 1.1 pipelining" which enables browser to send multiple GET requests in a single header.

The *Requests per Connection* setting within *Persistent Connections* allows you to set the maximum number of requests for a persistent connection. The default is 100 and the range is 1 to 999.

Newer browsers automatically send a header element if they want to request a persistent connection for a page (or frame). When *Kerio WebSTAR* gets one of these elements in an HTTP request, it holds a connection open for that browser, to avoid the overhead of creating new connections for every transaction.

Persistent connections remain open for the connection duration. Therefore, they take up a connection even when the server is not actively sending a file. The *Timeout* value allows you to specify a time (in seconds) for which no activity will timeout the persistent connection. The default value is 5 seconds and the range is 1 to 999 seconds.

When you enable *Persistent Connections*, you'll notice that the status windows will report that the number of active connections is often increased by 25% to 50%. To avoid rejecting other requests with a server busy message, you should increase the

number of connections allowed as well.

Timeout

This number determines the maximum amount of time (in seconds) that *Kerio WebSTAR* will wait before dropping an inactive connection. Timeouts can occur when a CGI application takes too long to execute or when a client doesn't respond to *Kerio WebSTAR*, implying that the connection has been dropped.

The default is 30 seconds, the minimum value is 1 second, and the maximum value is 240 seconds.

Buffer Size

This setting controls the size, in kilobytes (KB), of the data blocks that *Kerio WebSTAR* uses when reading binary files from disk (such as HTML, GIF, and JPEG images). *Kerio WebSTAR* reads from the disk and writes in chunks known as buffers, which are then passed to the client via the network connection. Adjusting this value will have little impact on server speed, although it may improve disk file access efficiency.

You can reset the buffer size based on the size of data you are serving. Browsers on slow connections may have trouble transferring large buffers, so you should experiment to find the best setting for your client mix.

On fast network connections (T1 or greater), set the buffer size in multiples of the disk block size (512). On slower connections, set it in multiples of 1500 (the TCP/IP MTU size).

The default buffer size is 20KB; the minimum size is 5KB and the maximum is 40KB.

Web Caching

Use to set some parameters regarding caching web pages to improve performance.

Caching is a method of storing data in memory (RAM) rather than reading it from the hard disk every time. *Kerio WebSTAR* includes caches for file information, and for the contents of the files that the server has sent (such as text, images, and sound).

The cache stores information about files recently served: the file name and suffix, type, creator, size, and last modified date. Because this information is accessible in RAM, *Kerio WebSTAR* can perform suffix mapping lookups and other processing very quickly. In most cases, enabling this option will improve performance considerably.

It also uses this cache to track folder requests, where the URL does not include a slash at the end, like this:

`http://www.domain.com/widgets`

The server must check to see that there isn't a file named `widgets` before converting the URL to the folder format (`widgets/`) and then getting the index file. Storing this

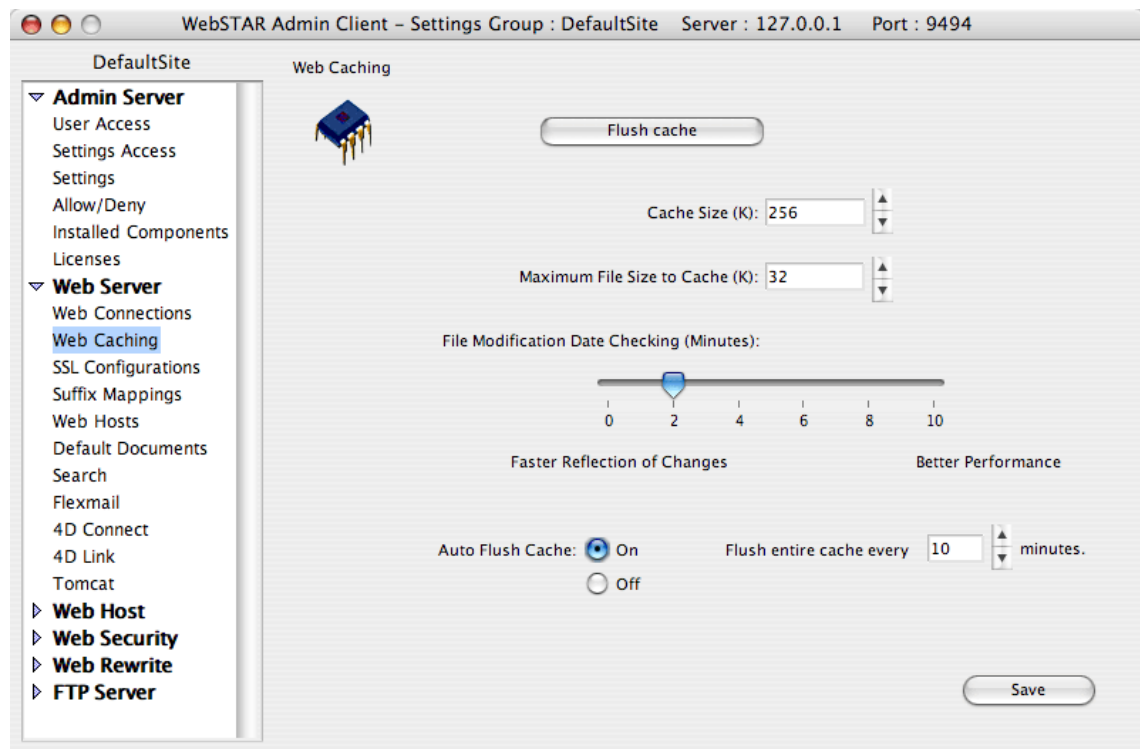


Figure 10.7 The Web Caching panel

information in the file cache lets the web server perform this step very quickly, which will improve performance on busy servers.

Note that this cache cannot tell when your files have been modified on the disk. You must flush it to make sure that the newest pages are served.

Data Cache

Web server data caching stores copies of the contents of the files on your web site in memory (RAM) so that the server doesn't have to open each file to serve the data inside it. This is particularly useful for sites with many small graphics files (such as logos and navigation buttons) used throughout the site. Opening and reading files requires calls to the Mac file system and accessing the hard disk, so serving the contents of these files directly from the *Data Cache* will speed up access to your site considerably. In addition, the *WebSTAR SSI Plug-In* automatically caches its data in memory using the *Data Cache*, saving time and processing.

All files not explicitly handled by plug-ins are cached. This includes those with suffixes `.html`, `.gif`, `.jpeg`, and so on. SSI pages are also cached automatically. Unlike other web server caches, you do not have to designate the best files to cache: the *Data Cache* will do it for you.

The *Data Cache* is an intelligent cache, using a frequency-based algorithm to automatically adapt to changing server loads. The Least Recently Used system tracks

when files are requested: when the *Data Cache* runs out of memory, the files requested recently are retained and the others removed.

As files change on the server disk, you want the *Data Cache* to serve the most recent version. However, checking the file before responding to every request takes significant time, especially on busy servers. You can schedule how often the *Data Cache* should check to see if the file has been changed. See the section File Modification Date Checking (Minutes).

Plug-Ins can use the *WebSTAR Data Cache*, and a number of thirdparty plug-Ins do so, or have their own caching.

Cache Size

The cache size is the amount of RAM allocated for this purpose, in kilobytes. The default setting for this field is 256KB. You can change the *Data Cache* size to any number between 100KB and 256,000KB (256MB).

Maximum File Size to Cache

You can control the *Data Cache* storage by defining how large a file can be cached. For example, if you have many 5KB files and a few 200KB files, it's much more efficient to serve the smaller files from the *Data Cache*. Large files take less advantage of the *Data Cache* — *WebSTAR* overruns most network connections long before the file transmission is completed, so reading the file from disk does not slow the transfer rate.

If you have many large files, and your server has a lot of RAM, you can allocate more memory to the server, increase the *Data Cache* size, and increase the maximum file size.

The default is 32KB, and the range is from 16KB to 25,600K (25.6MB).

Optimizing the Data Cache

Selecting the best cache size requires experimentation. A cache that is too small has no effect on performance, while a cache that is too large could waste memory. In addition, the *Data Cache* only avoids the overhead of disk access. You still have the network delays, even if you are on a T1 or a T3 internet connection. Data caching is most effective for small files used often in your web site.

For testing, you may want to use a utility such as Pounder or Netsplat to simulate a high load on your site, as well as checking your site from several different locations. For testing tools, see the Extending WebSTAR pages on the StarNine site:

<http://www.starnine.com/extendingwebstar.html>

The *Status* windows and *Browser Admin* monitor page will show you the memory usage while you're testing. Be sure to wait while the *Data Cache* fills up during your experiments.

After your server has been running for a while, refer to the *WebSTAR Data Cache Info* window. When the Efficiency and % of Cache Used numbers are high (over

75%) the Data Cache is working properly. These numbers are also displayed as the speedometer and fuel gauge.

If the cache usage is low, you may want to increase the *Data Cache* size. Make sure you also allocate more memory to the server. Keep testing and incrementing until you reach a balance.

Sometimes the *Data Cache* window will report a different amount of data sent than *WebSTAR* server status window, even though all suffixes are cached. Your *Data Cache* may be running out of RAM (it will say 100% efficiency). In addition, some items never appear in the *Data Cache* number, including CGI data, error files, no access files, and File-Not-Modified responses.

File Modification Date Checking (Minutes)

This slider specifies how often the *Data Cache* should check the server disk for the file modification date when it gets a client request for that file. It will only read the file from disk again if the date and time indicate that the file has changed.

The default is to check when a request comes at least 5 minutes after the last check. The allowed range is from *Always* (10-20 seconds) to *Never* (about 20 minutes).

Updating Caches

Because the *Data Cache* checks file modification dates, it will read your changes at most 20 minutes after you save the file to the server disk. However, if you are updating your site, you may want to see changes immediately. There are several ways to make sure you can see the changes.

Flushing the Cache

The Web server can flush the cache automatically. By default, this feature is enabled. It can be disabled by clicking the *Auto Flush Cache Off* radio button.

When you quit and restart *WebSTAR*, all caches are automatically flushed.

Manual Cache Flush

To flush the cache immediately, click the *Flush Cache* button.

Auto Flush Cache

The *WebSTAR* Admin Web Settings Cache panel and Browser Admin Controls page allow you to set an interval for the system to automatically flush both the *File Info Cache* and the *Data Cache*. This is rarely necessary.

The default setting is to have *Auto Flush Cache* set *On*. If you enable it, the default interval is 5 minutes. The range is 5 to 99,999 minutes.

SSL Configurations

If you are using the *Secure Sockets Layer* (SSL) protocol to provide authentication, encryption, and message integrity, you use the *SSL Configurations* panel to enter information about the certificate and encryption.

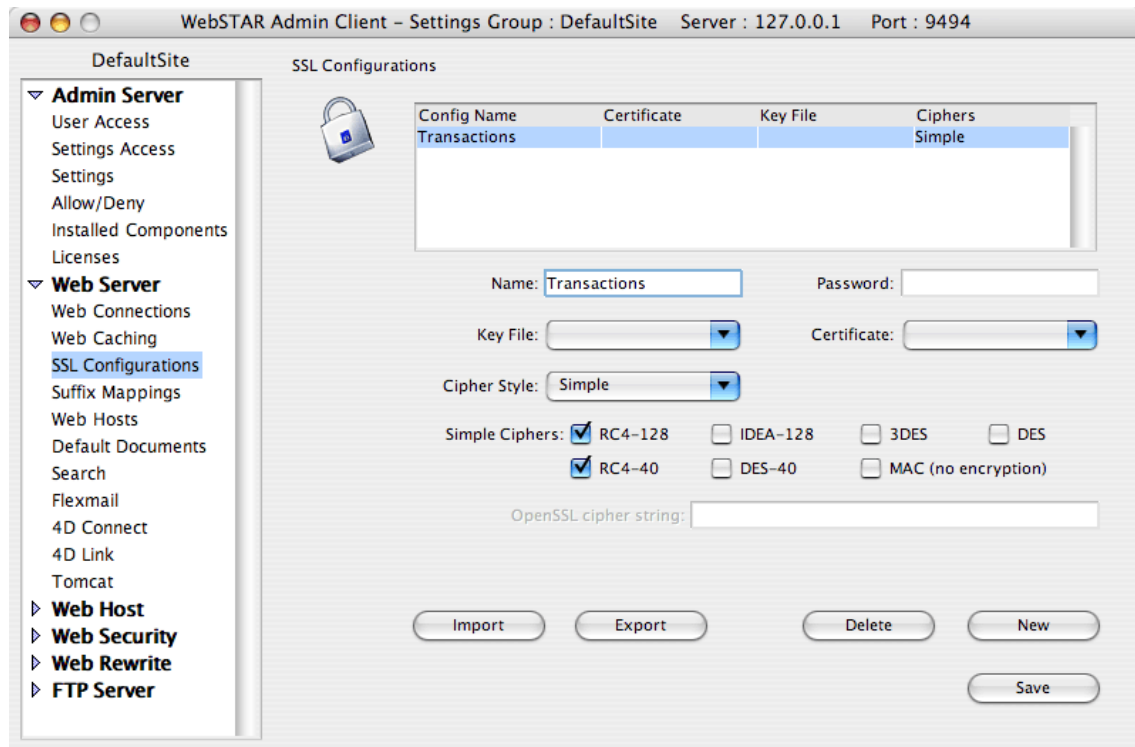


Figure 10.8 The SSL Configurations panel

Web Suffix Mappings

Kerio WebSTAR 5 uses CGIs and plug-ins for file listing, database access, multimedia, Server-Side-Includes, and other dynamic interactions. To launch most CGIs and Plug-Ins, the *WebSTAR* web server uses special URLs (as specified by those add-on programs). When one of these the URLs is received, the server will examine the file name element and send the request to the appropriate CGI or plug-in.

Entries in the *Suffix Mapping* table define what should happen, based on the file name suffix, file creator or file type code. This could be processing by a CGI or plug-in, or serving the file without any modification on the server side.

Chapter 10 Server-Wide and Virtual Host-Specific Settings

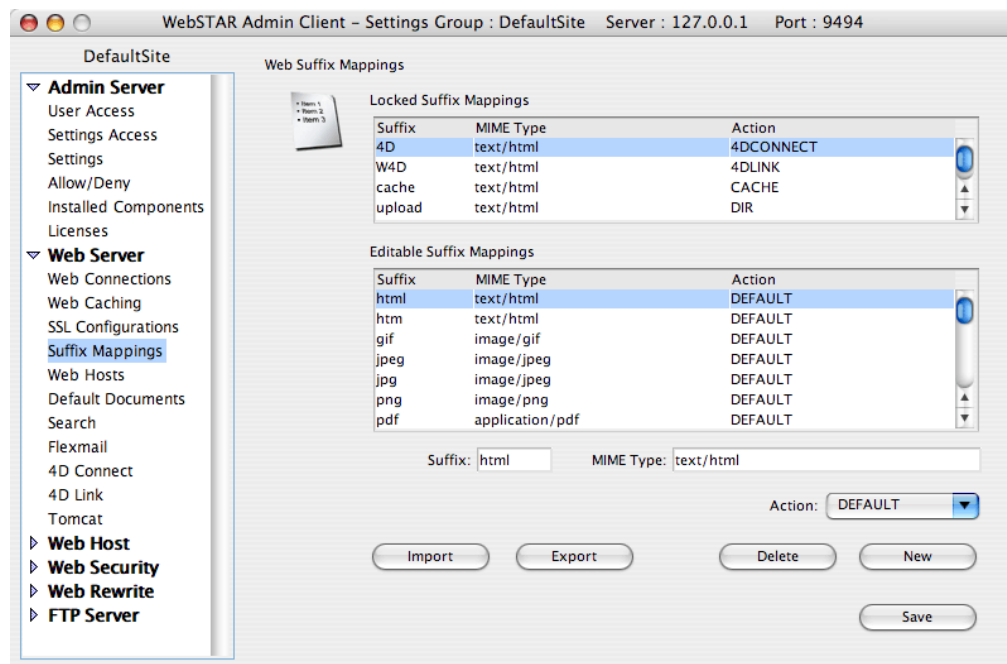


Figure 10.9 The Web Suffix Mappings panel

Web Virtual Hosts

Use to enter your virtual hosts and the table that maps incoming requests to particular virtual hosts.

Creating your virtual host is one of the first tasks you will do using *Admin Client*. For detailed information, see chapter 5.

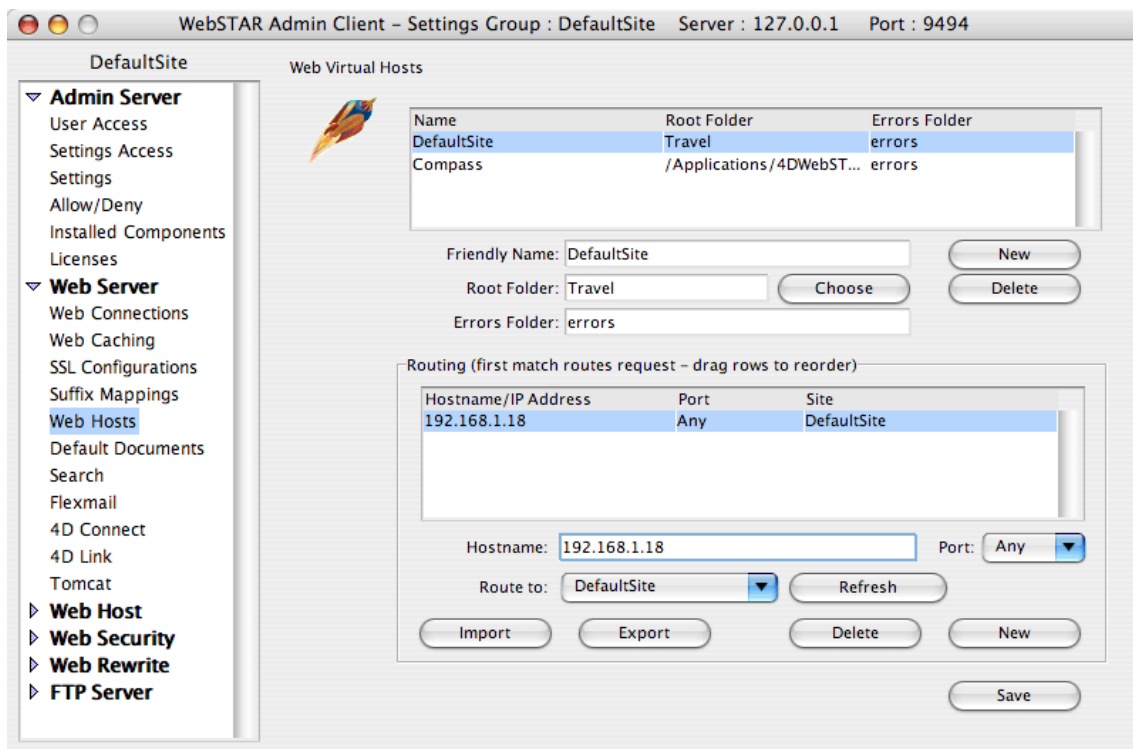


Figure 10.10 The Web Virtual Hosts panel

Default Document

Use to enter the names of the documents you wish to use as the home page for your web sites.

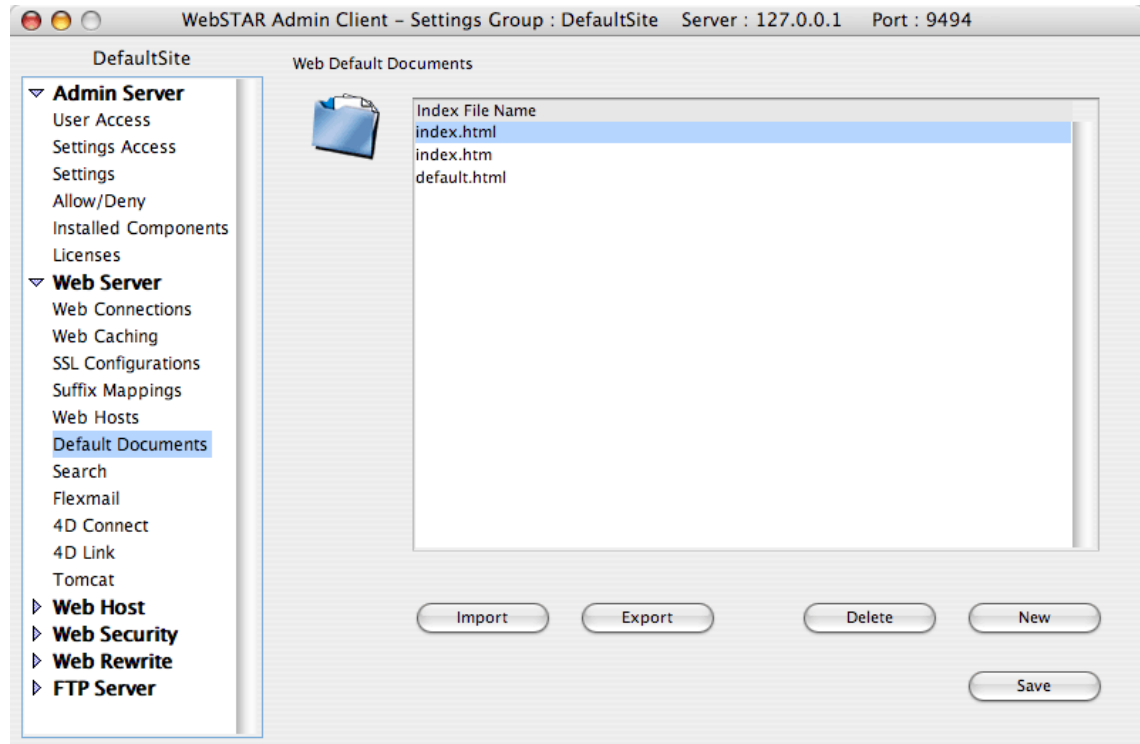


Figure 10.11 The Web Default Documents panel

When a browser logs onto any virtual host served by your web server, the first document whose name appears in this list will be served.

Search

Use to create an index of the text used in your web sites to enable fast web searching.

With a search index, visitors can search the contents of the files on your site very quickly. Search indexes text files (including HTML) and PDF files, and provides an interface for very fast and powerful searching, including Boolean operators, relevance ranking, results formatting, and more.

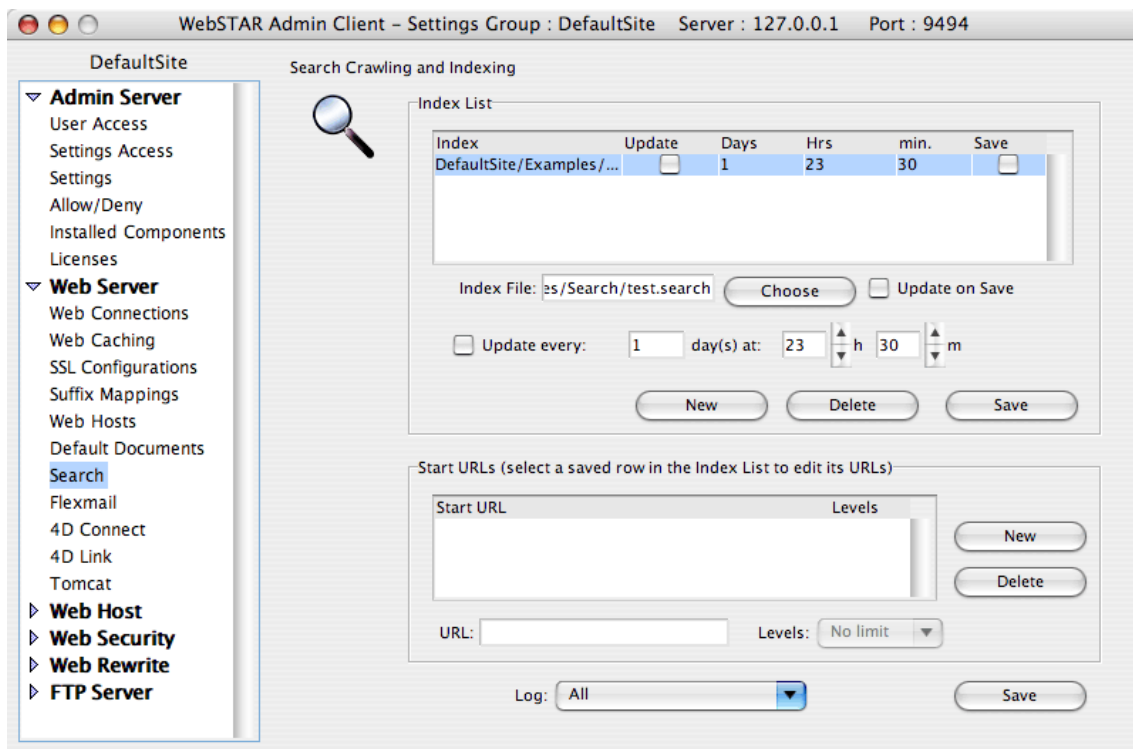


Figure 10.12 The Search panel

4D Link

The *Kerio WebSTAR 4D Link* plug-in allows you to build a web site using a *4th Dimension* database to serve dynamic pages. For example, if you are running a real estate database, you can keep information about each home in the *4th Dimension* database and create a web site that allows browsers to search the database by price, location, size, etc. With *4D Link*, you can send a browser's query to the database. 4D can do the search, create a real estate listing on-the-fly, and return the html for the listing to *Kerio WebSTAR 5*.

The *4D Link* page lets you identify the database (or databases) that serve pages. You enter the IP or domain name of the machine that is running the database, the port it is using, and the timeout. If you are serving from more than one 4D database, you also need to enter an ID for each database.

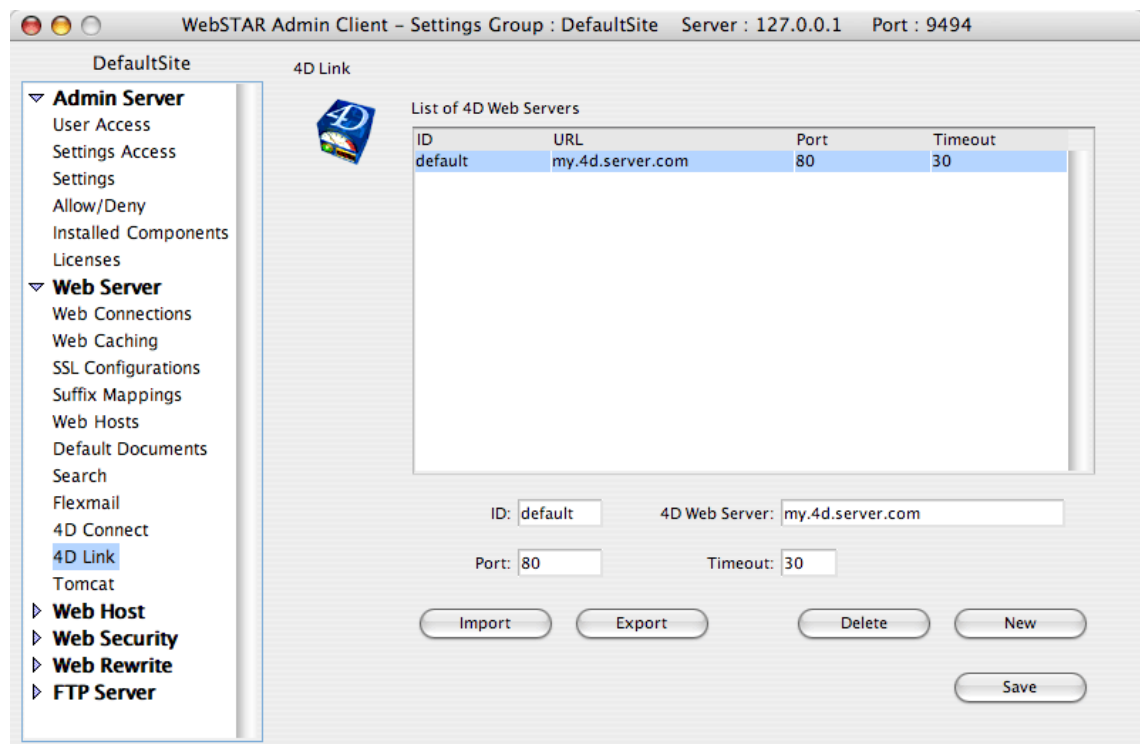


Figure 10.13 The 4D Link panel

4D Connect

The *4D Connect* is also a *WebSTAR* plug-in that allows you to use *WebSTAR* as the web server for a 4D database. When requests that contain a *4D Connect* directive are received from a browser, *4D Connect* passes the request on to the *4th Dimension* database along with an HTML template file. Kerio then returns the requested information to *WebSTAR* by replacing the placeholders in the template with values from the database.

4D Connect provides essentially equivalent functionality as *4D Link* (chapter 20), but the latter does not use the template approach. Both *4D Link* and *4D Connect* can be used simultaneously.

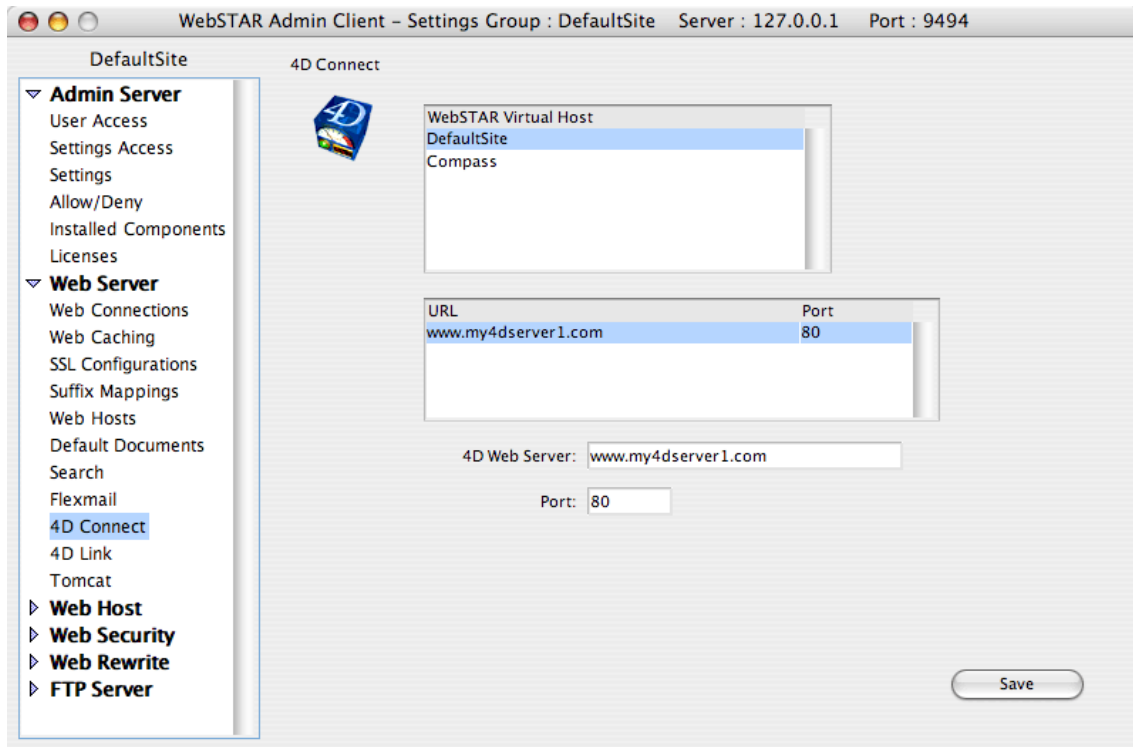


Figure 10.14 The 4D Connect panel

10.2 Virtual Host-Specific Settings

The following aspects of *Kerio WebSTAR 5* are configured separately for each virtual host:

- Configure CGI actions
- Monitor and log web usage
- Configure web security. You can create secure realms within your web site and control access via a password system and or prohibit or allow specific hosts/IP addresses to connect.
- Set up shared folders using WebDAV and set permissions. You establish WebDAV service using realms. See chapter 7 for more information.

CGI Settings

Use to link CGIs to actions and to limit execution of CGIs to certain cases (Permissions). By default, CGIs are placed in the global cgi-bin folder, which is in the *WebServer* folder rather than any virtual host. The unchecked options, which allow CGIs to be executed in each virtual host's own cgi-bin folder or even any folder with a virtual host folder can create greater security risks.

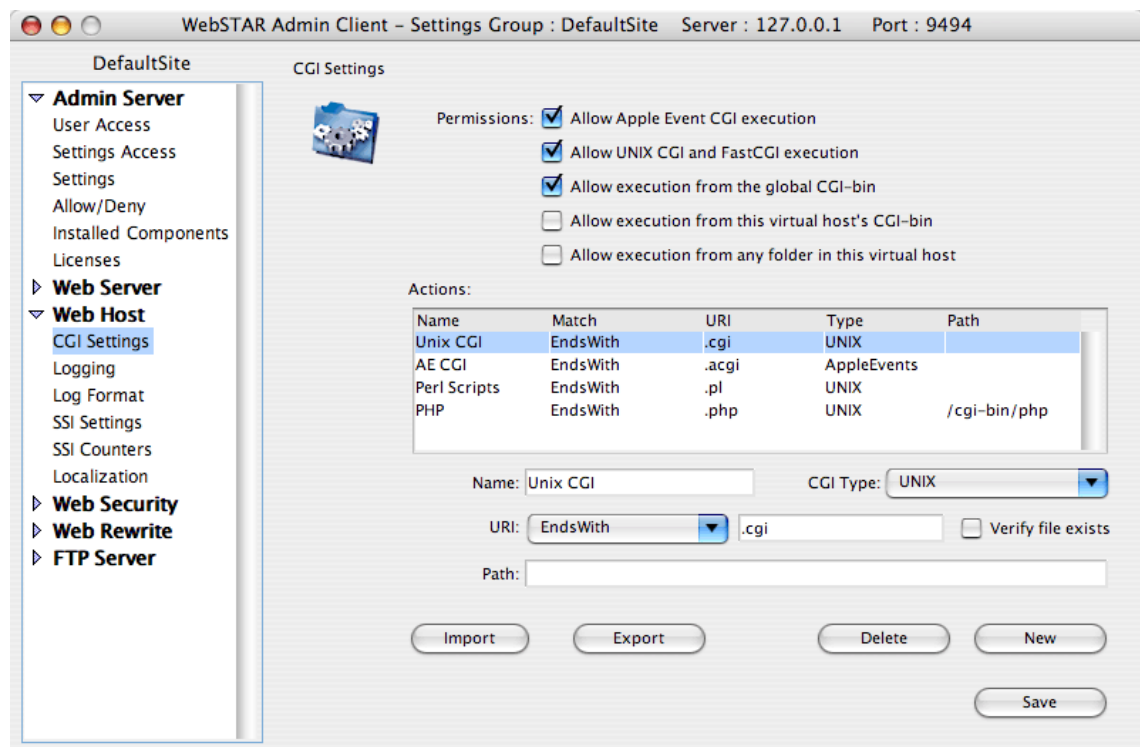


Figure 10.15 The CGI Settings panel

Web Logging

Use to control the transaction information that you see in the monitor windows and save in the *Log* file.

Log Archiving allows you to save separate log files covering specified periods of time, such as every day, every week on Thursday, or monthly. The log files are automatically created at the time you set, filled, and then closed when the next one is created.

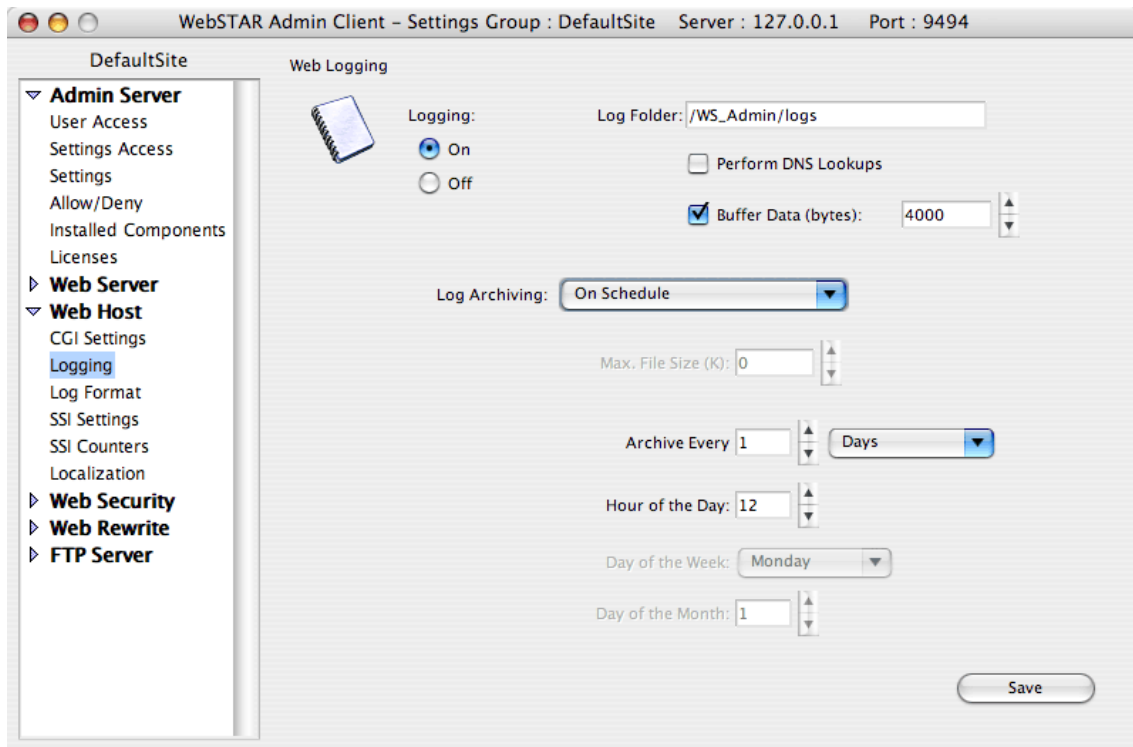


Figure 10.16 The Web Logging panel

Log Formats

For each WebSTAR transaction, the server displays a line in the monitor window (not available in this release) and saves an entry in the log file.

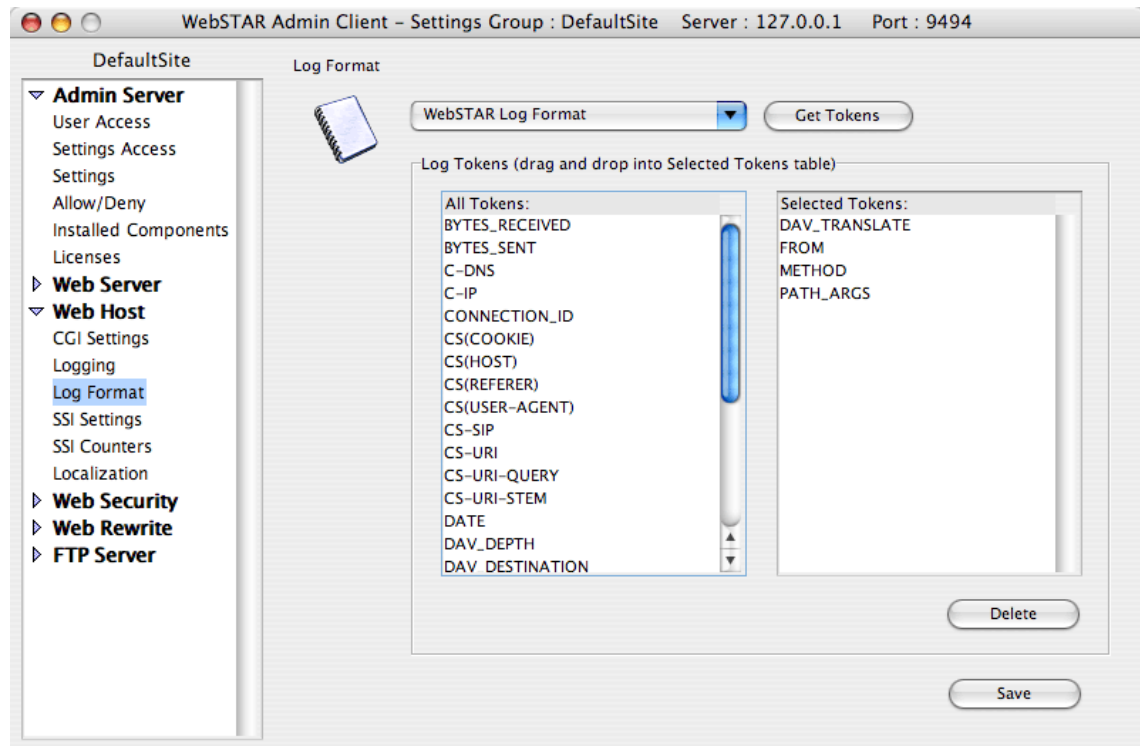


Figure 10.17 The Log Formats panel

The *Log Format* screen lets you specify what kind of information to include in these entries. You can choose between the *WebSTAR Log Format*, the *Common Log Format* (CLF) and *Extended Log Format* (ExLF). They allow you to save data in formats compatible with many third-party log analysis programs. These applications will read through your log file or multiple files, and create statistical reports on web site usage.

<http://www.webstar.com/exws.html>

Note: The following panels are closely interrelated and implement web security in *Kerio WebSTAR 5*. The topic is discussed in more detail in chapter 6.

Web Realms

Use to restrict access to all or parts of a virtual domain. A realm is defined by a string that is contained within a request to the server. Often, a realm corresponds to a particular folder or file. When the string is found in the request, the security system intercepts the request. Depending on how you configure the realm, the security system may check the IP address/domain name of the requesting computer or display a dialog box in which the user must enter a valid username and password. The *Web Realms* panel allows you to enter the string and specify how the security system will respond when *Kerio WebSTAR 5* receives a request to serve the realm.

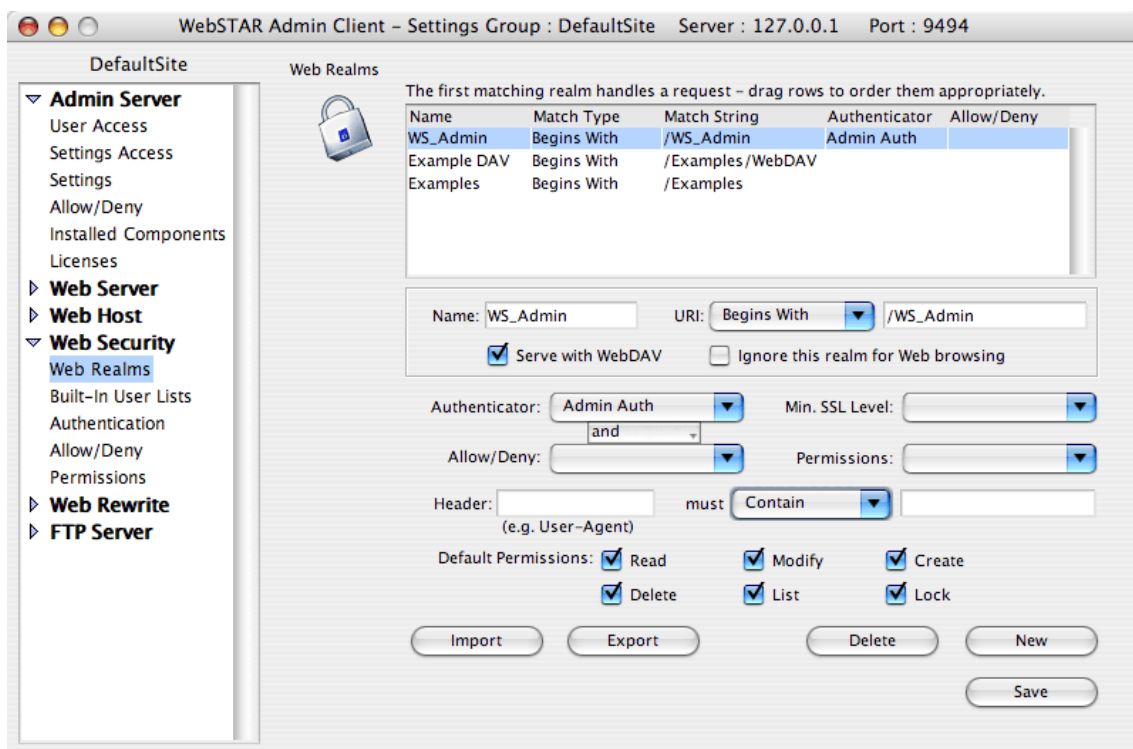


Figure 10.18 The Web Realms panel

Access to this realm is controlled by the authenticator “Customers”, which references a list of authorized users. The user list is defined elsewhere and may be linked to other realms. Permissions are controlled by the *Default Permissions* settings at the bottom of this screen.

Built-in User Lists

Use the *Built-in User Lists* panel to enter usernames and passwords for authorized users. Since user lists are stored separately from realms, they can be reused.

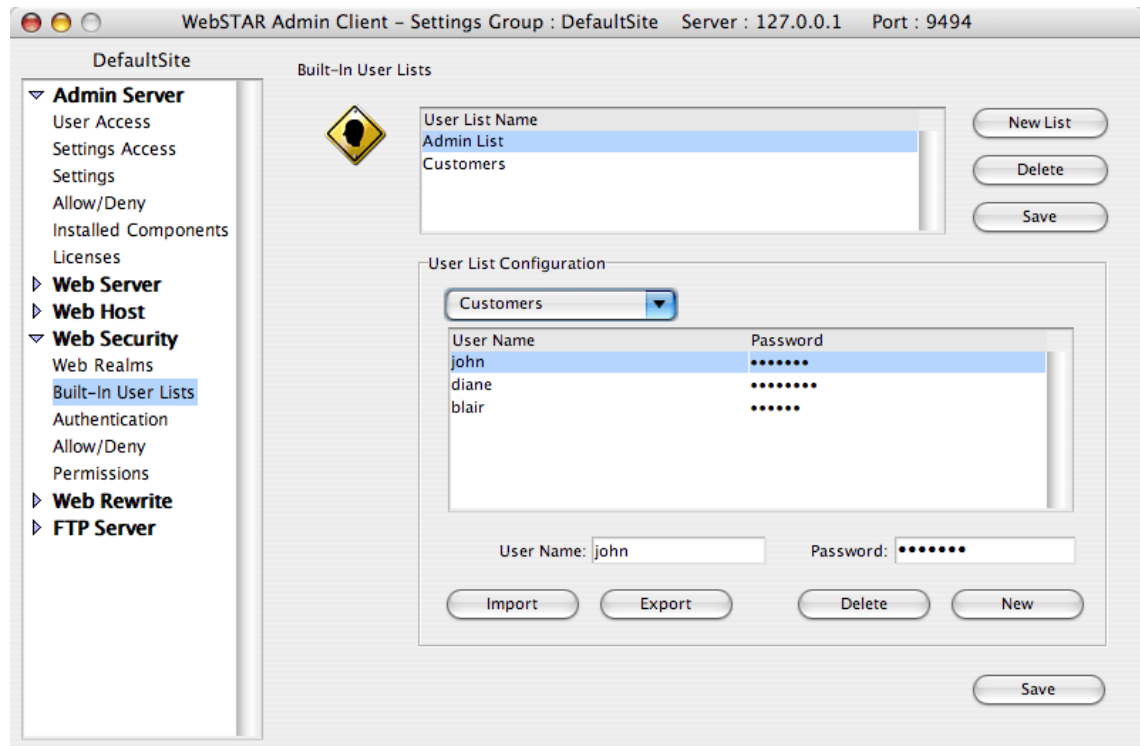


Figure 10.19 The Customers user list referenced by the realm in figure

Authenticators

Use the *Authenticator* panel to define one or two user lists and set the type of authentication.

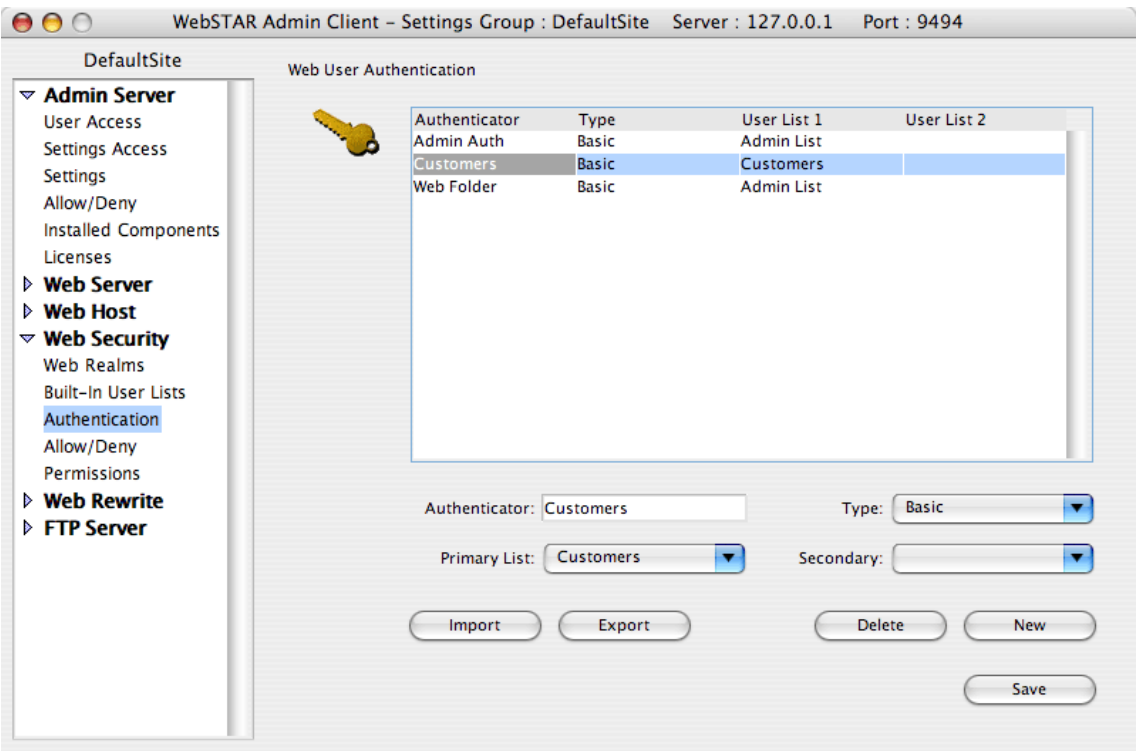


Figure 10.20 The Customers authenticator consists only of the Customers user list defined in figure

Permissions Lists

Use the *Permissions* list panel to set access rights to realms on a person-by-person basis. You can instead use the *Web Realms* panel to grant the same level of permissions to all users on the user lists associated with the realm.

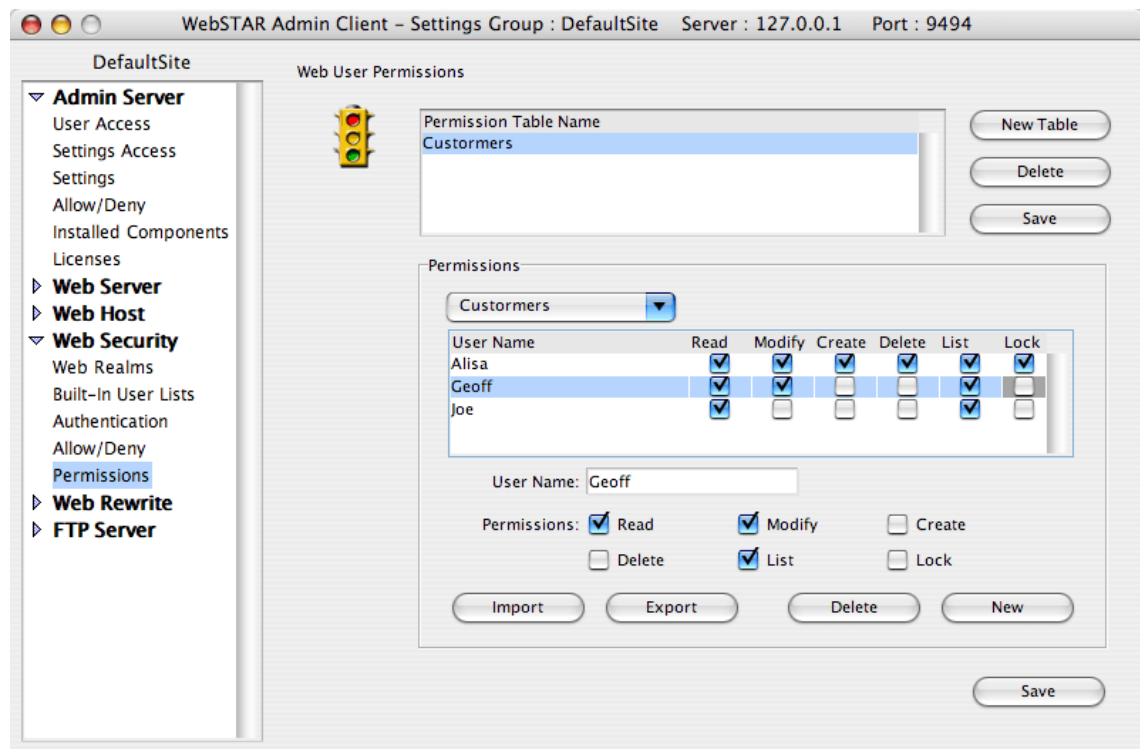


Figure 10.21 The Permissions list, Customers grants full access to one user and limited access rights to the other two users

In figure 10.21, there is a one-to-one relationship between the users on the user list (figure 10.19) and the *Permissions* list, but there is no such requirement. Users on a *Permissions* list may be drawn from several user lists and a person on a user list may appear on several permissions lists (i.e., if a user has different levels of access to different realms). The only requirement is a logical one: a user on a permissions list must appear on a user list — otherwise, he has no username and password.

Allow/Deny Table

Use the *Allow/Deny* panel to list domains and/or IP addresses that are either permitted or prohibited from accessing a realm. You can use this method of controlling access to a realm instead of or in addition to user lists and authenticators.

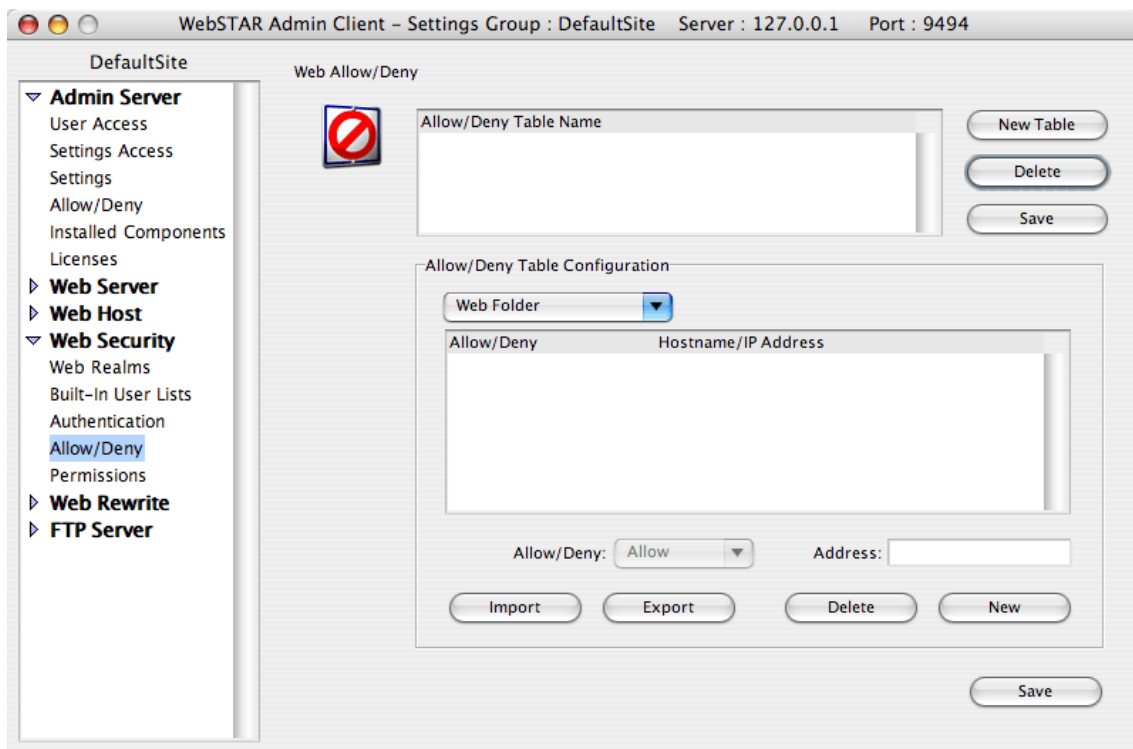


Figure 10.22 An Allow/Deny table is configured to allow access from only three computers

See chapter 6, for more information this group of panels.

Chapter 11

WebSTAR FTP Server

FTP (File Transfer Protocol) is a protocol for moving files from one computer to another. It allows clients to both upload and download files. Many browsers include FTP support, and stand-alone FTP client applications are also available.

Kerio WebSTAR FTP provides controlled access to your server folders using this protocol. You can create users who have to log in with a name and password, and specify which folder (and subfolders) they can view, and so on.

In addition, you can create a public FTP area. According to FTP conventions, anyone can download data from this area by logging in with the name anonymous. For information about creating access privileges for anonymous users, see the section [11.4](#).

Kerio WebSTAR 5's support for FTP is an alternative to WebDAV. WebDAV uses the http protocol rather than the FTP protocol. It offers increased security since it is well integrated into *Kerio WebSTAR 5's* web security features.

Kerio WebSTAR 5 administers FTP at the server-wide level, while WebDAV is administered at the level of the web virtual host. Among other things, this means that you can set up separate WebDAV folders per virtual host that are completely invisible to other virtual host users.

11.1 Setting Up an FTP Server

Before you use *Admin Client* to set up your FTP server, you should decide which folder will be the root folder for each user. The root level folder is the one the user sees when he/she first connects to the FTP site. If the required folders don't exist, you'll have to create them.

WebSTAR V FTP allows you to use any folder on the server machine as an FTP root folder. This is a powerful feature, but should be used with caution. FTP users can access all the subfolders within their root folder, but they cannot see any other folders on the disk. This allows you to take best advantage of your available disk space, but can cause major security risks, so think carefully before you designate the root folder. It is best to stay within the *WebServer* folder, as you do when configuring your virtual hosts.

In addition to local files and folders, *WebSTAR FTP* will attempt to follow aliases to other files and folders. Be sure that you know what aliases you include in FTP folders.

Using Admin Client to Configure the FTP Server

Since FTP is administered server-wide, you must have server admin privileges to access the FTP Server panels in the *Admin Client* application.

While logged in with server admin privileges, the *FTP Server* group appears in the browser. It contains two items: *Users* and *Connections*.

The Users Panel

Use the *Users* panel to enter the following settings for each FTP user:

- *User Name and password* — The user must enter his/her username and password via the FTP client application to access the root folder. The exception is the anonymous user who does not need to enter anything in the username field and does not require a password. For more information, see section 11.4. The username and password can each be up to 31 characters long, and can include spaces.
- *Maximum Directory Size* — This parameter refers to the total size of the root folder. Its main purpose is to prevent users from uploading large files that take up all your disk space. If you want to allow a user to upload files, be sure that this parameter is set to a value larger than the current size of the root folder; otherwise the user will be unable to upload anything.
- *Permissions* — For each user, you can grant the following permissions:
 - *Allow Login* — Uncheck this item to temporarily disable this user's access.
 - *Download Files* — Allow this user to copy files from the server to their local disk.
 - *Upload New Files* — Allow this user to copy files from their local disk to the server, in their root folder.
 - *Delete and Change Files* — Allow this user to delete files in their root folder, or replace a file with another of the same name.
 - *Create Folders* — Allow this user to make new subfolders in their root folder on the server.
 - *Delete Folders* — Allow this user to delete subfolders from the server. They cannot delete their root folder. Clients cannot delete folders if there are any files in them. Some FTP clients do a recursive delete, removing the contents of the child folders and then the parent folders.

By default, only *Allow Login* and *Download Files* are selected.

- *Full path to FTP root folder* — The root folder is the top-level folder that the user sees when he/she connects to the FTP server. All subfolders within the root folder are visible. This pathname must be entered in a different format than in other *Admin Client* panels.

The path to the FTP folder must be the full path from the parent volume. It begins with a slash, followed by the volume name, and ends with the name of

the FTP folder. The intermediate subdirectories are separated by slashes. For example, if you use a folder within the *WebServer* folder named *DropBox*, the path would be:

```
/Macintosh HD/Applications/KerioWebSTAR/WebServer/DropBox
```

(assuming the name of the hard disk is *Macintosh HD*).

You must enter the FTP folder for each user even if they all access the same folder.

The Connections Panel

Use the *Connections* panel to set the global properties for FTP service. These properties are:

- *Max Connections* — This is the number of users that can be using FTP service simultaneously. The default number of connections is 12, and the maximum is 9999.
- *Timeout* — *WebSTAR FTP* must automatically close connections to FTP clients. Unlike HTTP clients (web browsers and WebDAV clients), FTP clients log in and retain a connection. However, many users do not bother to log out or close their FTP session, so the server must close it for them, after they stop communicating (no client will be closed while it is active). The timeout value defines the inactivity interval.

The default timeout interval is 300 seconds, and the range is from 30 to 999 seconds.

- *Reported Server Name* — When an FTP client connects to a server, it asks what kind of server it is. Unfortunately some FTP clients display, format, or return results differently depending on what type of server they are talking to. Setting the server to *UNIX WebSTAR FTP* with the Append *.bin* suffix to file-names option turned on should be the most compatible as it will cause both Macintosh and Windows browsers to display information correctly and allow for downloading of files in *MacBinary* format.

Other options, included for Mac-only intranets, are *Mac OS WebSTAR FTP* and *Mac OS Peter's Server*.

- Use *MacBinary* for *.bin* files: *MacBinary* is a Mac-specific extension to the FTP standard to transfer both the data and the resource forks of Macintosh files. Many files, especially applications, contain important information in this resource fork, so Mac FTP clients support *MacBinary*, as do most Mac browsers, since *Netscape 4* and *Internet Explorer 3*.

If this option is checked, the *WebSTAR FTP* server will assume that all files with the suffix *.bin* are *MacBinary*, and will use this data type to transfer the files.

Note: If an FTP client attempts to get a resource-only file without using *MacBinary*, *WebSTAR FTP* will return an error. The solution is to put the FTP client

into *MacBinary* mode.

Unfortunately, the **Restart** command, used by FTP clients to resume downloads if they are interrupted, is not compatible with *MacBinary* files, because all the file information is at the beginning of the file.

Displaying Files with Resource Forks

The *WebSTAR FTP* server will not display the names of files with resource forks unless you have one of the following checkboxes selected.

- *Append .bin suffix to filenames* — Checking this box will automatically put the suffix `.bin` after file names of files with resource forks, when the FTP server sends the listing to the FTP client. You can see the filenames with the suffix, and the client will then transfer and decode them properly.
- *Display actual file names* — Checking this box will display files with resource forks without adding the `.bin` suffix. However, some FTP clients may not transfer the resource forks correctly because they don't recognize the format.

Both Display Options Selected

If you check both boxes, you'll show two copies of each file.

Connection Messages

Some FTP clients, especially text-based clients on Unix machines, display a message when you login and logout, and when you open each folder. *WebSTAR FTP* allows you to designate text files that store these messages and will display them at the appropriate times.

- *Log-On Message* — The path to the text file containing the welcome message. Enter the full path from the parent volume in the same format as the path to the root directory for each user.
- *Log-Off Message* — The path to the text file containing the goodbye message. Enter the full path from the parent volume in the same format as the path to the root directory for each user.
- *Folder Message* — This is a file name, like the web server index file name field. Enter the name here (such as `.readme`). If you put a text file of that name in a folder, *WebSTAR FTP* will send that file to the client whenever they open that folder. Some FTP clients display it to the user; others ignore it.

11.2 Creating FTP Users

You enter specifications for a user and for an FTP root folder in one operation.

To enter a user:

1. Click the *Users* item in the *FTP Server* group.

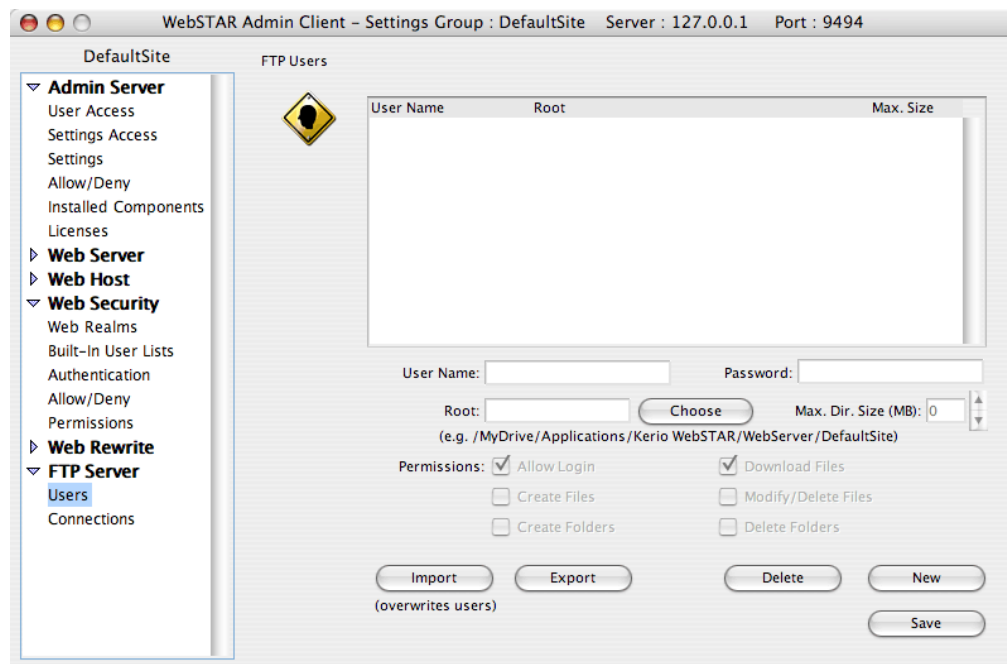


Figure 11.1 The FTP Users panel

Each row in the table stores an entry for a username and password, the FTP root folder he/she can access, permissions, and maximum directory size.

2. Click *New* to create a new row in the table.

When *Admin Client* creates a new row in the table, use the entry areas below the table to enter:

- The person's user name and password,
- The full path to the FTP folder in the format noted above,
- The maximum directory size (MB),
- Permissions for the user.

If you are creating an entry for anonymous FTP, enter *anonymous* in the *Username* area and leave the *Password* area blank.

3. Enter the full path to the root folder in the format shown below the entry area.

The format for the full path begins with a slash, followed by the volume name and ending with the name of the root folder.

For example:

`"/Macintosh HD/Applications/KerioWebSTAR/WebServer/pub"`

designates the folder “pub” within the *WebServer* folder as the root folder.

The easiest way to do this is to click the *Choose* button to display a *Choose Folder* dialog box. If you do it this way, you don’t need to worry about the syntax of the pathname.

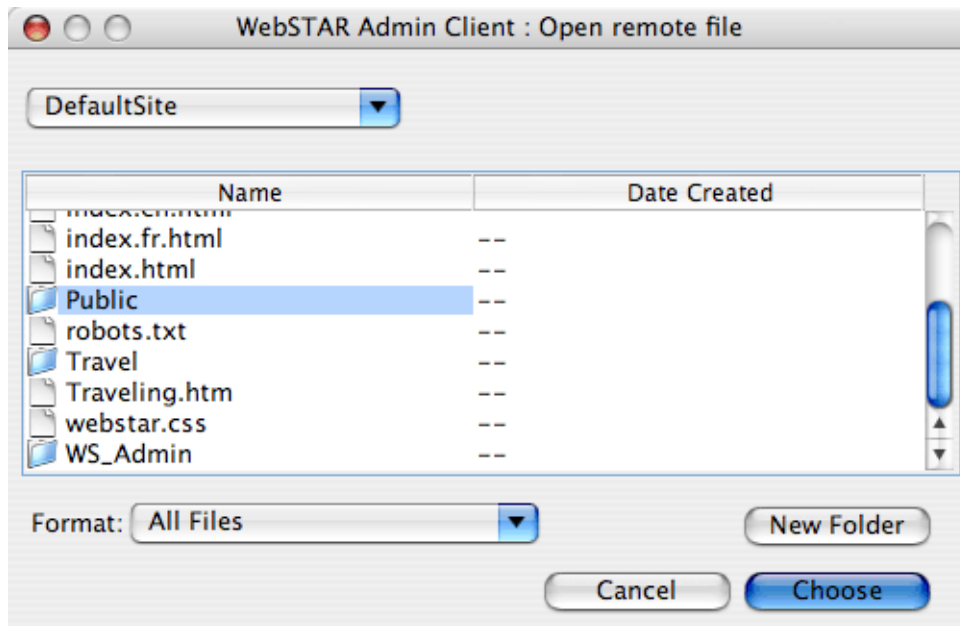


Figure 11.2 The FTP root folder Choose Folder dialog box

If a suitable folder does not already exist, click *New Folder* to add one within the current directory. Highlight the desired folder and click *Choose*. When you do so, the (correct) full path is written into the *Root* field.

4. Enter the maximum directory size for this user.

Keep in mind that the space available for uploads is this value minus the current size of the root directory.

5. Use the *Permissions* checkboxes to grant appropriate permissions to this user.
6. (Optional) Repeat this process for each user.
7. Click *Save* to save these users to the server.

Here is an example FTP user.

11.3 Setting FTP Connections Options

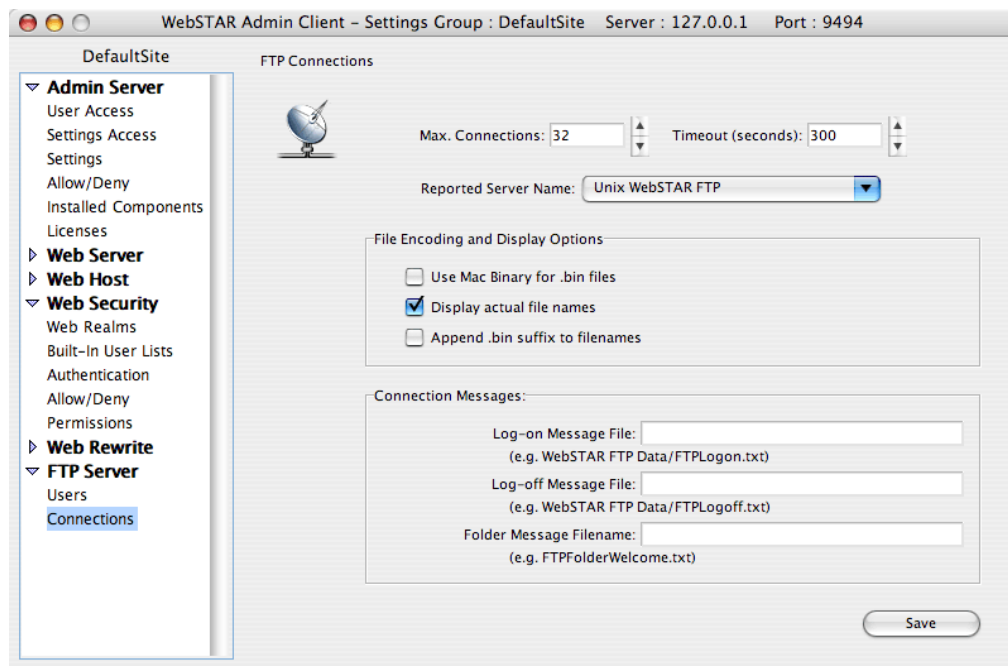


Figure 11.4 The FTP Connections panel

11.4 Anonymous FTP

The Internet custom is to allow public FTP access to a server via a special anonymous user: the FTP client enters anonymous as their user name. In many cases, the client automatically sends the user's email address as the password. The *WebSTAR FTP* server supports an anonymous user account.

To make a public-access folder for your FTP server:

1. Select the location of the FTP root folder carefully.
2. Name your FTP root folder. A common name for these folders is “pub”, short for “publications”.
3. Put an example file in the anonymous root folder, for testing.
4. Create an FTP user entry with the name anonymous, and the publicaccess folder as the root folder.
 - If you leave the password field empty, anonymous users are not required to send a password to the server.
 - If you put anything in the password field, anonymous users must send something, such as their email address, in the password field. You can use an asterisk (*) in this field to remind yourself that any word will match. Even if you require some text, it's not a real password: you cannot require a specific password on an anonymous account.
5. Set the FTP privileges. You may not want anonymous users creating folders or even uploading files.
6. Click *Save* to the entry to the server.

Here is an example anonymous entry. Note that only the *Allow Login* and *Download Files* permissions are set.

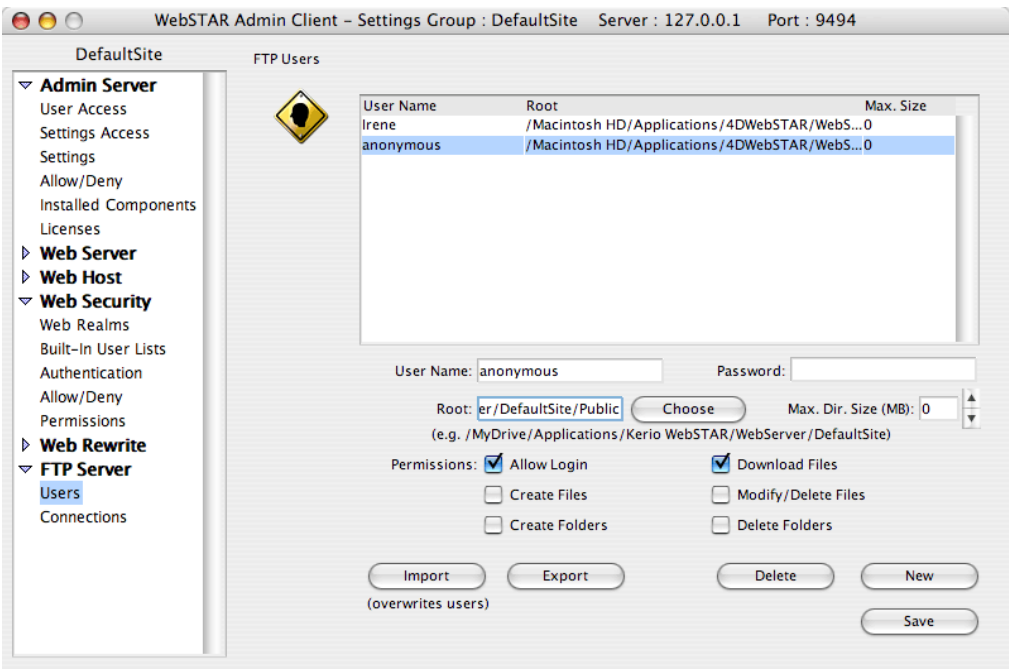


Figure 11.5 An anonymous FTP entry

7. Log into your FTP server's "pub" folder from any FTP client using the anonymous setting, to make sure that you can access the correct folder.

11.5 FTP Security

WebSTAR FTP provides a great deal of flexibility in choosing what folders are visible to FTP users, but this can compromise your server security if you are not careful. In addition, FTP passwords are not protected so anyone monitoring your data stream might be able to extract them and have access to your server folders.

Folder Security

If an FTP client attempts to open a folder with an alias to an unmounted network volume, the *WebSTAR* server machine will attempt to mount it. If a volume password is required, or the volume is unavailable, this may stop or slow your server until someone dismisses the alert on the *WebSTAR* server machine.

In general, you should allow FTP access to carefully specified folders, which contain limited amounts of material, and which do not include private or confidential data.

Note: Some older versions of Navigator don't handle FTP properly for Macintosh unless the server identifies itself as "Peter's Server".

File Security

WebSTAR FTP will not display or allow the transfer of *WebSTAR* control files (files with a *Creator Code* starting with WWW). These include *WebSTAR* application and plug-in files.

FTP users can also download applications, so you can distribute programs you have written or otherwise have the rights to. Make sure that you and any users with FTP upload privileges do not distribute commercial software.

Password Security

FTP passwords are always sent as plain text. This is a security problem: if someone is monitoring your data stream or that of one of your users, they can see the password clearly. They could then use the password to use your server for their own purposes, such as, passing pirated software or hacking your web site. The limits on FTP folder size can only help to a certain extent. To avoid this, limit user access to certain very specific folders.

If you use FTP to update your web site, try to use a staging folder for storing changes, and then transfer the changes to the main folder using another protocol. If you must use FTP to access your main server folder, change your password often.

Monitoring FTP Use

All FTP requests and responses are logged. You can watch your FTP server usage in the *WebSTAR Admin Monitor* window, and check your server logs for any unauthorized transfers.

11.6 FTP Data Types Supported

The *WebSTAR FTP Server* supports the following FTP data types:

- *ASCII* — For text files (default type)
- *Binary* — For graphics and other binary file types (this will transfer the data fork of Mac files but not the resource fork)
- *MacBinary* — For applications and other files that require the resource fork. Mac FTP clients can be configured to request the this data type. *WebSTAR FTP* will automatically use *MacBinary* if the file name ends in the suffix `.bin`.

11.7 FTP and Multiple Hosts

You can have the FTP server respond to requests for multiple hosts on the one server. FTP servers don't have to match the domain with any particular file (unlike web servers), and *WebSTAR FTP* doesn't use the web server's virtual hosts settings.

To respond to FTP requests for your other host and domain names, work with your ISP or network administrator to add an alias record (CNAME) in the host's DNS record.

For example, you can add a CNAME alias from `ftp.domain.com` to `www.domain.com`. FTP requests for the second domain name will be automatically routed to your *WebSTAR FTP* server.

FTP URLs in HTML

With most web browsers, it's very easy to link to FTP, using the URL `ftp://....`. For example, to see the *WebSTAR* directory, use this URL:

```
ftp://ftp.webstar.com/
```

In this case, the folder has anonymous access, so it will display immediately.

For private folders, the browser will display a login dialog, and you'll have to enter a user name and password. Then the browser will display the folder listing or start downloading a file.

You can also use FTP URLs with a user name and password included. To do this, include the user name, a colon, the password, and an at sign (@) before the host name:

```
ftp://HumanResources:hM3z9s@www.domain.com/humres/news.html
```

Most FTP clients can take this apart and send the correct URL and login information to the server. Be sure to test this before publishing such URLs, and limit the data in these folders carefully.

You should use only letters, numbers and standard punctuation in the folder and file names. It's best to avoid spaces or Mac extended characters in the folder and file names. Do not use the / (slash) or : (colon) characters in filenames. If you must use other characters, such as spaces, in file names, you'll have to encode the URLs.

11.8 FTP Standards and Conventions

WebSTAR FTP is written according to the FTP standards. It supports all common ports, all standard commands, and some optional commands.

FTP Specification

The official FTP specification is at:

```
ftp://nic.merit.edu/documents/rfc/rfc0959.txt
```

FTP Ports

FTP clients usually use two ports: port 20 for the data connection, and port 21 for the control connection. Browsers that act as FTP clients using the PASV command may use any port in the 9000 to 12000 range, and *WebSTAR FTP* will handle the requests properly

11.9 WebSTAR FTP Troubleshooting

This chapter helps you locate and fix problems with the *WebSTAR FTP* server.

FTP Server Troubleshooting

If you see errors on startup, in monitors, logs or users report problems, use this list of common problems to locate and fix your FTP server.

The WebSTAR FTP Server Startup Errors

Error FTP port already in use, server not running.

If you have another FTP server running on the same machine, the first one launched will be listening on port 21 for control transmission, and if *WebSTAR* is launched

second, the *FTP Plug-In* won't be able to listen on that port. You can't change this port number.

Check your *Sharing* preference panel in Mac OS X *System Preferences* and make sure that *Allow FTP Access* is Off.

Error startlistener failed (-3247).

You may have a second copy of *WebSTAR* running on the same machine. While the web server can listen on multiple ports, the FTP server cannot. You should remove the *FTP Plug-In* from the *Plug-Ins* folder of the alternate server.

Error in initialization: *The FTP Plug-in could not open the "WebSTAR FTP Users" file.* The *WebSTAR FTP Users* file was not found in the *WebSTAR FTP Data* folder in the *WebServer* folder.

If you have not yet created any FTP users entries, don't worry, *WebSTAR FTP* will create the file for you.

If you do have FTP user entries, quit the server, locate the file or a backup, make sure it's in the right folder and restart *WebSTAR*. Otherwise, you'll have to re-enter all the FTP user entries.

Error '*could not get ftp* configuration resource*'

The folder *WebSTAR FTP Data* is probably missing from the *WebServer* folder. Make a new folder of that name or re-install the *WebSTAR FTP* server, and the server will work properly.

Error *WebSTAR FTP: Could not determine IP address of server*

The DNS server is not responding properly: it may be down or otherwise inaccessible. *WebSTAR FTP* is trying to do a DNS lookup of it's own IP address, and the lookup is failing for some reason.

First, try using a web browser or FTP client to check another server. If your DNS server is down, you will get an error.

If that works, make sure you have entered a server name in the *Admin Web Settings* panel.

If the FTP server still doesn't work, check your TCP/IP control panel and work with your DNS administrator to check your DNS records.

Errors and Error Codes

Bad Folder entry in the Log means that the folder has a slash (/) somewhere in the path. Check your hard drive and enclosing folders for this character, and remove it.

Write Stream Error -5 means that the client has closed the connection. This is informational only, does not indicate a problem with the server or the client, and does not require any action by an administrator.

Error -17 means that the request failed. This usually occurs when the server requests that the client close the connection and the client has already closed the connection. In

most cases, it does not indicate any problem or require any action.

Error -550 (Write Permission Denied): this user account does not have the *Upload New Files* option checked, and the user tried to upload a file.

Error -552 (Exceeded storage limit): the user account has reached the folder size limit and the user has tried to upload additional files.

Client Errors

Server does not respond to requests outside company.

If your FTP server is behind a firewall, the network administrator will need to ensure that ports are available for access. Another option would be to configure the specific IP address for the server machine to be outside the firewall.

Folders Not Visible in FTP Client

Try changing the server response.

Can't Download WebSTAR Files

WebSTAR checks the *Mac OS Creator Code* before allowing users to download files. As a security feature, any files created by WebSTAR, with the *Creator Code* "WWWOmega", will not be transferred. This prevents accidental or malicious downloading of server settings files.

The *Kerio WebSTAR Tomcat* plug-in allows *WebSTAR* to pass data to and from the *Tomcat* server, which serves Java servlets and JavaServer Pages (JSP). For more information on running *Tomcat*, please see the documentation included with the *Tomcat* distribution. You configure *Tomcat* using the *Admin Client* application.

12.1 Configuration

First of all, all of your servlets and JSP documents will need to be located on the *Tomcat* server, in its directories, and not with *Kerio WebSTAR*. *Tomcat* needs to be able to reach these files locally. Please refer to the *Tomcat* documentation for more information on adding servlets, JSP documents, and creating zones.

The *Tomcat* plug-in needs some information about your *Tomcat* server in order to communicate with it. This information needs to be provided in the *Tomcat* panel in the *Web Server* topic.

To configure *Tomcat* using *Admin Client*:

1. Click the *Tomcat* item in the *Web Server* topic within the *Admin Client* application.
2. Click *New* to create a new row in the *Tomcat Server Connections* table.

The *Admin Client* populates the new row with placeholder values.

You can configure your plug-in to communicate with multiple *Tomcat* servers depending on zone and virtual host name. Four elements are required for each configuration.

- *Host* (e.g. *www.myhost.com*)

This is the virtual host that the client is contacting. Providing different host names allows you contact different *Tomcat* servers, or different servlet zones on the same *Tomcat* server depending on which virtual host the client is contacting. The virtual host you choose must be configured in *Kerio WebSTAR*.

- *Prefix* (e.g. *examples*)

This is a prefix in a URL that *WebSTAR* uses to register that you want to contact the *Tomcat* server. It is convenient to associate this with servlet/ JSP zones. These

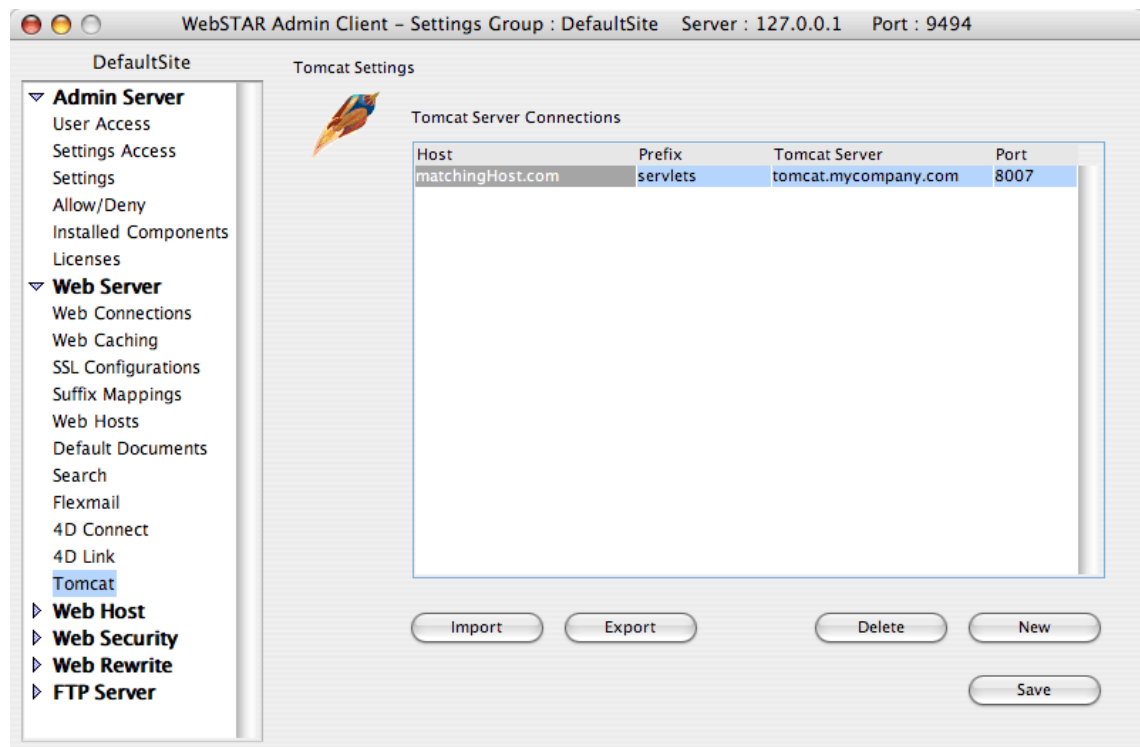


Figure 12.1 The Tomcat panel

zones generally relate to the folder in the *Tomcat* server where the servlet you seek is contained. For more information about zones, please refer to the *Tomcat* documentation. The default installation of *Tomcat* provides a pre-configured zone called *examples*. If you have installed *Tomcat* but not configured it, you may test it with the *examples* zone. For *WebSTAR*, this setting is just a prefix; it means you want to contact *Tomcat*. *Kerio WebSTAR* sends the rest of the URL to *Tomcat*.

- *Tomcat Server Address (e.g. 10.99.1.44)*

This is the address of the machine you installed *Tomcat*. It can be a URL, or an IP address. If you are running *Tomcat* on the same machine as *WebSTAR*, you can use *localhost* as the address here.

- *Tomcat Server Port Number (e.g. 8007)*

This is simply the port on which you will be contacting the *Tomcat* server. *Tomcat*'s servlet/JSP server port number is configurable, but by default it is 8007. If you have changed it, you need to put the current port in this field. If you have installed *Tomcat* but not configured it, then it should be serving on port 8007.

12.2 Quick Start

Tomcat provides servlets and a zone in its default configuration. If you have just installed *Tomcat*, you can test it with its default configuration.

All you need to do is try to connect to *WebSTAR* with a browser using the correct URL.

For example, if the hostname that you configured was `www.myhost.com`, then the URL will be:

`http://www.myhost.com/examples/servlet/HelloWorldExample`

For another example, use:

`http://www.myhost.com/examples/servlet/SnoopServlet`

To test JSP, use:

`http://www.myhost.com/examples/jsp/snp/snoop.jsp`

Check the servlets and JSP folders in your *Tomcat* directory for more servlets and JSP files that were provided in the default installation of *Tomcat*. Note that the URL after your virtual host is the folder path from *Tomcat*'s root to the servlet you are requesting. You can drop your servlets into those folders to test them, but you will need to restart *Tomcat* each time you add servlets and/or JSP documents.

Chapter 13

WebSTAR Flexmail

WebSTAR Flexmail (formerly Net.Dreams Flexmail and Net.Dreams MailAgent) is a form-processing utility. Administrators can generate multiple customized email messages from form data, save form data to disk, and write new HTML files to disk based on form input.

Several *Flexmail* examples are included in the *Flexmail* folder in the *Examples* folder in *DefaultSite*, and are very useful for reference.

13.1 Configuring Flexmail

Flexmail is configured using the *Flexmail* panel in the *Web Server* group in the *Admin Client* application.

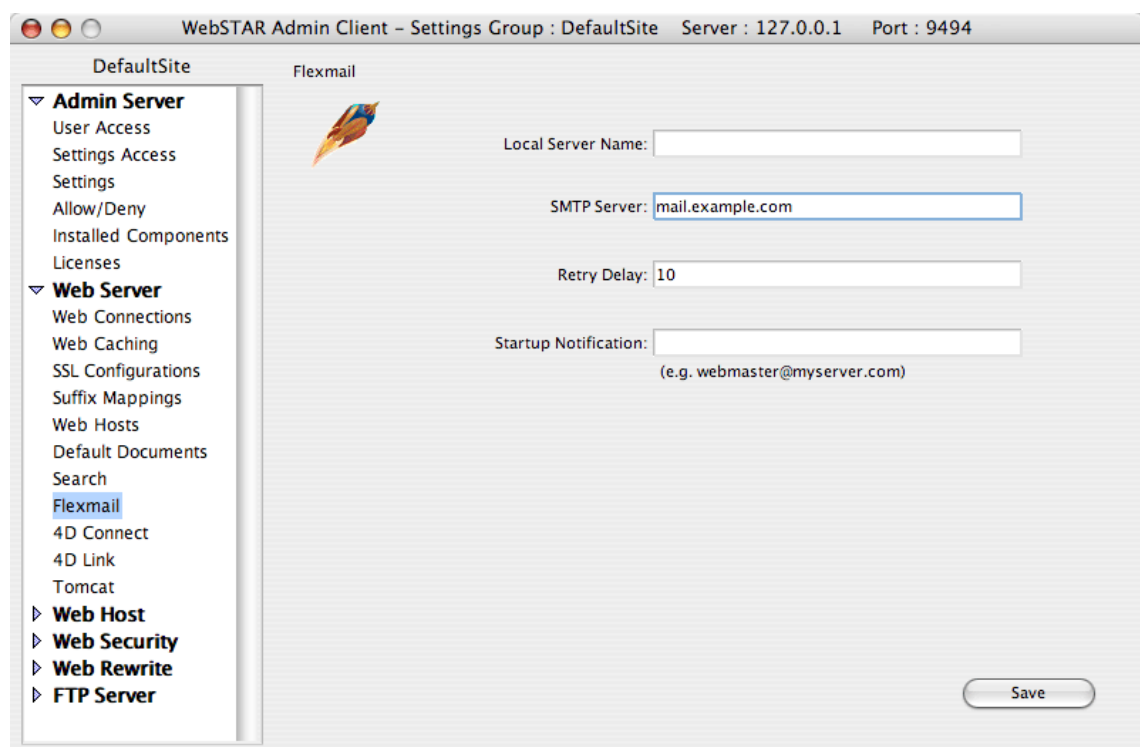


Figure 13.1 The Flexmail panel

The *WebSTAR Flexmail Data* folder in your *WebServer* folder contains the *Mail Log* document that tracks the numbers of messages processed in total, messages successfully sent, failed send attempts, and pending messages still in process.

This folder also contains two (initially empty) folders called *PendingMail* and *Lost Mail* whose roles are explained below.

The *Flexmail* panel in the *Admin Client* application contains four settings:

Local Server Name

This is the hostname or primary IP address of the machine running *Kerio WebSTAR 5*.

SMTP Server Name

This is a mail host via which all outgoing mail is sent. If you do not have your own mail server, you may be able to use your ISP's SMTP host.

Retry Delay

The retry delay gives the amount of time between attempts to resend a message that *Flexmail* could not successfully send on the initial attempt. After several attempts, the message is moved from *Plug-Ins:WebSTAR Flexmail Data:Pending Mail* to *Plug-Ins:WebSTAR Flexmail Data:Lost Mail*.

Startup Notification

You can use *Flexmail* to send a notification to the specified email address whenever *Kerio WebSTAR* starts up. Enter the email address to which the automated message is to be sent.

13.2 Flexmail Forms

The *Flexmail* plug-in will activate any time the server receives a request for a file with a name ending in *.flx*. The *Flexmail* plug-in takes any form data received from the client and processes it according to your instructions in the *.flx* file. In the *.flx* file, you can:

- Make sure that certain fields are filled out, and return an appropriate error message if they are left blank.
- Return a customized response to the user, including information the provided in the form.

- Instead of returning an HTML page to the user, you can redirect them to another URL.
- Save the posted form data to a file in any text format, including HTML or tab delimited text.
- Send an email message that is formatted in any way you like.

Example .flx files for various applications are included with both the demonstration and full versions of *Flexmail*.

Creating a Form

There are a few things to remember when building forms for use with *Flexmail*:

- Always use the POST method.
- Give every form element a name. If you don't name an item, its value won't be returned to you for processing.
- Element names should only contain alphanumeric, non-whitespace characters.

Flexmail Replacement

Any string of non-whitespace characters surrounded with % signs in your .flx file will be replaced with an item from the form. For instance, if your form has the following input box:

Enter your email address:

The following code within your .flx file will be replaced with the contents of that field:

%email%

There are a few built-in items that *Flexmail* provides to give you some information about the client accessing the Flex file:

- %_Referer% contains the HTML file that the person used to get to this Flex file.
- %_UserName% contains the log-in name of the user, provided s/he had to use a username to get to this area of your server.
- %_UserAgent% is the type of browser being used.
- %_Address% is the host name or IP address of the user's computer.

- `%_Date%` is the current date (e.g.: Mar 3, 1998).
- `%_Time%` is the time at which the `.flx` file was executed (e.g.: 12.35.00 PM).

Note: These last two are formatted without colons or slashes to allow their use as parts of filenames in `<savecustom>` tags.

Required Fields

The `require` tag specifies any fields that must contain data before the form will be processed. For example, you might require that users fill out the element named `email` in order for the form to be processed.

Example

```
<require email>
```

To require more than one field, simply add it to the `require` tag:

```
<require email name>
```

If a required field contains no data, the `<mail>` and `<response>` tags are ignored and the contents of the `<error>` tag is returned instead.

For Example:

```
<ERROR>You did not fill out all required fields.  
Please go back and retry.</ERROR>.
```

Responses

The `response` tag contains the HTML that will be returned to the user. Like the `<mail>` tag, it can contain replacement items from the form.

Example

```
<response>  
<center><h1>Thanks</h1></center>  
Thank you for filling out our feedback  
form. You said:<br>  
%comment%  
</response>
```

In this example, whatever the user entered into the form item named `comment` will be included with the response they see.

Redirecting

Instead of returning an HTML response to the user, you can redirect the user to another URL.

Example

```
<REDIRECTTO HREF=http://www.some.com/x.html>
```

Saving Data

You can use the `<savecustom>` tag to save the contents of the submitted form to a file.

Here is an example of a save to file in tab-delimited format:

```
<savecustom registrants.txt append>
%name%%emailaddress%%phone%
</savecustom>
```

Example of a save to file in HTML format:

```
<savecustom guestbook.html append>
<hr>
<p>
<b>At %_Time% on %Date% %name% wrote:</b><br>
%comment%
</p>
</savecustom>
```

Unique Filenames

To automatically create a unique file every time the .flx page is processed, use `unique` instead of `append` in the `<savedata>` tag:

```
<savecustom registrants.txt unique>
...
</savecustom>
```

This simply adds a string based on the current date and time to ensure that the file name is unique.

You can also create unique file names by using replacement data in the file name:

```
<savecustom %name%%_time%.txt>
...
</savecustom>
```

Replacing File Contents

To replace the contents of a file each time you process the .flx page, simply do not specify `append` or `unique` in the `<savecustom>` tag:

```
<savecustom registrants.txt unique>
...
</savecustom>
```

Sending Messages

The `<MAIL>...</MAIL>` tag specifies the email output to send, based on the form data. Keep the following points in mind when formatting the email:

- Each message has two parts — the header and the body.
- Each line of the header begins with an entity name. This is a name like `From`, `To`, or `Subject`. A header line should not begin with a space, a tab, or other non-alphabetic character.
- You can include any headers you want in the message. You are not limited to just the headers that *Flexmail* explicitly supports. The `Errors-To` header is especially helpful. If the mail server accepts a message from *Flexmail* but then can't deliver the message, the mail is usually forwarded to the error address.
- The name is followed by a colon and a space.
- There are no blank lines in the header.
- The first blank line in the message is assumed to be the end of the header and the beginning of the body of the message. You must leave a blank line between the mail header and the body of the message.

The message can be as long as you like, and can include information imported from the form.

Example

```
<mail>
To: %emailaddress%
From: webmaster@www.mydomain.com
Subject: Flexmail is Cool
Hey! This stuff really works!
Your comment was %comment%.
</mail>
```

If you don't want to send email from a Flex file, simply omit the `<MAIL>` tag altogether.

13.3 Sending Messages with Other Methods

You can use *Flexmail's* emailing capability from sources other than *Flexmail* forms. The format for messages is the same as the *Flexmail* form email format, with the exception that you cannot use `.flx-style` replacement data, since in these cases the message is simply sent out.

Sending messages from SSI

The following is an example of an SSI page that contains a `<MAIL>` tag. If *Flexmail* is installed, SSI will send it the contents of the `<MAIL>` tag to *Flexmail* for emailing. This can contain SSI tags, but not *Flexmail* form percent-wrapped replacement data.

```
<html>
<head>
<title>CJ's Mail Demo</title>
</head>
<body bgcolor=#f0f0f0>
<h2>Hello! This is a very short web page!</h2>
<mail>
To: you@your.address.dom (Your Name)
From: webmaster@theHouse.com (WebMaster)
Subject: A Visitor Stopped By
Visitor Number <!--#counter var="mail_counter"--> just stopped by.
They were using a <!--#echo var="http_user_agent"--> browser.
</mail>
</body>
</html>
```

13.4 Adding Attachments to Messages

In addition to text, you can also send binary data files through email. You tell *Flexmail* to attach a file to a message by the use of the X-Attach header. For example:

```
To: me@my.dom
From: webmaster@my.dom
Subject: Attachment
X-Attach: "subfolder/afile.gif"; type="image/gif"
You should get an attached file with this message!
```

Note: The file must exist somewhere on the server before the message is processed by *Flexmail*.

When attaching files from *Flexmail* forms or SSI, you can use a relative filename. That is, you can simply specify a filename or a folder/filename path. Flexmail will look for the indicated folders and file within the same folder as the document currently being served by your web server.

You may also use a full path from the current site's root folder:

X-Attach: `"/folder/afile.gif"; type="image/gif"`

When attaching files through *Apple Events*, the filename paths are always relative to your server's root folder.

Specifying a MIME Type

When the email is displayed by the recipient's software, it generally helps if the software knows what kind of file it is trying to use. You can specify the MIME type of the enclosed file with the type parameter. For a list of common file extensions and their MIME types, consult your web server's suffix mappings.

Sending Multiple Files

You can send as many files as you want, and they can be as large as you like. Just keep in mind that large attachments will take longer to process and send, and will require more disk space to store until they are sent. For example:

```
To: me@my.dom
From: webmaster@my.dom
Subject: Attachment
X-Attach: "subfolder/afile.gif"; type="image/gif"
X-Attach: "fileTwo.html"; type="text/html"
X-Attach: "fileThree.sit"; type="application/x-stuffit"
X-Attach: "fileFour.pdf"; type="application/pdf"
You should get four attached files with this message!
```

Chapter 14

WebSTAR Search

Kerio WebSTAR 5 contains a built-in site search engine, allowing visitors to search the content of your site or remote web sites. The search engine can be used to index local content, or as a web crawler (sometimes called robot or spider) to download and index other web sites. *WebSTAR Search* supports the downloading and indexing of text, web pages (HTML), and PDF files, Boolean search operators, relevance ranking, and more.

14.1 Overview of WebSTAR Search

When you add search capabilities to your site, you can present a standard search page such as this:

With a screen such as this, the visitor can enter a search string which is passed to the search engine. *WebSTAR Search* displays the results of the search in a screen such as this:

The format of the listing can be customized. You can control the fields listed within each item, as well as the appearance of the listing (font, color, etc.). You do this by customizing a search template file. The *DefaultSite* example contains such a file; you can use it as-is, or modify it to suit your needs. For more information, see the section [14.6](#).

To enable searching, you need to build an index for each site to be searched. You do this with the *Search* panel in the *Admin Client* application. When you set up your search page (i.e., the HTML for [figure 14.2](#)), you reference that index in your HTML.

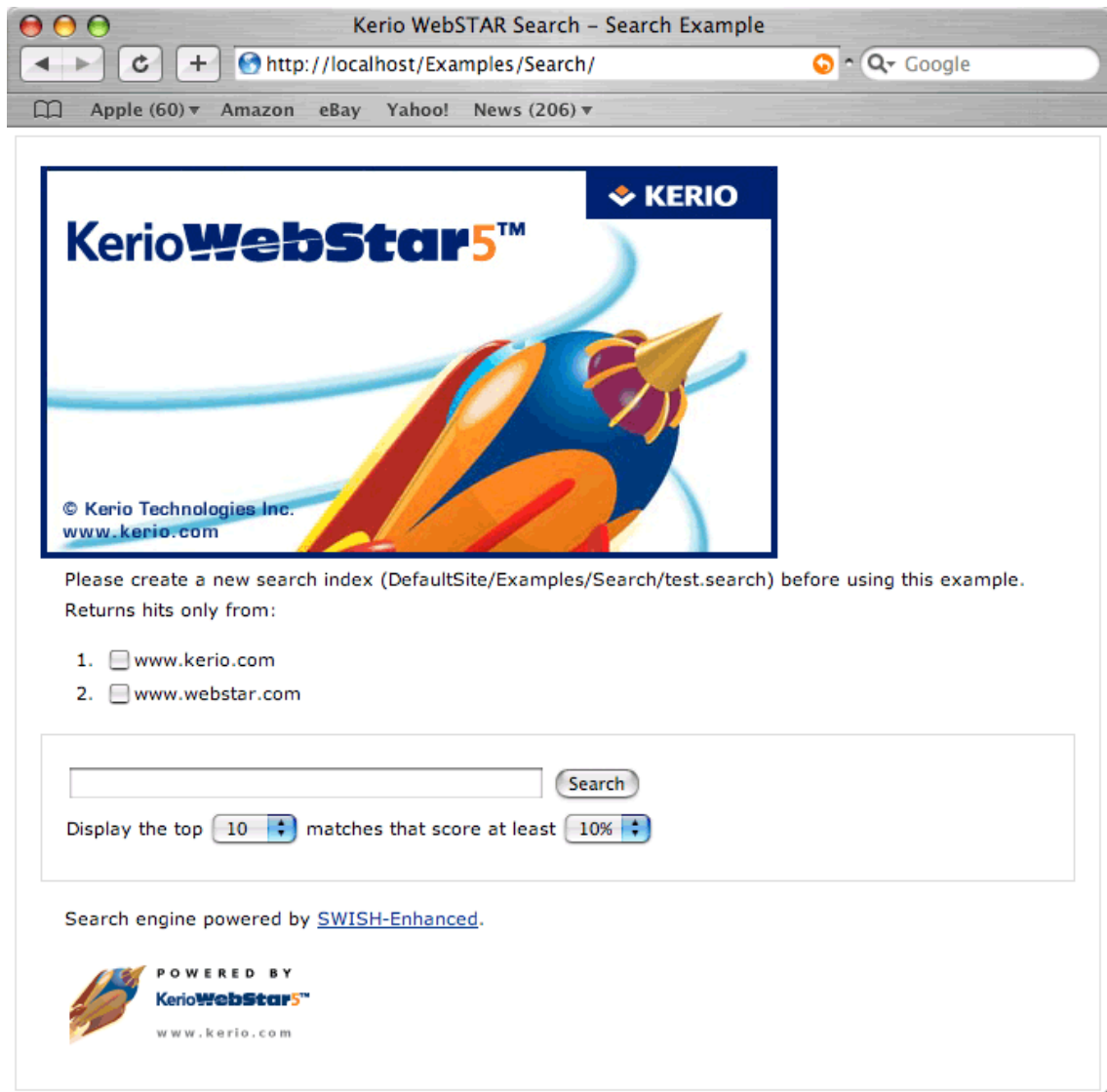


Figure 14.1 A typical Search screen

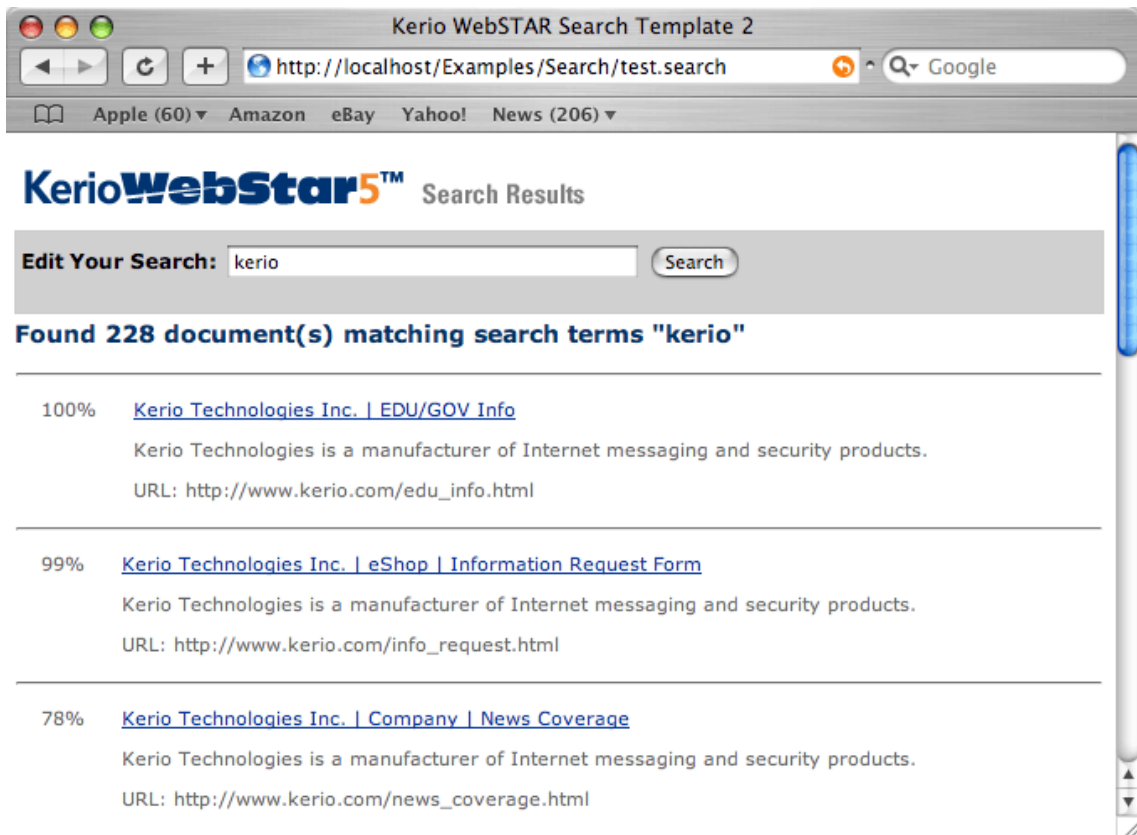


Figure 14.2 A Search results screen

Using WebSTAR Search as a Web Crawler

The *Admin Client* interface for *WebSTAR* search allows you to index either your own virtual hosts or remote sites. As a web crawler (sometimes called robot or spider) *WebSTAR Search* acts as an HTTP-client, locating files by following URL links. You only need to enter the URL of the root directory belonging to the site to be indexed. When you specify a remote site, *WebSTAR* makes a temporary local copy of the files to be indexed.

If you are creating an index for a large site, it is a good idea to limit the number of levels to 3 or less.

The *Search Indexer* will open, analyze, and index each text and PDF file within the root folder. It will not follow aliases to other files or folders.

This may take several minutes to hours, depending on the amount of data on the remote site and the speed of your machine. It will operate, more slowly in the background if you need to use the machine for other purposes.

Note: When crawling remote web sites, *WebSTAR Search* temporarily downloads their contents into the *WS_SearchFiles* folder. Make sure that you have enough hard disk space for these files.

WebSTAR Search uses this URL as the starting point which to follow links from. *WebSTAR Search* will not crawl URLs referenced outside this site; neither will it crawl outside assigned URLs containing subdirectories. URLs which require username/passwords to be accessed can be entered in the form:

`http://username:password@website.com/`

Note: While the *WebSTAR Search* crawler supports the downloading of *Adobe Acrobat* PDF files, it does not support the indexing and summarization of their content.

Robot Exclusion Standard

WebSTAR Search fully supports the robot exclusion standard as defined at: <http://www.robotstxt.org/wc/norobots.html>

Before downloading the content of a web site, *WebSTAR Search* first looks to see if a file named `robots.txt` is present in the root directory of the specified site. If it is, *WebSTAR Search* downloads the file, looks for a user-agent of *WebSTAR* or `*` and adheres to the `Disallow` statements assigned to that user-agent.

Search Index Security

To avoid having a file indexed, move it out of the search root folder hierarchy, rearrange your folder hierarchy, or use an file resource editing application (e.g. *ResEdit*) to change its *Creator Code* to `noIX`, which will indicate to the *WebSTAR Search Indexer* that this file should be skipped.

If you have security realms or confidential information in one of your indexes, name it carefully, and consider protecting it by using a security realm entry. That way, no one can search through it to find private data.

Using the Example as a Template

The *DefaultSite* web site that ships with *Kerio WebSTAR 5* has a worked example of *WebSTAR Search*. You can easily modify this example to work for your site and/or remote sites.

14.2 WebSTAR Search Basics

Before you can perform a search query, you must create a search index. This is a `.search` file that stores data about the files being queried.

A `.search` file should be stored within the *WebServer* folder, preferably in the same folder as the search form page that references it.

Once you have an index, visitors to your site can perform a query using a search form. This is an HTML form with an Action link to the index file. When a user enters search terms into the form and clicks the *Submit* button, the browser sends a form action request to the server. Because the form URL suffix is `.search`, the request is passed to *WebSTAR Search*, which searches the index for the requested terms, and returns the results in a formatted page.

Note: Avoid using the *WebSTAR Search* to search folders that have applications or other non-searchable content. These files may cause problems during indexing.

For security, you should not make your *KerioWebSTAR* or *WebServer* folders the search root. Use the virtual host folders or subfolders within a virtual host. You should also index your secure realms separately from public data.

The DefaultSite Search Example

The *Examples* folder in the *DefaultSite* folder includes search example. Open the *Search* folder within the *Examples* folder. The `index.html` file is the *Search* screen. It contains HTML for the pop-up menus and this form statement:

```
<form method="post" action="test.search"
  enctype="x-www-formencoded">
```

In other words, it references an index file named `test.search`. This is the file you must create using the *Search* panel of *Admin Client* (or use a different filename and make a commensurate change to the file referenced by the action statement). The file must use the `.search` suffix.

We recommend that you follow the *DefaultSite* example and locate the *Search* screen and associated index file inside the same folder as the virtual host it belongs to.

Note: *WebSTAR Search* is implemented at the server wide level, meaning that you must have full admin privileges to access *WebSTAR Search*.

To create a search index:

1. Highlight the *Search* item in the *Web Server* group in the *Admin Client* browser.

The following panel appears.

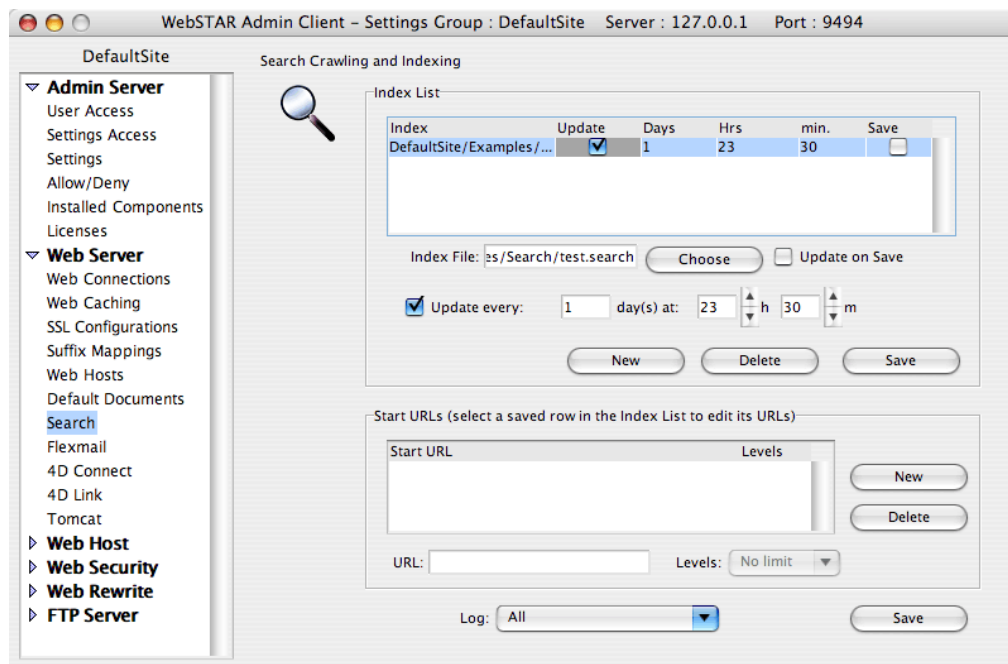


Figure 14.3 The Search Crawling and Indexing panel

2. In the *Index List* area, click *New*.

A new row appears in the index table, with default values. the path in the *Index File* area is the relative path from the root directory of a virtual host to the index file itself. In the case of the example search, the correct path is:

DefaultSite/Examples/Search/test.search

3. Enter the path to the index file you are creating.

It begins with the virtual host's folder name and ends with the name of the index file. This file must use the *.search* suffix.

4. Click the *Update* checkbox to instruct *WebSTAR Search* to update the index periodically.

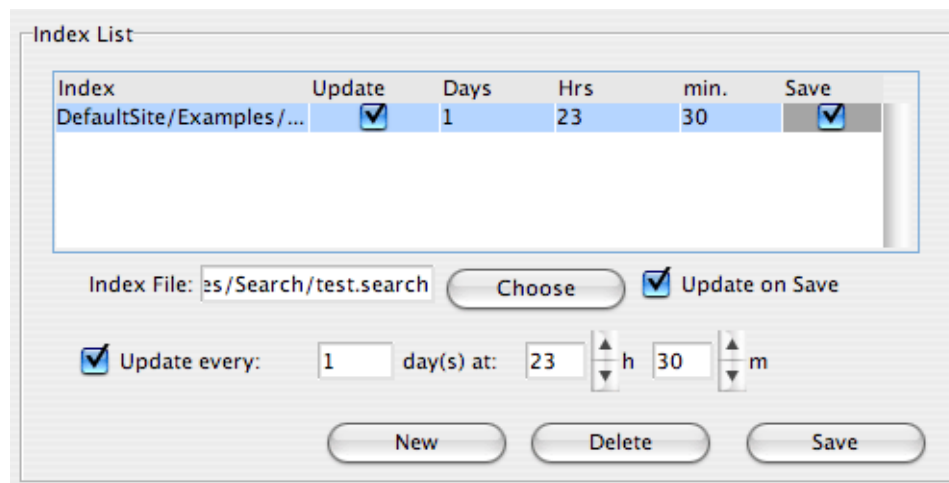
5. Use the day and time (hours and minutes) areas to indicate the time period between automatic updates.

Ideally, the time should be in periods of low traffic on your web sites.

To follow and download a given link, the URL must be an index file or end in a suffix of: .HTM, HTML, .SHTML, .SHT, .ASP, .PDF, .SSI, .JSP, .TXT, .TEXT, .LASSO, .LAS, .TMPL, .TAF, .CGI, .PL, .CFM, .XML or .SGML.

6. Click the *Update on Save* checkbox to instruct *WebSTAR Search* to update the index when you click the *Save* button for this panel.
7. When you are finished, click the *Save* button in the *Index List* area.

A finished *Index List* entry looks like this:



The screenshot shows a dialog box titled "Index List". Inside, there is a table with the following columns: Index, Update, Days, Hrs, min., and Save. The first row contains the text "DefaultSite/Examples/...", a checked checkbox, the number "1", "23", "30", and another checked checkbox. Below the table, there is a text field labeled "Index File:" containing "es/Search/test.search", a "Choose" button, and a checked checkbox labeled "Update on Save". Further down, there is a checked checkbox labeled "Update every:", followed by a text field with "1", the text "day(s) at:", a text field with "23", a dropdown menu showing "h", a text field with "30", and a dropdown menu showing "m". At the bottom, there are three buttons: "New", "Delete", and "Save".

Index	Update	Days	Hrs	min.	Save
DefaultSite/Examples/...	☒	1	23	30	☒

Index File: ☒ Update on Save

☒ Update every: day(s) at: h m

Figure 14.4 An index list entry for the DefaultSite example

Next, you need to tell *WebSTAR Search* the folder or folders to be indexed and the number of levels of subdirectories to be included. You use the *Start URLs* area for this.

To specify the root folder(s):

1. Highlight the row in the *Index List* table for which you wish to specify the root folders.

This identifies a particular *.search* file.

2. In the *Start URLs* area, click the *New* button.

A new row is added to the *Start URLs* table.

3. In the *URL* area, enter the URL of the root folder and the number of levels of subdirectories to be included.

Of course, you can index your own site or create indexes for remote sites. If you are indexing one of your virtual hosts, you use the URL to the root folder of that virtual host.

4. (Optional) Add additional folders for the selected `.search` file, as needed.

For example, if you want to index selected folders within a virtual host rather than the entire virtual host, you can add each folder to the *Index List* rather than the virtual host folder itself. A finished *Start URLs* area for a remote site looks like this:

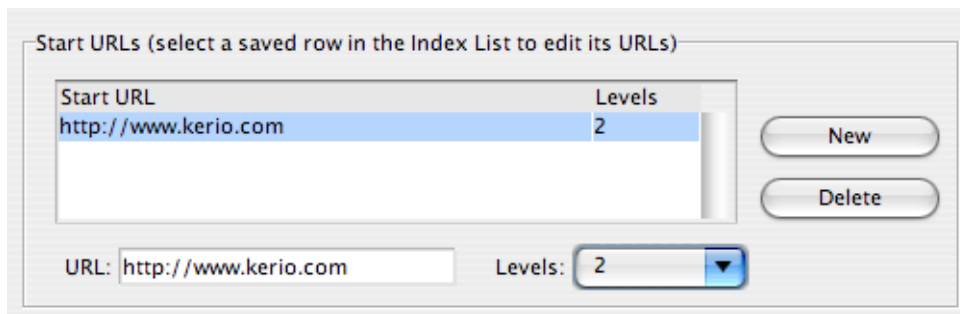


Figure 14.5 A Start URLs entry for a remote site

5. Click *Save* at the bottom of the panel to save these specifications to the server.

Once done, you can try out your index.

Modify the `index.html` template in the *Search* folder within *DefaultSite* so that it references the index you've just created. Display your search page in a browser, enter a query, and see if the search is working.

14.3 About Search Queries

With *WebSTAR Search*, you can enter the words in your query in any order. The search engine will compare them to the data in the index: it will find all files with at least one of the search terms. When it returns the results, *WebSTAR Search* will automatically rank them from most relevant to least relevant.

WebSTAR Search performs a vector search, where every indexed document is stored in the index as a point in a multi-dimensional space (a vector), and the dimensions correspond to terms found in the body of all documents. Normal queries are then interpreted as very short documents, and are also converted into a vector. The search engine then looks at the index for all document vectors within a certain distance of the query vector. Thus, the more that your query resembles the documents you are looking for, the better your results will be. That's why natural language queries, such as questions or descriptions, work so well with *WebSTAR Search*. For example, you could enter the query:

improve performance of web server

and get back results ranked according to how closely they resemble that query.

Adding Boolean Operators to Searches

In addition to the vector searching, you can add Boolean operators to your searches, if you prefer.

- & (ampersand) used between search terms is the Boolean AND operator. It indicates that both terms must be in a document. This is very useful for getting a more relevant set of results. For example, to search for documents that mention both Sherlock and Apple, enter `sherlock & apple`.
- | (vertical bar or pipe character) used between search terms is the Boolean OR operator. It indicates that either term can be in a document, for example, to search for documents that either AIAT or Sherlock, enter `aiat | sherlock`.
- ! (exclamation point) used between search terms is the Boolean NOT operator. It indicates that only documents matching the terms before it and not matching the term after it should be returned. Again, this removes irrelevant results. For example, to search for documents that mention Sherlock but not Holmes, enter `sherlock ! holmes`.

Grouping

Another way to control the query is to explicitly group the terms — control the order of processing.

For example, you may want to link the terms Sherlock and Apple and ignore Holmes. So you can use the grouping brackets [and] to specify how the search should be grouped:

```
aiat | [[sherlock & apple] ! holmes]
```

14.4 Search Forms

To search an index, you must have an interface. The simplest interface is an HTML form. The easiest way to start making such a form is to copy the simple search form file (`index.html`) from the *Search* folder in the *Examples* folder. For testing, just change the form action so that it refers to your index, for example:

```
ACTION="/widgets/widgets.search"
```

Once you have a form, you can open that page on your server and search your index.

Search Form Error

If a visitor sees the message *The search parameters were not received, probably because you pressed Return instead of clicking on the submit button. Go back, and try clicking the button.*, it may also mean that there is no such index. To allow visitors to press the *Return* key in your form, move the *Submit* button out of the table. That way, all the browsers will do the right thing.

Search Links

You can set up URLs and links in pages that passes the query parameters in the search arguments section of the URL. For example, this URL:

```
http://www.domain.com/Tools%20%26%20Examples/Search%20Example  
/Test%20Collection/index.search?query=webstar
```

will search the example index for the word *webstar*, assuming you have created the index and you changed the host name from *www.domain.com* to your host name.

Likewise, the following link, in a file in the *Search Examples* folder, will search the test collections index for the query *web server*:

```
<A HREF="http://Test%20Collection  
/index.search?query=web&20server">
```

Each link can also include parameters, as described below.

Note: Note that the space between *web* and *server* is encoded (%20). You will have to make sure that all special (non-alphanumeric) characters are encoded as part of the URL.

14.5 Search Results

WebSTAR search will automatically display formatted results of your search, as shown earlier:

Note that the links for the results will take you to the beginning of the document containing the matched text. For an exact location, you must use the *Find* function of your browser.

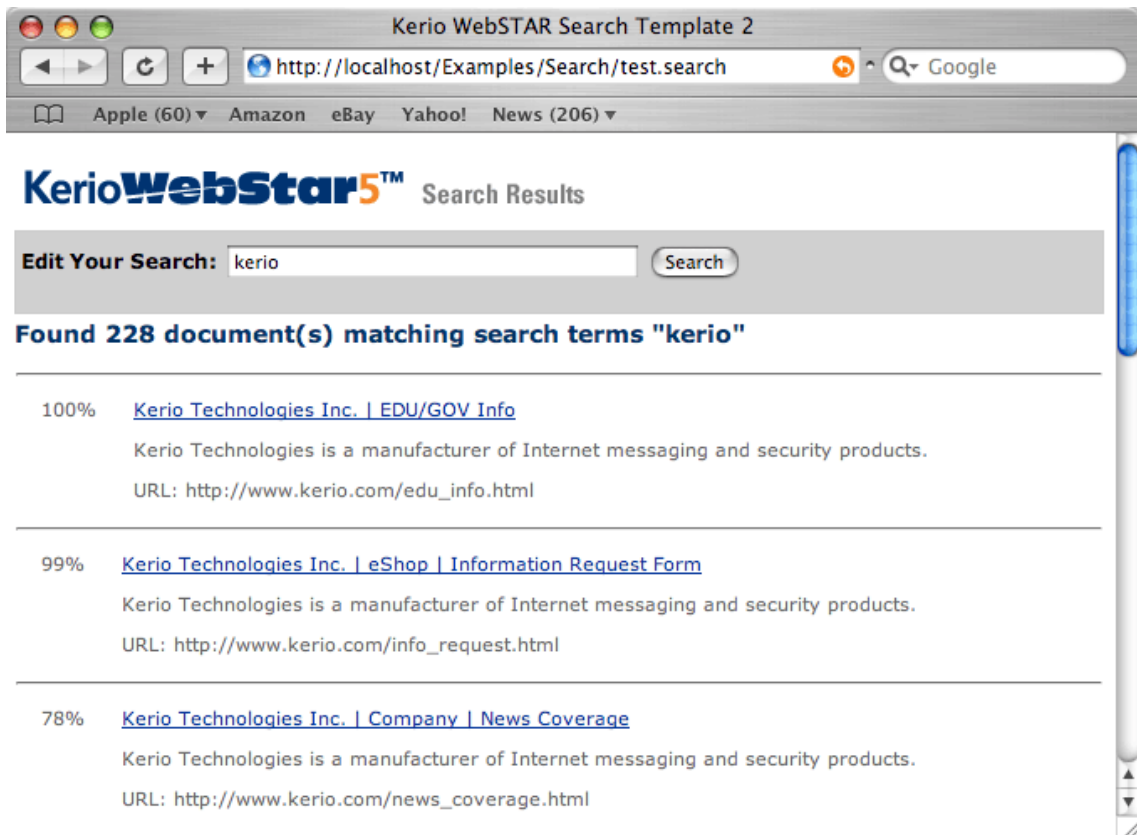


Figure 14.6 A Search Results page using the default template

Search Results Elements

By default, a *Search Results* page shows the following items. This list can be modified by editing the `Default_Search_Results.xml` file for that virtual host. For a description of this file, see the following section [14.6](#).

- *Number Found* — The number of matching results displayed on the page (not the total number of matches). Note that if you do not specify a maximum number of results, *WebSTAR Search* will return no more than 10 matches.
- *Relevance Ranking* — The percentage number in the left column (100%, 58%, etc.) shows the how relevant the document is to your query. This number is calculated by the search engine.
- *File Link* — The name of the matching document, with a link to the file itself. Acrobat files include the document title if there is one.
- *Summary* — For PDF documents which include *Subject* and *Keyword* fields, they are displayed here. For other files, the search engine computes the most important words and phrases from the document and displays them.
- *Matching Terms* — The query words which were matched in the document.

14.6 Customizing Search Results

By default, *Kerio WebSTAR 5* returns search results using the format illustrated in figure [14.6](#). It uses an internal format that is used automatically if no custom format is provided.

To customize the format of the search results, you use a search template. The search template is an `.xml` file that you must reference within the query form for the search.

The search example that is included with the *DefaultSite* virtual host includes a custom template called

`Custom_Search_Template.xml` and it is located in the *Search* folder in the *Examples* folder. The example search uses this template.

To override the default format and use a custom template, you must include an `<input>` statement in your search form that references the template. If you don't include such a reference, the custom template will be ignored, i.e., it is disabled unless it is referenced in a query form.

Since you reference the search template in your query form, you can use a different custom template for each search.

The query form used in the search example in the *DefaultSite* folder contains the following `<input>` statement.

```
<input type="hidden" NAME="template"
value="/Examples/Search/Custom_Search_Template.xml">
```

The NAME keyword must be equal to `template` and the VALUE keyword specifies the path of the template file from the virtual host root folder.

For example, if you move the file to the root level of the *DefaultSite* folder, the path would be:

```
/Custom_Search_Template.xml
```

You can name the template file whatever you wish as long as you use the suffix `.xml` (and it is a valid XML file).

Note: If you want to ignore the template file, simply delete the `<input>` line and *WebSTAR Search* will use the internal default format.

To modify a template, reference it in your query form and make changes to it using any text editor. Save your changes to disk and then flush the web server's cache by clicking the *Flush Cache* button on the *Web Caching* panel in the *Admin Client* application (see figure 17.5).

After you flush the cache, use a browser to reload the query form and re-execute your search.

Search Template Sections

The search template is divided into six sections, some of which may include variables (described below):

- `<RESULTS_PREVIOUS>` and `</RESULTS_PREVIOUS>` — Used to define the text to use in *Previous Results* link.
- `<RESULTS_NEXT>` and `</RESULTS_NEXT>` — Used to define the text to use in *Next Results* link.
- `<RESULTS_HEAD>` and `</RESULTS_HEAD>` — Used to define header and form results to return. Supports the following variables (described below): `###SEARCH_INDEX`, `###NUMBER_DOCS`, `###MINIMUM_SCORE`, `###RESULTS_HIT`.
- `<RESULTS_HIT>` and `</RESULTS_HIT>` — Used to define the format of the actual query results. Supports the following variables (described below): `###RELEVANCE`, `###URL`, `###TITLE`, `###SUMMARY`.

- `<RESULTS_FOOT>` and `</RESULTS_FOOT>` — Used to define footer to return at bottom of results. Supports the following variables (described below): `###RESULTS_PREVIOUS`, `###RESULTS_CURRENT`, `###RESULTS_NEXT`, `###PREVIOUS_SEARCH_QUERY`.
- `<RESULTS_FAILURE>` and `</RESULTS_FAILURE>` — Used to define the page to return when no matching items were found. Supports the following variables (described below): `###SEARCH_INDEX`, `###NUMBER_DOCS`, `###MINIMUM_SCORE`, `###PREVIOUS_SEARCH_QUERY`.

Search Variables

To display results specific to a particular search query, you can insert variables into the search template. Search variables include:

- `###SEARCH_INDEX` — Defines the path name to the index file to be used in the search, as for use in the Form Action.
- `###NUMBER_DOCS` — Defines the maximum number of matching documents to return.
- `###MINIMUM_SCORE` — Defines the minimum match score for documents that are to be returned in the results.
- `###RESULTS_HIT` Defines the number of documents returned for display on the results page (not the total number of matches).
- `###RELEVANCE` Defines the relevance ranking, showing how relevant the document is to the query.
- `###URL` — Defines the URL of the found document.
- `###TITLE` — Defines the title of the found document.
- `###SUMMARY` — Defines the summary of the found document.
- `###RESULTS_PREVIOUS` — Defines a link to the previous results displayed. This variable may not always be displayed.
- `###RESULTS_CURRENT` — Defines the number range of results currently displayed.
- `###RESULTS_NEXT` — Defines a link to display the next results. This variable may not always be displayed.
- `###PREVIOUS_SEARCH_QUERY` — Defines the terms used in the previous search query.

14.7 Search Troubleshooting

Search Form Error

If a visitor sees the message *The search parameters were not received, probably because you pressed Return instead of clicking on the submit button. Go back, and try clicking the button.*, it may also mean that there is no such index. To allow visitors to press the *Return* key in your form, move the *Submit* button out of the table. That way, all the browsers will do the right thing.

Chapter 15

WebSTAR Image Map

Image maps let you specify hot spots in the graphics that you serve. This allows you to create hyperlinked maps, logos, and other graphics. You do this by mapping regions within the image to URLs. Not surprisingly, this mapping is stored in a `.map` file. Many HTML editors allow you to create this `.map` file automatically.

The *WebSTAR Image Map* plug-in supports server-side image maps, not client-side image maps. *WebSTAR Image Map* supports the following NCSA methods in the map file, giving you flexibility in defining the regions that are hyperlinked.

- Circle
- Poly
- Rect
- Point
- Default

The *Image Map* plug-in is compatible with NCSA format, but not with CERN format, so if you are using an HTML editor or an image-map generator, be sure that the preference is set to the NCSA format.

You do not need to do anything to administer the *WebSTAR Image Map* plug-in. There is no interface in the *Admin Client* application, but you can use the *Installed Components* panel in the *Admin Server* group to check that it has loaded and is running properly. If so, you will see an entry for *WebSTAR Image Map*.

The *Image Map* plug-in will be called when you receive a URL that ends in `.map`. If you want to use other suffixes or files without suffixes, you can add the appropriate entries to the *Suffix Mapping* table and use MAPPY as the action. For more information, see chapter 18.

15.1 Using the WebSTAR Image Map Plug-In

To use the *Image Map* plug-in, you provide both an image and a `.map` file. The `.map` file contains descriptions of each region within the graphic and links each region to a URL.

Each region is described by specifying the shape of the region and the coordinates that describe the boundaries of the region. The format changes slightly depending on the shape of the region. *WebSTAR Image Map* plug-in supports the following shapes:

- **Circle** — The coordinates are the center of the circle and an edge.
- **Poly** — Each coordinate is a vertex of the polygon. You can use up to 100 vertices.
- **Rect** — The coordinates are the top, left, and lower, right corners.
- **Point** — The coordinate is the point.
- **Default** — Specifies the default URL. There are no coordinates. Use this to specify the page to jump to if the user clicks on the graphic in an area that is not enclosed by any of the specified regions.

The `.map` file includes one statement per region. The statement specifies the type of shape, the URL (which can be either local or absolute), and the coordinates. That is, each statement follows the format:

```
Shape URL coord1 coord2 ...
```

Example

The *Examples* folder within *DefaultSite* contains a simple example of the use of the *Image Map* plug-in. It is in the *ImageMap* folder.

The image, shown in figure 15.1, contains logos for both *WebSTAR* and *Apple Computer*. The region bounded by each logo is linked to the home page of each company's web site.

You need to indicate that a graphic is a clickable map in your HTML. This example uses the following link in its `index.html` page:

```
<BODY>
<A HREF="poweredbywebstar.map">
<IMG SRC="poweredbywebstar.gif" BORDER="0" ISMAP></A>
</BODY>
```

In the `.gif` file, the *Powered by WebSTAR* logo links to the *WebSTAR* site, and the Mac OS logo links to the Apple site.



Figure 15.1 The graphic used by the Image Map plug-in example

The image file is named `poweredbywebstar.gif` and the `.map` file is `poweredbywebstar.map`.

The `.map` file is very simple, since it only needs to specify the two rectangles:

```
# Format: NCSA
default /
rect http://www.apple.com/ 218,1 272,54
rect http://www.webstar.com/ 0,0 218,54
```

The `#` is a comment saying that the map file uses the NCSA image map format.

- All clicks on non-specified locations will go to the default page for this site.
- Clicks in the rectangle defined by 0 at the left, 0 at the top, 218 at the right and 54 at the bottom will go to *WebSTAR*.
- Clicks in the rectangle defined by left, top at 218, 1 and right, bottom at 272,54 will go to *Apple Computer*.

To link to an image or map file in another folder, just use the file path.

For more information on image maps, see the NCSA tutorial at:

<http://hoohoo.ncsa.uiuc.edu/docs/tutorials/imagemapping.html>

Chapter 16

SSL Web Security

SSL (Secure Sockets Layer) is an Internet protocol that specifies how to pass secure communications between your server and HTTP clients, such as browsers. This provides world-class security for your web site, allowing you to accept credit card numbers, serve medical records, human resources information, or other sensitive data without fear of interception.

16.1 About SSL Security

Kerio WebSTAR 5 includes full implementation of the Secure Sockets Layer 2 and 3 protocols and TLS 1.0 (Transport Layer Security) as part of the server. This allows you to serve specific pages which are encrypted when sent to the browser. Any form responses, such as private personal data, are also encrypted when they are sent back to the server. To invoke it, simply create URLs that use the https (secure HTTP) protocol. For example, you can protect a credit card acceptance URL so that the information is automatically encrypted by the browser before it is sent.

Secure Sockets Layer (SSL) Protocol

WebSTAR SSL provides server authentication, data encryption, and message integrity. These combine to keep the HTTP data truly secure as it travels across the Internet.

The secure sockets encryption technology developed by Netscape. For more information about the SSL 3.0 protocol, see:

<ftp://ftp.isi.edu/in-notes/rfc2246.txt>

SSL is layered below application protocols such as HTTP and above TCP/IP. When SSL is implemented in both the client and the server software, all data is encrypted before it is transmitted.

Kerio WebSTAR SSL supports SSL versions 2 and 3, TLS 1.0, and is based on the OpenSSL library.

Encryption Levels

The *Kerio WebSTAR SSL* server uses an ISO X.509 Certificate that is authenticated with RSA public key cryptography (encryption). With public-key cryptography, the server publishes a key and browsers contacting that server encrypt their messages

using that public key. Only the server can decrypt the messages, because only the server has the private key. For more information, see:

<http://www.rsasecurity.com/products/>

The client and server use public-key cryptography to exchange randomly generated session keys that are used to encrypt data for that connection.

Kerio WebSTAR 5 SSL implements the ciphers described below.

- *RC4-128* — The 128-bit version of the RC4 stream encryption algorithm provides very high-level encryption protection.
- *IDEA-128* — This cipher uses 128-bit keys and is frequently used in banking transactions. According to Netscape, using IDEA-128 to encrypt messages is 309,485,009,821,345,068,724,781,056 times safer than using 40-bit encryption. In the USA and in Europe, a license from Ascom AG is required to use this cipher.
- *3DES* — Triple DES, encrypted three times with three different DES keys. This is much more secure than DES and slightly less secure than RC4-128 (and much slower to process).
- *DES* — DES encryption is required by some United States government agencies. Allowing only a 56-bit key, DES provides less protection than RC4-128. It is also slower.
- *RC4-40* — This cipher uses a 40-bit key size for the RC4 stream encryption algorithm. Although this version contains a lower level of encryption protection, RC4-40 is still a formidable cipher.
- *DES-40* — This cipher uses the same algorithm as the DES encryption, and takes the same amount of time to process, with a 40-bit key.
- *MAC-only (no encryption)* — This provides message authentication only: it proves to the browser that the page was served by your server, but does not encrypt the HTML, so anyone monitoring your data stream can see the page.

The server negotiates to the encryption level understood by the accessing browser.

How Browsers Access a Secure Server

Once you have installed and configured SSL in the *WebSTAR*, clients can connect to the secure server using a URL that starts with `https` instead of `http`. The letter “s” at the end of the code `https` indicates that the connection will be secure.

For example the following URL will bring up a secure page (if you replace `domain.com` with your host, and have a file in this path).

`https://www.domain.com/test/secure.html`

Behind the scenes, the browser automatically connects to the correct port (usually port 443), and performs a certificate negotiation. In this phase, the browser sends a message

explaining what encryption ciphers it recognizes, and the server uses the strongest cipher it has available. If you have them all enabled, they are evaluated in the following order:

1. RC4-128
2. IDEA-128
3. 3DES
4. DES-56
5. RC4-40
6. DES-40
7. MAC (Authentication, no encryption)

You can turn these ciphers on and off in the *SSL Configurations* panel, but you can't change the order.

Once it has found the strongest cipher they can both use, the server sends its certificate (proving who the owner is) and uses that to create the encryption key. This makes the secure connection.

Then the browser encrypts data it sends to the server — such as credit card numbers — and decrypts data it gets from the server, such as medical data or bank records.

SSL and Multiple Domains

You can add an SSL certificate for each IP address on your server. In addition, the host name must be specified using an A *Name* entry in the DNS record: it must not be a CNAME alias.

However, the certificate authorities generally charge a significant amount of money per host certificate, per year. Therefore, a good solution is to obtain a domain name that, with a relevant host name, will seem appropriate to your clients and online users in the SSL URL. For example, you can get domain name like `secureserver.com`, then you could get a certificate for the host name `safe`, so your shoppers will see a URL like this:

`https://safe.secureserver.com/knickknacks/OrderForm.html`

Remember that the HTTPS URL will only be used for the few web pages where private data is collected or displayed for verification. It never has to be promoted, remembered, or even displayed on a web page, since your SSL pages only need be accessed from a link on another web page.

Certificate Chains and Intermediate Certificates

Some large institutions have authorization power: they can sign internal certificates. Because they are between the Certificate Authority and the server certificate, they are called intermediate certificates.

If you get a certificate in this way, it's part of a certificate chain: It includes not only the subject certificate (for your host), but also the root certificates and intermediate certificate information.

For example, a Certificate Authority could sign a certificate for `Domain.com` that authorizes them to sign web server certificates and personal/email certificates. `Domain.com` could then use that certificate to sign all of their internal SSL server certificates and certificates for their employees.

WebSTAR SSL supports these features, but intermediate certificates are very expensive.

Wildcard Certificates

Thawte allows you to generate wildcard certificates, (using an asterisk instead of the host name, as in: `*.domain.com`), with *Kerio WebSTAR 5*. However many versions of *Microsoft Internet Explorer* will display an error message to the user when they connect to one of these certificates.

Therefore, you should only use wildcard certificates in circumstances where you are confident that most browsers will be version 4 and later, and you are prepared to reassure people with earlier browsers that their data is truly secure.

16.2 Configuring SSL

Before you enable SSL in your server, you should consider how it will interact with regular serving of unencrypted pages. In most cases it is not a good idea to use HTTPS for all of your pages. Because SSL encrypts each page before transmitting it, clients may notice that it is slower than the HTTP interchange. In addition, some older web browsers do not support encryption and will not be able to access documents served via the HTTPS protocol.

Steps to Secure Serving

To start serving your website securely, follow these overall steps:

- Check your DNS records to be sure you know the authoritative host name and the exact IP address.
- Generate a Private Key file for this host.
- Generate a Certificate Request for this host.

- Choose the Certificate Authority, and follow the authority's directions to buy a certificate. A certificate proves that you are authorized to use the business or institutional name.
- Add your certificate and key information to the *SSL Configurations* panel in *Admin Client*.
- If you have private data areas, set up a security policy with realms and passwords.
- Test your secure uploads and private areas carefully.

To access your secure pages, clients will need a browser that supports SSL, such as *Netscape Navigator* or *Internet Explorer*.

SSLTools Folder

The *SSLTools* folder in the *KerioWebSTAR* folder contains two folders that store keys and certificates, keys and *ServerCerts*. The *Admin Client* application automatically places the private key files in the *Keys* folder and you install certificates that you receive from the certificate authority in the *ServerCerts* folder.

16.3 Obtaining and Installing Certificates

To operate securely, SSL requires a certificate (sometimes called a digital certificate or Digital ID). A certificate is issued by a trusted third party known as the Certificate Authority.

From a user's perspective, the certificate signifies that an independent party (such as *VeriSign* or *Thawte*) has verified that the information in the certificate accurately represents who it claims to represent, and that communications can be encrypted using the certificate's public/private key. This is similar to the Department of Motor Vehicles verifying that the picture on your driver's license corresponds to your name, birth date, and address. The certificate ensures that the user is actually communicating with your host domain name, not with an imposter who claims to be you.

Certificate Formats

Kerio WebSTAR 5 is compatible with PKCS (the most common standard), Server Gated Cryptography and Netscape formatted certificates.

Certificate Authorities

The two most popular Certificate Authorities issuing certificates for *WebSTAR SSL* servers are *VeriSign* and *Thawte*. For information, we recommend that you visit both web sites.

- *VeriSign*: <http://www.verisign.com/> (see section 16.4).
- *Thawte*: <http://www.thawte.com/>, (see section 16.5).

16.4 Creating Certificate Signing Requests

The procedures for creating private keys and generating *Certificate Signing Requests* are the same for both *VeriSign* and *Thawte*.

- Generate a *Certificate Signing Request* (CSR).
- Submit the paperwork and CSR to the certificate authority, with payment.
- Receive the certificate.
- Install the certificate.
- Store these vital materials in a safe place.

Generate a Key

Use the *Generate Keys and CSRs* item in the *Tools* menu to generate a key. Follow these steps:

1. Choose *Generate Keys and CSRs* from the *Tools* menu.

The following screen appears.

2. Enter name of the key and password in the *Generate Key* area.

You'll need it later to authorize *WebSTAR SSL* to use your public/private key pair. Do not forget your password! If you do, the private key cannot be recovered: there is no back door to this security.

Write the password down and store it in a secure place, such as a safety deposit box. If you lose the password, you will have to purchase a new certificate.

Make sure that the password is at least 8 characters long, includes letters, numbers and punctuation, and is not a name or a word.

DefaultSite : Generate Keys and CSRs

Generate Keys and Certificate Signing Requests

 Generate Key

Name: Bits:

Password:

Generate Certificate Signing Request

This information describes your Distinguished Name. It must match the requested format or you will have to resubmit your Certificate request.

Common Name:

Organization: Org. Unit:

City/Locality: Country Code:

State/Province: (2 letter code, e.g. "US")

Phone Number: Fax Number:

Email Address:

Key: Password:

Output (for Certificate Authority):

Figure 16.1 The Generate Keys and CSRs dialog box

3. Click the *Generate Key* button to generate your *Private Key* file.

A key will be created which uses the specified name and password. It will be saved into the *Keys* folder located inside the *SSLTools* folder in the *KerioWebSTAR* folder.

Create the Certificate Signing Request

The Certificate Signing Request process requires that you supply an email address and certain identifying information, as described here.

To create the certificate signing request:

1. Type the host name of your web site in the *Common Name* field, for example:

`www.domain.com`

Make sure that the *Common Name* you specify will be the actual host name of your SSL server — it will be encoded into the signed certificate and cannot be changed later without purchasing a new certificate.

Furthermore, this host name should be the main *A name* entry for your machine on your DNS server. Your certificate may have problems if you use an IP address or if the host name is a *CNAME* entry (DNS alias), for example. Contact your network administrator for guidance if necessary.

2. Type the name of your organization in the *Organization* field, for example:

`Widgets, Inc.`

3. Type the name of the department or other organization unit in the *Org Unit* field, for example: `Support`

You must enter something in the *Org. Unit* field, if only `store` or `security`.

4. Type the name of the city or town in which your organization is located in the *City/Locality* field, for example: `Frostbite Falls`

5. Type the name of the state or province in which the organization is located in the *State/Province* field, for example: `California`

Do not abbreviate the state or province name: spell it out as shown above.

6. Type a two-letter code for the country in which you are located in the *Country Code* field, for example: `us`

Do not spell out the country name: use a 2-letter code. The code for the United States is `US`. For Canada, the code is `CN`. For other codes, contact *VeriSign* at www.verisign.com.

7. In the *Phone Number* field, type a phone number where the certificate authority can reach you, and type your fax number in the *FAX Number* field.

8. Type the email address of the site's Webmaster or administrator in the *Email Address* field, for example: `webmaster@domain.com`

The *Email Address*, *Phone Number*, and *FAX Number* fields are not part of the certificate. The certificate authority uses these fields to contact you if it finds a problem with the certificate request.

9. Use the *Key* pop-up menu to choose the *Private Key* file you created.
This pop-up will be populated only after you click the *Generate Key* button in the *Generate Key* area of the dialog box.
Once you select a private key file, the key file and the certificate you will receive will be a signed certificate pair, and cannot be separated.
If you lose the private key file and generate a new one, your certificate will no longer match. You will have to send a request to the certificate authority for a new certificate, which you may be charged for. For this reason, be sure to keep backup copies of your file in a secure location.
 10. Enter the password required to access your public/private key pair (the password you entered when generating the key pair, as described in section 16.4).
 11. Normally, choose *Normal Certificate* from the pop-up menu.
Normal refers to a certificate issued by an commercial certificate authority such as *Verisign*. A Self-Signed certificate is one that is issued by your own company.
 12. Click the *Generate CSR* button to generate your encrypted certificate request form.
The application creates a the certificate request in the *Output* area. This is the file that you submit to the certificate authority of your choice. You can copy it by placing the text insertion cursor in the *Output* area, pressing **command-A** (Select All) and then **command-C** (Copy). Then paste it into a text editor program such as *BBedit*.
- Don't edit the Certificate Request in any way and do not save it in any format other than text only.
- A completed form looks like this:
- At this point, the private key file will have been placed in the *Keys* folder in the *SSLTools* folder, under the name you gave it in the *Generate Key* area of this screen. Your Certificate Signing Request may be rejected if the information is not properly formatted. Be sure to closely follow the conventions outlined in the instructions. If any of the information is improperly formatted, the Certificate Authority will ask you to correct it and send the request again.

16.4 Creating Certificate Signing Requests

The screenshot shows a web application titled "DefaultSite : Generate Keys and CSRs". It has two main sections: "Generate Key" and "Generate Certificate Signing Request".

Generate Key Section:

- Name:
- Bits:
- Password:
- Generate Key button

Generate Certificate Signing Request Section:

This information describes your Distinguished Name. It must match the requested format or you will have to resubmit your Certificate request.

Common Name:

Organization: Org. Unit:

City/Locality: Country Code:

State/Province: (2 letter code, e.g. "US")

Phone Number: Fax Number:

Email Address:

Key: Password:

Normal Certificate Generate CSR button

Output (for Certificate Authority):

```
-----BEGIN CERTIFICATE REQUEST-----
MIIBIjANBgkqhkiG9w0BAQwvJCPwA1IghKANTZKE7/1ZUCB+SZqno0ed1PM
R5Z5Um+QNR7eMo1Xwh4PkdF8zIyeQoIO7HXQWUnAiYcqBlgpa11C/gAANw
SHe1DXg7mZjEK2ENmrPpqGcb0e4=
-----END CERTIFICATE REQUEST-----
```

Figure 16.2 A completed Certificate Request

Using the VeriSign Certificate Authority

VeriSign calls their certificate a *Digital ID*. To be sure that your *Digital ID Request* is processed quickly, and that you are following the current procedures, we suggest that you check the instructions at www.verisign.com. The following is a description of the procedure that is correct at the time this manual was written.

To obtain a certificate from Verisign:

1. Follow the steps to create a certificate request using the process described in the previous section.
2. Log onto www.verisign.com.
3. Click on *Products* in the main *Menu*.
4. From the *Products* page, select *SSL Certificates* in *Retail Certificates*.
5. In the *Secure Site Services* option, click the *Buy* option.

For basic 40 to 128 bit SSL Encryption, you will select the *Secure Site* option and then follow the instructions for *Secure Site* configuration.

For 40- to 128-bit SSL Encryption and 40- to 128-bit Server Gated Cryptography (allowing the server to send 128 encrypted data to 40 bit browsers), you will select the *Secure Site Pro* option and then follow the *Secure Site Pro* configuration.

Secure Site Configuration

1. Click the *Buy* option in *Secure Site Services*.
2. Follow the Verisign instructions through the next couple of pages to fill in company info, and so forth.
3. When you get to the Submit CSR page, enter WebSTAR for the Server Software Vender.
4. For the Enter CSR Information field, copy and paste the data created in the *Output for Authority* field that was generated when you used the *Generate Keys and CSR Requests* dialog box.

The data will start with `--BEGIN LINE--` and will finish with `-- END CERTIFICATE--` and you will need to paste all data including those two lines.

5. Follow the rest of the instructions to complete the Verisign certificate purchase. After Verisign authenticates and creates your certificate, they will send you an email with your certificate.

Copy the Begin Certificate and End Certificate data onto a new text document called Certificate, and place it in the *ServerCerts* folder. Use a text editor such

as *BEdit* to save the document as text only, not one that saves text as RTF or a binary format.

6. Follow the instructions for *SSL Configurations* to allow *Kerio WebSTAR* to access these files.

Secure Site Pro Configuration

1. Select the *Buy* option for *Secure Site Pro*.
2. Follow the *Verisign* instructions through the next couple of pages to fill in company info etc.
3. When you get to the Submit CSR page, enter Apache for the Server Software Vender.
4. For the *Enter CSR Information* field, copy and paste the data created in the *Output for Authority* field that was generated when you used the *Generate Certificate Request* dialog box.

The data will start with `---BEGIN LINE---` and will finish with `--- END CERTIFICATE---` and you will need to paste all data including those two lines.

5. Follow the rest of the instructions to complete the Verisign certificate purchase. After *Verisign* authenticates and creates your certificate, they will send you an email with your certificate.

You will then need to go back to *Verisign's* site to get a root certificate.

1. Go to www.verisign.com and click the *Support* option in the main *Menu*.
2. From the *Support* page, click the *Product Installation / Setup Instructions* option.
3. Select the *Secure Site Services* option under *Installation and Setup Instructions*.
4. When you know you have completed the first two steps, select the *Certificate Installation Instruction* option in step 3.
5. On this page you will want to select the *Get Intermediate CA Here* option.
6. Copy and paste the `---BEGIN LINE---` and all the data up to and including the `---END CERTIFICATE---` into a new document called *Root*.
7. Create a new document called *Certificate*.
8. Copy the Begin Certificate and End Certificate data from the certificate sent from *Verisign* and paste that data as the first part in this new certificate.
9. Then press the *Enter* key to add a space, and then finally copy the Begin Certificate and End Certificate data from the Root document and paste it in the last part of this certificate.
10. Place this Certificate document in the *ServerCerts* folder.

Follow the instructions for installing SSL Certificates to allow *Kerio WebSTAR* to access these files. This information is in the section [16.6](#).

16.5 Using the Thawte Certificate Authority

Thawte calls their certificate a server certificate (much like the *VeriSign Digital ID*). *Thawte* also provides wildcard certificates and test certificates.

As their instructions may change, you may want to check their website to be sure that your certification request is processed quickly: <http://www.thawte.com/>

The process includes these steps:

- Generate a *Private Key* file.
- Create a certificate signing request.
- Fill out *Thawte's* online certification request form.
- Send *Thawte* your documentation and payment.

You can expect to receive an email message from *Thawte* containing a URL where you can pick up your server certificate approximately three working days after *Thawte* has received all the appropriate paper and email documents.

Thawte Compatibility

If you or any of your customers are using a browser created before August, 1996, you may see security errors. Go to

<https://www.thawte.com/servertest.crt>

and follow the instructions to install the *Thawte* root certificate.

What Paperwork Does Thawte Need?

Thawte requires that you submit a signed letter on your organization's letterhead identifying the webmaster for your site, accompanied by Proof of Your Right to Use the Certified Organization Name and proof of the Distinguished Name of Your Web Server.

To avoid delays, submit all paper documents together.

Proof of Your Right to Use the Certified Organization Name

Proof of your right to use the certified organization name means proper documentation of your organization's identity.

Materials provided as Proof of Right to Use may be in your local language, if *Thawte* has an international representative in your country. Contact *Thawte* for more information.

Choose one of the following types of documentation as appropriate for your type of organization:

Companies, corporations, partnerships or proprietorships

Thawte needs a copy of your company registration documents. Thawte will also accept a copy of your articles of incorporation, partnership declaration or income tax registration if it is stamped by the relevant authority.

DBAs and Individuals

If you have a business name (DBA means Doing Business As), Thawte needs a copy of your DBA registration papers for local levies and taxes. Any official correspondence indicating your right to use the name given on your request will generally be sufficient.

Government Departments

Thawte requires an original signed letter from the Head of that department on the appropriate letterhead, along with contact information for his or her office and immediate superiors.

NGOs: Non-Government Organizations

Thawte requires an original signed letter from the Chief Executive, Chairman or Managing Director of the NGO, on the appropriate letterhead.

Universities and University Departments

Thawte requires an original signed letter from the Dean or Vice-Chancellor responsible for that department. Note that this must be on the appropriate letterhead, accompanied by contact information for the University concerned.

Special Interest Groups

Special-interest groups are popping up all over the Internet: The Apache Group, Debian, Linux Groups and others. Thawte will issue such groups certificates, but requests that you contact them to discuss verification and authentication.

Distinguished Name of Your Web Server

The Distinguished Name of a web server certificate is, by convention, set to the domain name of the server upon which the certificate will be used. Your organization will have gone through a process of registration in order to register this domain name. Because Thawte will be issuing a certificate tied to a domain name they need to be sure that you have the right to use a domain.

The procedure that you followed depends upon your location. If your domain ends in .com or .se, you do not need to do anything.

For all other domains, you must include a letter with the following domain name registration information:

- Which authority you registered through (InterNIC, etc.).
- The name and contact information you gave as an administrative contact.
- The name and contact information you gave as a technical contact.

Thawte uses this information to verify your right to use the domain name they are certifying. All you need to send *Thawte* is a printout of the domain registration application you made, or the WHOIS output for your domain.

Sending Information

Compile the required documentation and print your letter onto letterhead. Then attach payment and fax the entire package to the *Thawte* representative in your area. See *Thawte's* home page at: <http://www.thawte.com/> for information about how to do this.

Payment Methods

You can use one of these payment methods:

- Company check (in local currency)
- Credit Card (Visa, MasterCard, Diners Club, American Express)
- Bank draft
- International (SWIFT) Inter-bank Transfer
- Stronghold bundle
- Enterprise Account

Sending the request to Thawte

To apply for a server certificate, follow these steps:

- Use a text editor, such as *BEdit*, to open the Certificate Request document.
- Copy the entire contents of this document to the clipboard. Do not alter the contents.
- Use a web browser to navigate to *Thawte's Server Certification Application* form at: <https://www.thawte.com/cgi/server/step1.exe>
- Choose *Web Server* and continue.
- Paste the Certificate Request text into the Certificate Signing Request (CSR) window as indicated.
- On the pop-up menu for *Web Server Software*, choose *WebSTAR/SSL 4.0* and later.
- Indicate your payment method with the appropriate radio button.
- Continue with the forms pages by clicking the *Next* button.
- On page 2 of the *Server Certification Application*, examine the information obtained from your CSR to make sure it's correct.
- Use the company description pop-up menu to select an entry that most accurately describes your organization, or use the text box to enter a description of your own choosing.
- Fill out the authorizing contact information as specified.
- Fill out the web server administrator information as specified. The email address you provide here is the email address where your notification is sent.
- Make sure you're able to comply with the Payment Procedure.

- Continue with the forms pages by clicking the *Next* button.
 - Choose the currency type from the pop-up menu.
 - Provide the street address to which any correspondence to you must be sent. Provide an office fax number, if desired.
 - Choose the *Thawte* office closest to you, where you want to send your paper documentation and payment.
 - Click the *Submit Application* button.
The next page of the process contains a summary of your *Server Certificate Application*.
 - Print and save the *Server Certificate Application Complete* page.
- Don't forget to send your paper documentation and payment to the address provided on the summary page.

Downloading Your Certificate

Shortly after clicking the *Submit Application* button, you'll receive an email message from *Thawte* acknowledging receipt of your application. Contained in the message is a URL specific to your application which, when browsed, displays a *Server Cert Request Status* page. You can use your web browser to check the status of your application on this web page.

When *Thawte* has verified your documentation, received payment and processed the other information it requires, you are sent another confirming email message and the *Overall Status* item on the *Server Cert Request Status* page will read, ISSUED. At that time you can download and install the certificate.

Follow these steps:

- Use your web browser to connect to the URL in the email message.
- Make sure the radio button for *Standard Format* is checked.
- Scroll down so you can see and click the *Fetch Certificate* button to download your certificate.

The getcert page is displayed.

On this page you will see your returned certificate in an encoded form in a new web browser window. The text has a Begin and End statement like --BEGIN CERTIFICATE-- through --END CERTIFICATE-- (other certificates may say NETSCAPE CERTIFICATE CHAIN, or PKCS7 CERTIFICATE CHAIN).

- Save this as a text file in the *WebSTAR* folder, or the host folder if you have a secondary IP host.

Obtaining a Test Certificate

To obtain a test SSL certificate, you must first create a Certificate Signing Request, or CSR (see section 16.4).

When finished, open the CSR Request in a text editor such as *BBedit* and copy the contents to the clipboard.

Once you have created your CSR Request you next need to submit the CSR Request to *Thawte* or *VeriSign*.

Obtaining A Test Certificate From Thawte

1. In your browser, connect to
<https://www.thawte.com/cgi/server/test.exe>
2. If you have not already done so, download the test root certificate at <https://www.thawte.com/servertest.crt>, which is also linked to from this page.

Failure to download this root certificate may prevent your browser from accepting connections from any test certificate you create here.

3. Paste the CSR Request into the Certificate Signing Request text field.
4. Under Type of Certificate, select either Test SSL Cert or Test SGC 128-bit SSL Cert.
5. For the Certificate Format, choose either the *standard format*, the *Netscape cert chain format*, or the *PKCS#7 chain format*.
6. From the *Custom Certificate Options*, you can optionally select *Generate a X.509v3 certificate* and/or *Use an intermediate keysigning cert*.

Click the *Generate Test Certificate* button and save the resulting text to disk. Use a file name such as Certificate.

Warning: Do NOT save the file as *Digital ID*.

7. Move the newly created certificate file into the *WebSTAR* folder, storing it within the root of the appropriate host folder.

Obtaining A Test Certificate From VeriSign

1. In your browser, connect to
<https://www.verisign.com/server/trial/index.html>
2. Complete the *Free Trial Secure ID* form, entering your *First Name*, *Last Name*, *Company*, *E-Mail Address*, *Phone*, *Web Server* and other various information. When asked to select *Web Server* select *Other* if *Kerio WebSTAR 5* is not listed. Click *Continue*.
3. Read the *Before You Start* information and click *Continue*.
4. When prompted to generate a CSR Request click *Continue* if you have already completed this step as described above.
5. Paste the CSR Request into the *Certificate Signing Request* text field.
6. Enter additional *Technical Contact Information*, read the license agreement and click *Accept*.
7. Read the *Install Test CA Root* information and click *Accept*.
8. To complete the process *VeriSign* will email you a test certificate that looks something like this:

```
-----BEGIN CERTIFICATE-----
MIICNzCCAeECEE86maRLnPU0r2uYw/ogNEwDQYJKoZIhvcNAQEEBQAwdgkx
FjAUBgNVBAoTODVZcm1TaWduLCBjb2R0b3R552ZXJpc2lnbi
5jb20vcnVwb3NpdG9yeS9UZXNOQ1BTIE1uY29ycC4gQnkgUmVmLiBMaWFiLi
BMVEQuMUYwRAYDVQQLZ1Gib3IgVmVyaVNpZ24gYXV0aG9yaXplZCB0ZXNOaW5
nIG9ubHkuIE5vIGFzc3VyYW5jZXMgKEMpV1MxOTk3MB4XDTE1MDYxMDAwMDAw
MFoXDTE1MDYxNDIzNTk1OVowdDELMAkGA1UEBhMCVVMxEzARBgNVBAGTCkNh
bG1mb3JuaWExETAPBgNVBACUcEJ1cm1bGV5MREwDwYDVQQKFhAhTGFyTm1uZT
EPMA0GA1UECmQYXN0ZHNmMRkwFwYDVQQDFBB3d3cuc3Rhcm5pbmUuY29tMHw
wDQYJKoZIhvcNAQEBBQADAwAwAAJhAMRpi5BAe+x3q94BIqI66TieIP0jBQVH
1/frgipp+/257tGnbk1JmiCRhh0Bkn4sbYOLq2ag8H476r0twCfjNyNtfnN1I
+DKcRZfkkNvON2F8pVR6EInBKvxZTteEB80zQIDAQABMAOGCSqGSIb3DQEBA
UAA0EAmji0zPz4kNm6fu/b+DWewI1RV1zsmXHQnAspxQA1A8Bmtr9oWmf1Drk
118nash5gTDAb0ukTNJr116sXQD475A==
-----END CERTIFICATE-----
```

9. Copy and paste the text into a new text file, saving it with a file name like *Certificate*. Use an application that saves in the text-only format, such as *BBedit*. Do not save the document in the RTF (rich text format) format, any other markup language, or in

any word processor's format. The latter options include styled text information. Although there is no styled text in the certificate, the format itself included provisions for saving styled text information.

Warning: Do NOT save the file as Digital ID.

10. Move the newly created certificate file into the *ServerCerts* folder in the *SSLTools* folder.

Certificate files in this folder will be recognized as such in the *SSL Configurations* panel in the *Admin Client* application.

16.6 Installing Your Server Certificate

When you have received your certificate from the certificate authority and installed the document in the *ServerCerts* folder in the *SSLTools* folder, you are ready to configure the server to encrypt and decrypt transmissions using the certificate.

1. In the *Admin Client* application, click the *SSL Configurations* item in the *Web Server* group.

The following screen appears:

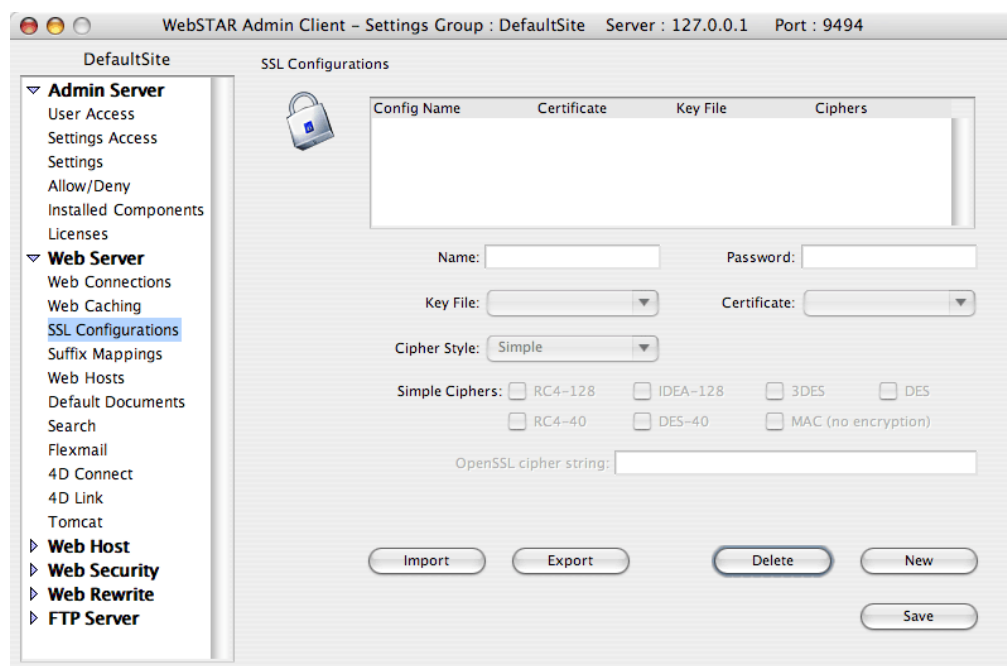


Figure 16.3 The SSL Configurations panel

2. Click *New* to enter a configuration that uses the key and certificate that you just generated.

A blank row in the *SSL Configurations* table appears.

3. Enter the name and password of the key in the *Name* and *Password* areas.

After you have generated your key (which is automatically stored in the *Keys* folder in the *SSLTools* folder) and saved a text copy of your certificate in the *ServerCerts* folder, the *Key* file and *Certificate* pop-up menus will be populated with these items.

4. Choose the key and the matching certificate from the pop-up menus.

The following illustration shows a completed SSL configuration.

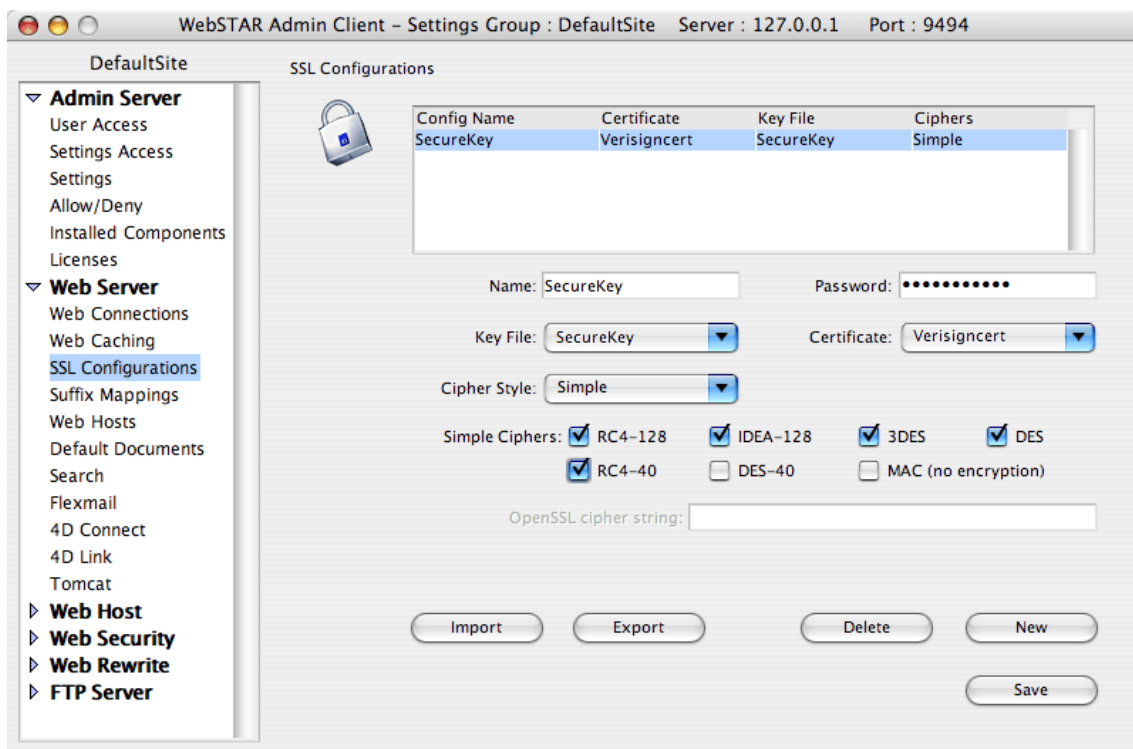


Figure 16.4 A completed SSL configuration

The lower area sets your security options, including certificate and private key data. The checkboxes set your policy regarding incoming connections.

Encryption Ciphers

The cipher checkboxes indicate which encryption algorithms you will support. When you purchased your certificate, you will be aware of the highest level of security that it supports. Both *Verisign* and *Thawte* charge more for the strongest encryption ciphers.

The client can connect only if they support at least one of the ciphers you enable, and they negotiate to find the best fit.

- Very high-security sites will just enable 3DES , IDEA-128, and RC4-128.
- Some U.S. government sites require DES only, so if you are in that situation, do not enable the RC4 options.

If you decide that your server does not require DES as the primary method, consider whether to allow your server to negotiate DES (which is more computationally intensive), or to allow only RC4.

Most sites that want to allow overseas users will need to turn on DES, DES-40 and RC4-40.

MAC is a little different, and should only be used if you need to allow users to connect to your SSL server in an unsecure mode. There are a few countries where authentication is allowed but encryption is not, and the MAC cipher is sometimes used by clients in these countries. The MAC cipher will send your certificate to the client and ensure the integrity of the data you send, but it won't encrypt the data.

When you have chosen your cipher settings, click *Save* to send the information to the server.

16.7 Using SSL in Kerio WebSTAR

In the *Web Connections* panel in the *Web Server* group, you should add a row that indicates the IP address and Port that will use SSL. By default, *Kerio WebSTAR* listens to all IP addresses on Ports 80 and 8000. Normally, you handle secure connections on Port 443. It may or may not be a different IP address as well.

To add an entry for secure connections:

1. Click the *Web Connections* item in the *Web Server* group.

The following screen appears.

This table shows that the web server is not yet configured to handle encrypted connections.

2. Click *New* to add a row to the *Web Listeners* table.

A blank row appears. The IP address field contains a pop-up menu that has an item for *All* as well as the IP addresses that are set up for that machine. If the machine handles one IP address, it is set in the TCP/IP panel in *Network Preferences*. To handle more than one IP address, you need to do using Unix commands.

3. Choose *All* or the desired IP address from the *IP Address* pop-up menu.

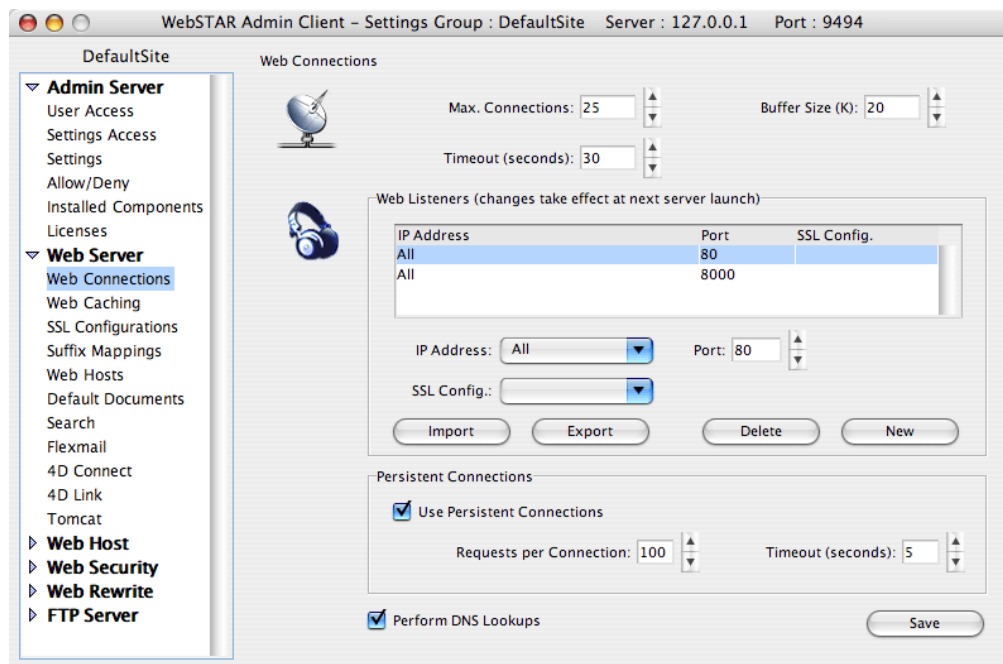


Figure 16.5 The Web Connections panel

4. Enter the Port that you want to use for encrypted transactions, usually 443.
5. Choose the SSL configuration from the *SSL Config* pop-up menu.

This pop-up menu lists the names of keys that you've created using the *Generate Keys and CSRs* dialog box.

6. Click *Save* to save your changes to the server.

A *Web Connections* page configured for encrypted communications on Port 443 is shown figure 16.6.

When you have installed the certificate, you should test your server to make sure it is encrypting communications properly. Then you can start serving and accepting information with assurance that it is secure.

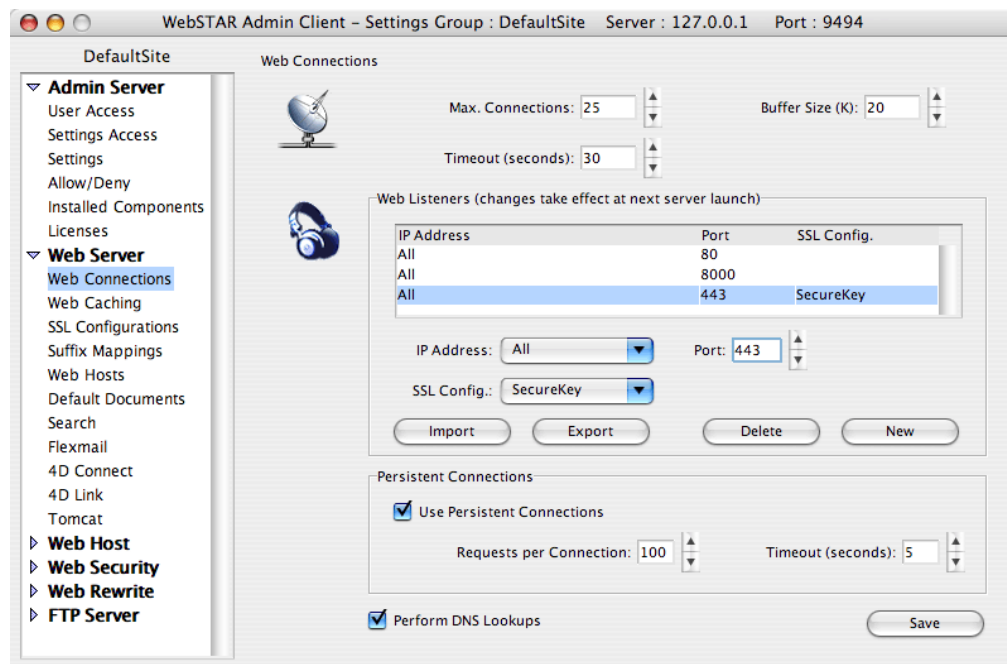


Figure 16.6 The Web Listeners table after adding a port for encrypted communications

16.8 Testing Your Secure Server

To connect to *WebSTAR SSL*, follow these steps:

1. Launch your SSL-compatible browser (such as *Netscape Navigator* or *Internet Explorer*).
2. Create a test file named `secure.html` in a folder named `secure` in a virtual host folder that you are serving.
3. Use HTTPS to open your page (using your host name), such as:

`https://www.domain.com/secure/secure.html`

The browser connects to the server and opens in secure mode. You can verify the security of a document by examining the security icon in the lower-left corner of the browser window.

In early versions of Netscape's browsers, the icon consists of a gold key on a blue background to show secure documents and a broken key on a gray background to show unsecure documents. In version 4 of Navigator and Communicator, it's a small padlock icon, open except for secure documents. Other browsers have similar icons to indicate security status.

Access Control

To preserve privacy, you should know who is accessing the information. Using *Kerio WebSTAR's* realm-based access control with user names and passwords (see chapter 6) will provide some of this accountability. Be sure that you use safe passwords and change them regularly, as well as using a log file analysis program to track all transactions.

You should also take extra security precautions with the machine serving private data.

Limit or remove FTP access

You should never allow FTP access to the folders with the private data, or their parent folders. FTP passwords are sent in the clear, so anyone monitoring your data stream can see them.

Install a server monitor application

A monitor, such as *LogDoor* from *Open Door Networks*, lets you track whether outsiders are attempting to crack the security on your server, and can monitor more carefully if you are under attack.

Consider a firewall

Macintosh servers are among the most secure, because they do not respond to remote commands or provide easy access to the data. However, if you store private data on your server, including credit card numbers, a firewall application or hardware will improve security. The firewall will allow only certain IP numbers to use specified protocols, which limits your vulnerability.

Additional Privacy Considerations

Serving private pages, such as medical records, personnel information, or financial data, is different from accepting credit cards. When you use SSL to accept a credit card, the browser encrypts the information before sending it to you. When you use SSL to display private information, the server encrypts it before sending it to the browser.

There are a number of ways to make sure you serve private pages via the secure HTTPS protocol:

- Use a security realm and set the *Minimum SSL* level to a value higher than *None*.
- Create a virtual host just for secure data. You can specify that all connections arriving on a given address/port combination are routed to this virtual host.
- Run another server on a separate machine.

SSL Server Performance Issues

Serving and accepting encrypted data via HTTPS takes more time than serving plain text pages and even graphics.

The most processor-intensive portion of the SSL protocol is the key exchange, when the browser first connects to your server using the HTTPS protocol. *Kerio WebSTAR* implements the resumable sessions, which allow browsers to re-use previous key exchange information. The second and subsequent connections by a browser doesn't need the same expensive session setup as the first connection, so they are much faster. In addition, the persistent connections feature allows the server to keep the connection to the browser open until all elements of the page are sent, which is much faster than opening a new connection.

If you are concerned about speed, the RC4 cipher is twice as fast as DES, and 6 times faster than 3DES, while providing the same amount of encryption.

16.9 Updating Your WebSTAR 4.x SSL Certificate

WebSTAR uses two files to enable the SSL protocol: the *Key* file and the *Certificate*, normally purchased from a certificate authority.

The *Public Private Key* file would have been created by using the *WebSTAR 4 CSR Utility*, and the *Certificate* is the file that was purchased from *Verisign*, *Thawte*, etc., unless it was a self-signed certificate.

More details on using/creating certificates for *WebSTAR 4.x* can be found starting on page 209 in the *WebSTAR 4.x* manual.

To use your *WebSTAR Key* file and *Certificate*, you need to migrate it to the *SSLTools* folder in the *KerioWebSTAR* folder.

1. Copy your valid *Public Private Key* from the *WebSTAR 4.x* folder into the *Keys* folder.
2. Copy the certificate from your *WebSTAR 4.x* folder into the *ServerCerts* folder.
3. Restart or launch the *Kerio WebSTAR Launcher*.

16.10 SSL Troubleshooting

The only noticeable difference between the SSL server and the regular version, is that the SSL version encrypts all data it sends and decrypts all the encrypted data it receives. In practical terms this means that an identical page will take two to four times longer to serve from the SSL version.

Questions specific to the SSL version or issues relating to encryption, digital IDs and certificates are covered in this section.

Kerio WebSTAR 5 quits immediately after launch.

- If the *Private Key* file has been regenerated, it will no longer match with your certificate.

Make sure that the *Private Key* file is the same one that was used to generate the original Request for Certificate — if necessary, use Get Info under the File menu in the Finder to compare the creation dates on these files to see that they are the same. The *Certificate* and the *Private Key* file make up a pair, and must match; if they do not, browsers will be unable to access your *WebSTAR SSL* server.

- The *Certificate* or *Private Key* files have been moved or renamed. Put them back and try again.

If you are still unable to launch *Kerio WebSTAR* after these steps, please contact *Kerio, Inc.* technical support for further assistance.

- URLs routed through third-party plug-ins don't work with HTTPS

Some third-party plug-ins may not support HTTPS and SSL connections. If you have difficulties, contact those developers directly.

- When users click on a hypertext link to a secure page the connection is refused.

Connections to a secure server require the client software to communicate via the SSL (HTTPS) protocol. Older browsers do not support the SSL protocol. The most recent versions of most browsers do support SSL.

- Users coming from networks which use a proxy gateway cannot connect to my secure pages.

SSL does not work through many proxy servers and firewalls. Consequently, you should provide both secure and unsecure versions if you want access to be available to everyone.

- *WebSTAR* launches as usual, but browsers are unable to connect and the log contains messages that report "SSL Handshake failed"

This could indicate that the certificate is corrupt. If you have a backup of the *Certificate* file, replace the one in the server's folder with the backup. Otherwise, you can regenerate the certificate from your *Certificate* file.

The "SSL Handshake failed" message could also indicate that the certificate does not match the public/private key.

- Browsers display an error message or security warning

- Certificate authority is expired
- Certificate issuer for this site is untrusted or unknown

When browser certificates expire, you will see these messages. For *Thawte*, that was July 28, 1998. For *VeriSign*, many certificates will expire at the end of 1999. You may have to help your customers or users change their browser certificates to recognize the new authorities.

To update browsers for *Thawte* server certificates, follow the instructions at:

<https://www.thawte.com/certs/server/rollover.html>

- Unable to establish a secure connection... (The identity certificate has expired)
- invalid site name
- An error occurred in the secure channel support

For all these errors, see the advice at

<https://www.thawte.com/support/server/browsers.html>

- When people click a link on the SSL page to go back to the non-SSL page, the new page is still secure.

If you are serving SSL and non-SSL pages from the same folder and use relative references in your HTML, then links from your SSL page will use the HTTPS protocol instead of HTTP. Make sure that links in the HTML of your SSL pages specify `http` when you intend a non-SSL link.

- When clients connect to a secure page they receive the message, *You have requested a secure document that contains unsecure information. The unsecure information will not be shown.*

Netscape does not allow both secure and insecure documents to be displayed on the same page. The problem is likely that the HTML portion is being served by *WebSTAR SSL* (HTTPS) while navigation buttons or graphics are being served out by *WebSTAR* (`http`). Make sure that all referenced documents are also linked with `https`.

Chapter 17

Kerio WebSTAR Directory Indexer

By default, *Kerio WebSTAR 5* serves the default home page when it receives a request that ends in a directory name rather than a document. The *Directory Indexer* plug-in gives you another option. Instead of displaying a home page, you can display a list of the files and folders in the directory.

To take advantage of this feature, you need to configure the directory using a small group of text files and grant appropriate permissions to the directory.

There is no user interface for the *Directory Indexer* in the *Admin Client* application; however, you can verify that the plug-in is loaded by checking the *Installed Components* panel in the *Admin Server* group (You must have full admin privileges to view this panel). If the plug-in is loaded, you will see an entry for *WebSTAR DirIndex*.

The `/DefaultSite/Examples` directory has an example folder set up to demonstrate the *DirectoryIndexer* plug-in. The *DirectoryIndexer* directory has no default page; instead, it is configured to list the files in the directory and provide uploading capabilities.

Visitors to your site can display or download the files listed in the directory index by clicking on them.

By default, folders within the directory are also listed, but you can suppress the listing of folders using a configuration parameter. This is covered in the following section.

The directory index listing will also include aliases to files in other folders. You can use this to allow people viewing your web site to open or download pictures, text, and HTML files in other folders, or any other file that you have access to from your server.

Before you move aliases into the directory, review your security strategy and think about who will have access to that file. It may be better to make a copy for the web site, so you know exactly what the public will see.

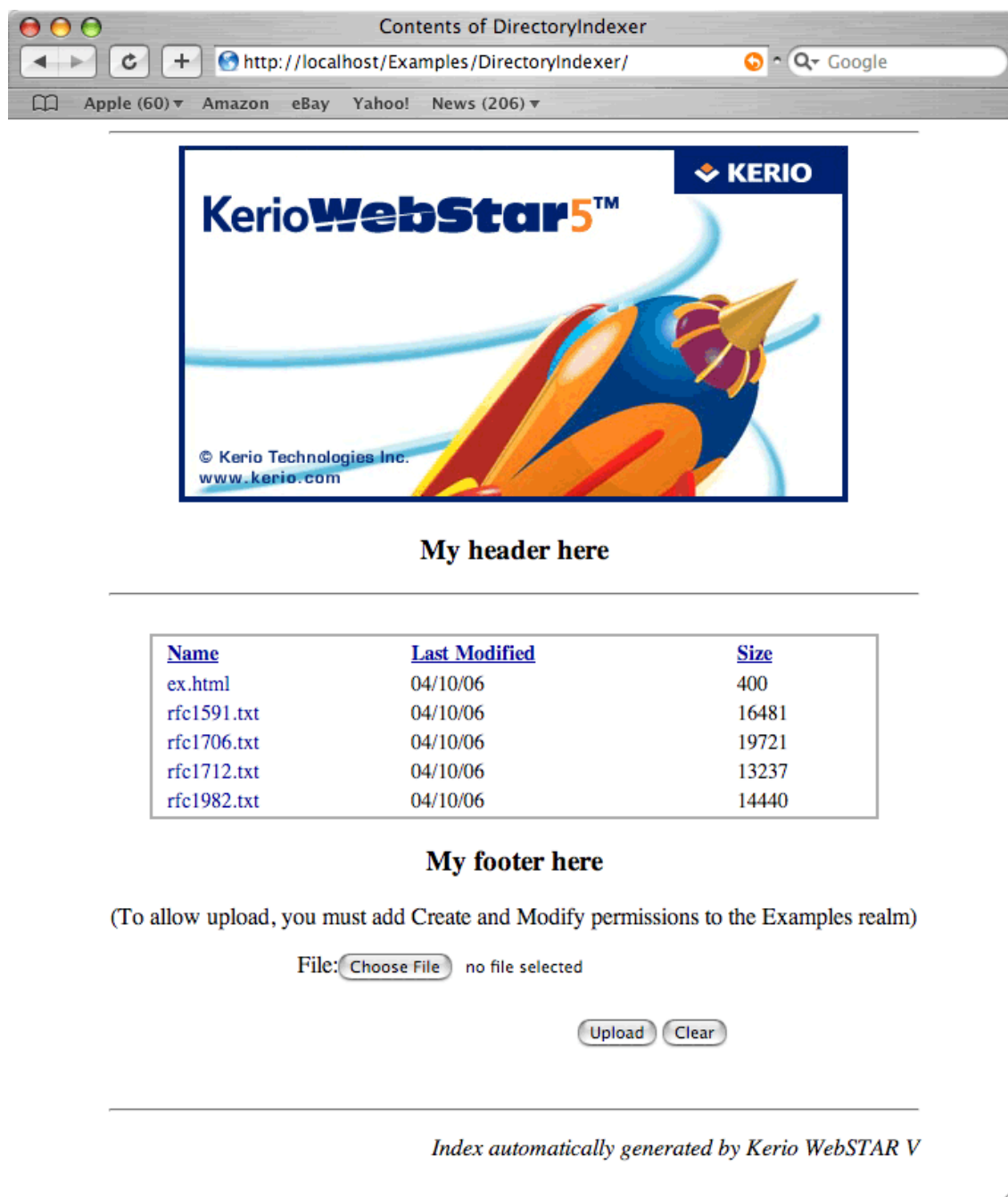


Figure 17.1 An example of the use of Directory Indexer

17.1 Using Directory Indexer

To tell *WebSTAR* to display the directory index listing rather than a default page for a folder, put a text file named `_message` into the directory, as described in this section. The `_message` file is an XML file that contains the settings for the *DirectoryIndexer* plug-in.

When a browser requests that directory in a URL (that is, the URL ends with the directory name and a slash), for example:

```
http://www.domain.com/widgets/
```

the directory index listing for the `widgets` folder will be displayed.

If you also place a default HTML file in a folder that also contains a `_message` file (such as a file named `index.html`), a browser can display the default HTML file instead of the directory listing by including its name in the request. For example:

```
http://www.domain.com/widgets/index.html
```

If a folder does not contain a `_message` file, then there is no way to get a directory listing.

There are two requirements for enabling the *Directory Index* feature.

- Create a realm that includes the directory and give that realm list privileges. If you want to support file uploading, you must also give the realm *Create* and *Modify* privileges.
- Create a `_message` file at the root level of this directory that contains the settings for the *DirectoryIndexer* plug-in.

Creating a Realm

You create a realm for the directory using the *Web Realms* panel in the *Web Security* group. You follow the same process as described in the section 6.2, but you are not required to set up an *Authenticator* or use an *Allow/Deny* table. That is, you don't need to protect the realm against unauthorized access. Of course, you can if you want to.

To create a realm with list privileges:

1. Connect to the *Admin Client* application and access the virtual host that contains the directory of interest and click the *Web Realms* panel.

Since *Web Security* is administered at the virtual host level, you must choose the correct virtual host when connecting to *Admin Client*.

2. Click *New* to create a new realm and give it a meaningful name.

It is easiest to define the realm using the name of the directory whose contents are to be indexed. In the example included with *Kerio WebSTAR*, the name of the directory is *DirectoryIndexer*.

3. In the match string area, enter the name of the directory and use the *contains* operator.
4. In the *Default Permissions* area, click the *Read* and *List* options and, if file uploading will be supported, choose the *Create* and *Modify* options as well.
5. (Optional) If you want to protect the realm against unauthorized access, use an *Authenticator* and/or an *Allow/Deny* table as described in the section 6.1.
6. Click *Save* to save your realm.

The following is an example realm that allows unrestricted list access to the *DirectoryIndexer* directory.

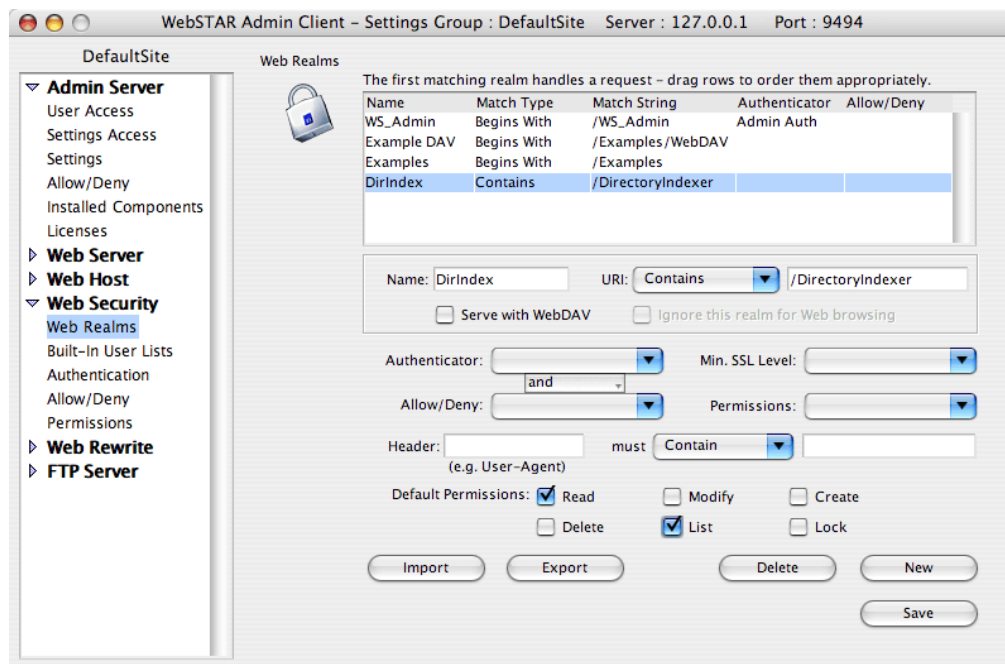


Figure 17.2 A realm with Read and List permissions

This realm supports browsing and downloading but not uploading.

Creating a _message File

You must create a _message file in each folder that will be display a listing. It is an XML file that contains the settings for the *DirectoryIndexer* plug-in. It begins with the line:

```
<DIRINDEX>
```

and ends with:

```
</DIRINDEX>
```

Here is a description of each field that can be set in the XML. If a field that has a default value is omitted, the default value is used automatically.

CSSFILE

You can optionally include a .css file in the directory whose files will be listed. If present, it will be used to format the directory listing. The *CSSFILE* parameter indicates the name of the .css file. The name of the .css file should begin with an underscore; otherwise it will appear as an item in the listing.

For example:

```
<CSSFILE>_WSDirIndex.css</CSSFILE>
```

If there is no CSSFILE statement, the default listing format will used.

The following example .css file is used in the *DirectoryIndexer* directory. It contains specifications for a table with a border drawn around the list, as shown in figure 17.1.

```
body { margin-left: 10%; margin-right: 10%; color: black;
background:white;}
:link { color: #000099; background-color:white;
background-image:none;} /* for unvisited links */
:visited { color: #990099; background-color:white;
background-image:none;} /* for visited links */
:active { color: #FF0066; background-color:white;
background-image:none;} /* when link is clicked */
:hover { color: #0060FF; background-color:white;
background-image:none;} /* when mouse is over link */
A.WSdir { font-family: Garamond, "Times New Roman",
serif; font-size:120%; }
A.WSfile { font-size: 100%; text-decoration: none; }
.WStable {Width: 90%;Border: 2px solid #AAA;}
.WStable TH {Font-Size: 85%; Vertical-Align: Middle;
Text-Align:Left;}
.WStable TD { Font-Size: 85%; line-height: 1.2em;
Text-Align: Left;Vertical-Align: Middle;}
```

```
.WStable .size {Width: 5%; Text-Align: Center;}  
.WStable .modified {Width: 20%; Text-Align: Center;}
```

The HTML generated by *Kerio WebSTAR* for the directory listing uses the classes `WSfile`, `WSmodified`, and `WSSize`, for the filename, size, and date modified fields. For example, a row in the listing in figure 17.1 was created using the following HTML:

```
<TD><A class="WSfile" href="rfc1591.txt">abc1591.txt</A></TD>  
<TD class="WSmodified">01/14/02</TD>  
<TD class="WSSize">16481</TD>
```

If you want to customize the appearance of the listing (font, font size, style, etc.), you can add specifications for the `WStable`, `WSfile`, `WSSize` and `WSmodified` classes. You can also change the background and other body information.

DATE

DATE sets the date format you want to use. You use symbols shown in table 17.1. For example:

```
<DATE FORMAT="%m/%d/%y" />
```

This format specifies a standard MM/DD/YY format.

Symbol	Description
%a	The abbreviated weekday name.
%A	The full weekday name.
%b	The abbreviated month name.
%B	The full month name.
%c	The appropriate date and time representation.
%C	The century number (the year divided by 100 and truncated to an integer)as a decimal number [00-99].
%d	The day of the month as a decimal number [01,31].
%D	Same as %m/%d/%y.
%e	The day of the month as a decimal number [1,31]; a single digit is preceded by a space.
%h	Same as %b.
%H	The hour (24-hour clock) as a decimal number [00,23].
%I	The hour (12-hour clock) as a decimal number [01,12].
%j	The day of the year as a decimal number [001,366].
%m	The month as a decimal number [01,12].
%M	The minute as a decimal number [00,59].
%n	A newline character.
%p	The equivalent of either a.m. or p.m.
%r	The time in a.m. and p.m. notation; this is equivalent to %I:%M:%S %p.
%R	The time in 24 hour notation (%H:%M).
%S	The second as a decimal number [00,61].
%t	A tab character.
%T	The time (%H:%M:%S).
%u	The weekday as a decimal number [1,7], with 1 representing Monday.

Table 17.1 Symbols used in setting the date format 1

Symbol	Description
%U	The week number of the year (Sunday as the first day of the week) as a decimal number [00,53].
%V	The week number of the year (Monday as the first day of the week) as a decimal number [01,53]. If the week containing 1 January has four or more days in the new year, then it is considered week 1. Otherwise, it is the last week of the previous year, and the next week is week 1.
%w	The weekday as a decimal number [0,6], with 0 representing Sunday.
%W	The week number of the year (Monday as the first day of the week) as a decimal number [00,53]. All days in a new year preceding the first Monday are considered to be in week 0.
%x	The appropriate date representation.
%X	The appropriate time representation.
%y	The year without the century as a decimal number [00,99].
%Y	The year with the century as a decimal number.
%Z	The time zone name or abbreviation, or by no bytes if no time zone information exists.
%%	%.

Table 17.2 Symbols used in setting the date format 2

FIELD

FIELD indicates which information is to be displayed for each file and in which order. The fields *Name*, *Size*, and *Last-Modified* are available. For example, the following uses all the fields:

```
<FIELD>Name;Size;LAST-MODIFIED</FIELD>
```

CHUNK

CHUNK specifies how many files and folders are to be listed per page. Set it to *Infinite* if you want all files and folders to be listed on the same page. The default value is 20. For example:

```
<CHUNK SIZE="15"/>
```

If there are more files and folders in the directory than the value of CHUNK, the *Directory Indexer* will display a Next Page-Previous Page navigation bar below the listing, as shown in figure 17.3.

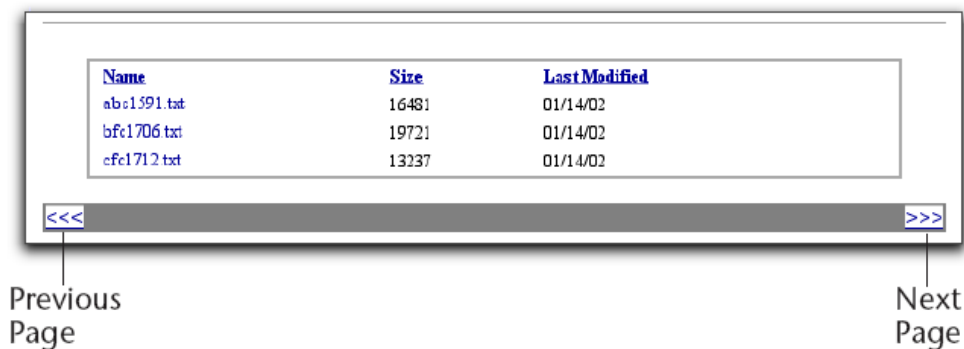


Figure 17.3 Navigation bar with Next Page and Previous Page buttons

The user would view files in pages using the buttons in the navigation bar.

HIDEDIR

The presence of HIDEDIR indicates that folders will not appear in the listing. By default, they are displayed. To suppress the listing of folders, enter

```
<HIDEDIR/>
```

in the `_message` file. To display folders, omit this parameter.

ORDER

ORDER sorts the listing. The *BY* field can be *Name*, *Size*, or *Last-Modified* and *WAY* can be either INCR (increasing order) or DECR (decreasing). For example:

```
<ORDER  
BY="LAST-MODIFIED"  
WAY="DECR"  
</ORDER>
```

Adding a Header

You can optionally add a header to the listing. You do this by adding an HTML or text file called `_header` to the directory. Place it at the same level as the `_message` file. If present, the contents of the `_header` file will be displayed above the listing.

In figure 17.1, the graphic and the text above the table were added by the header. In this example, the `_header` file simply consists of the lines:

```
<hr><center>
<IMG SRC="/images/Keriowsv.jpg" border="0"><br>
<h3>Kerio WebSTAR 5 Example Directory Indexer</h3>
</center><hr>
```

Adding a Footer

To display information below the directory index listing, create an HTML or text file called `_footer` and place it at the same level as the `_message` file. In the example (figure 17.1), the text below the listing and the file upload interface were added by the `_footer` file.

File Upload Interface

You can enable users to upload files directly from their browsers to the directory they are viewing. You do this by writing the appropriate HTML in either the `_header` or `_footer` files.

Such an HTML page will display a file name prompt, a field to type the name, a *Browse* button to choose a file, and buttons to Upload the file or Clear the field. In the example that ships with *Kerio WebSTAR*, this code has been added to the `_footer` file. Therefore, the upload options appear below the list.

The HTML for the `_footer` is as follows:

```
<left><H4>Example Directory Index</H4></left>
<CENTER><FORM ENCTYPE="multipart/form-data" ACTION="file.upload"
METHOD=POST>
<TABLE BORDER=0 CELLPADDING=10 CELLSPACING=5>
<TR><TD ALIGN=LEFT>File:<INPUT NAME="remotefile" TYPE="file"
SIZE=25>
<TR><TD ALIGN=RIGHT><INPUT TYPE="SUBMIT" VALUE="Upload"><INPUT
TYPE="RESET" VALUE=" Clear ">
</CENTER></FORM></TABLE>
```

In order to support uploading, the web realm must grant write permissions to browsers. If only list permissions are granted, *WebSTAR* will prohibit file uploads even if the HTML that implements the file upload option is correct.

If you want to support file uploading, use the *Web Realms* panel to grant *Create* and *Modify* permissions as well as *Read* and *List* permissions. This is shown in figure 17.4.

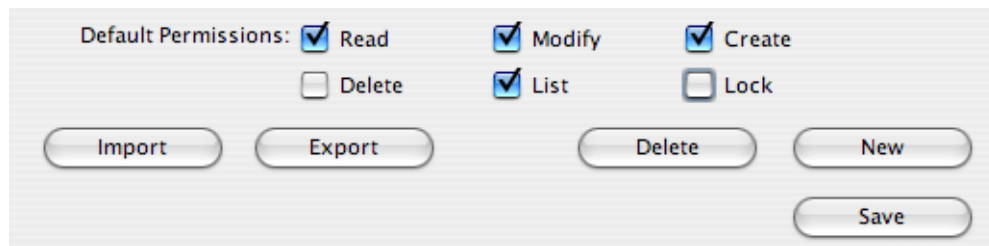


Figure 17.4 File permissions that allow file uploads

Only relatively new browsers (such as *Netscape* version 3 or greater or *Explorer* version 4 or greater), will be able to upload files. Some browsers will display the Upload button but only some can actually upload the files. You may decide to use the WebSTAR's WebDAV (see chapter 7) or WebSTAR FTP (see chapter 11) for file transfers.

17.2 Editing the Configuration Files

To improve performance, the web server caches the contents of the `_message`, `_header`, `_footer`, and `.css` files. If you make changes to any of these files while the web server is running, you must clear the web server's cache in order for your changes to take effect. You do this by clicking the *Flush Cache* button at the top of the *Web Caching* panel in the *Web Server* group in the *Admin Client* application.

The next time the listing is requested, your changes take effect.

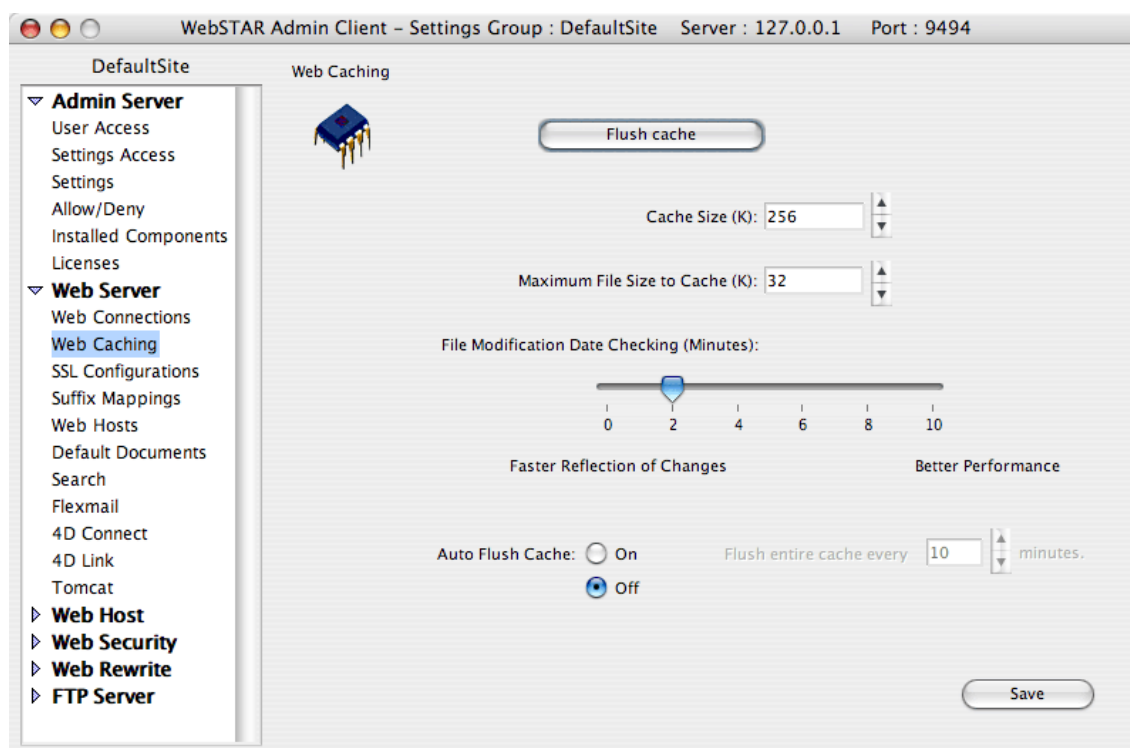


Figure 17.5 The Flush Cache button on the Web Caching panel

Chapter 18

Suffix Mapping

Kerio WebSTAR uses CGIs and plug-ins for file listing, database access, multimedia, SSIs (Server-Side-Includes), and other dynamic interactions. To launch most CGIs and plug-ins, the web server looks for special URLs as specified by those add-on programs. When one of these the URLs is received, the server examines the file name element and sends the request to the appropriate CGI or plug-in.

Entries in the *Suffix Mapping* table define what should happen, based on the filename suffix. This could be processing by a CGI or plug-in, or serving the file without any modification on the server side.

Kerio WebSTAR 5 ships with an extensive list of suffix mappings. For most users, you will not need to modify or add to items on this list.

Note: Each URL can only go through one Action, although plug-ins and CGIs can call others for additional processing.

18.1 File Suffixes

A file suffix is the element of the filename after the last period, as in these examples:

- `default.html` is a normal HTML file.
- `special.bluewidget.html` is also a normal HTML file. The data before the last period is ignored in defining the suffix.
- `bluewidget.gif` is a GIF graphic file.
- `red-widget.jpeg` is a JPEG graphic file.
- `letter.txt` is a plain-text file which will be displayed without special formatting.
- `morning.mpeg` is a video to be played by a browser plug-in.
- `guestbook.cgi` requests a guestbook CGI application to process the URL.

18.2 How Suffix Mapping Works

Here is an overview of how *Kerio WebSTAR* uses suffix mapping.

1. A browser sends a URL to Kerio WebSTAR. For example, a URL ends with:
`/products/shipping.shtml`
2. If there's a suffix at the end of the file part of the URL, *Kerio WebSTAR* looks up that suffix in the *Suffix Mapping* table. *Kerio WebSTAR* uses the *Action* in the first matching entry in the list.

In this example, `.shtml` maps to the *SSI Action*, by default.

3. If the suffix not entered in the *Suffix Mapping* table or there is no suffix, the web server will use the default MIME Type (usually TEXT/HTML) and the default action.

Once the web server locates the correct entry, it either serves the file directly or sends the request to the plug-in or CGI specified by the *Action* entry. For more information, see the section [18.6](#).

Note: Each URL can only go through one *Action*, although plug-ins and CGIs can call others for additional processing.

18.3 Examples of Suffix Mapping

Handling a GIF File

Suppose the server receives this URL:

`http://www.domain.com/logo.gif`

The web server looks up `gif` in the *Suffix Mapping* table. It finds only one entry with a suffix that matches, which has the action of default.

So the server locates the `logo.gif` file in the root folder, reads the data and returns it to the browser with the `image/gif` MIME type without any further processing. Most browsers will display images of this type directly in the browser window.

Example: Handling an Image Map

Suppose the server gets a URL such as:

`http://www.domain.com/logo.map`

It first looks up `.map` in the *Suffix Mapping* table, where it finds the first entry matching that suffix. Because the action is `MAPPY`, the web server passes the appropriate

data to the *WebSTAR Image Map* plug-in, which processes the request and sends back a result. Then the server returns the resulting data to the browser with the MIME type `application/ x-image-map`.

For more information about creating and managing image maps, see chapter 15.

18.4 Fields in the Suffix Mapping Table

The *Suffix Mapping* tables have three fields: *Suffix*, *MIME type*, and *Action*. *Kerio WebSTAR* uses the *Suffix* field to identify a file passed as a request, the *Action* field defines how the URL should be handled, and the *MIME Type* field specifies the type of information being returned to the browser.

Action Popup Menu

The *Action* field tells the *Kerio WebSTAR* server what to do with this kind of URL. The *Suffix Mapping* interfaces include the available *Actions* in the popup menu. Several of them are extremely useful, while others are for internal use only.

- *BINARY* — Return the file contents using the BINARY transfer method, without changing anything. Only the data fork of Macintosh files is sent (not the resource fork). An HTML header with the MIME type is always added to the data. To return a file with no header information, use the RAW action, described below.

Use this BINARY action for files unless they require special processing.

- *TEXT* — Return the file contents using the TEXT transfer method, which converts line ends to CR/LF for older browsers. This conversion will delay the response slightly.
- *CGI* — Load and execute the CGI application specified in the URL.

WebSTAR passes arguments to the application by using the *sdoc Apple Event*. The application is expected to generate a legal HTTP header and any response data and return it to the web server as a result of processing the *Apple Event*. This result will be transmitted to the browser without modification. For better performance, use the ACGI action.

- *ACGI* — Load and execute the ACGI application specified in the URL.

WebSTAR passes arguments to the application by using the *Search Doc Apple Event*. The application is expected to generate a legal HTTP header and any response data and return it to the web server as a result of processing the *Apple Event*.

Unlike the CGI action, ACGI applications can run asynchronously: the web server doesn't have to wait for the results before proceeding.

- *RAW* — Return the file contents without changing anything or generating an HTTP header for the file. The RAW file should have its header inside the file. Only the data fork of Macintosh files is sent: the resource fork is not sent.
- *SSI* — Processes SSI commands within HTML documents.
- *SEARCH* — The file is an index file, used by the *WebSTAR Search* plug-in for web-crawling and searching.
- *MAPPY* — Return URLs based on the point on the image map that the person clicked in the browser (see chapter 15).
- *4D Link* — Used by the *Kerio Link* plug in for serving dynamic web sites.
- *4D Connect* — Used by the *Kerio Connect* plug-in for serving dynamic web sites.
- *Flexmail* — The file contains instructions for how the *Flexmail* plug-in should handle the request. For information on the language used by *Flexmail*, see chapter 13.

MIME Type

The MIME Type appears in the form general/specific. These can be defined by the web community, like `text/html` or a vendor such as *Apple*, with `video/quicktime`. Your server can also send files to be opened by a specific application, when the MIME type is application and the program name. For example, you can specify `application/pdf` for *Adobe Acrobat* files or `application/x-pn-realaudio` for *RealAudio* files.

There's no definitive list of MIME types on the web, but a useful listing is available at:

<http://www.isi.edu/in-notes/iana/assignments/media-types/media-types>

Some third-party CGIs and plug-ins use the MIME types for their own internal purposes as well. Check their documentation for information.

In addition, publishers of software that requires special MIME types may have describe them in their documentation or web site, or the information may be available through Usenet newsgroups or email discussion lists on the software. If the software includes a browser plug-in, you can install it and then view the browser's application settings to discover the MIME type.

Kerio WebSTAR's default MIME Type is `text/html`.

18.5 Editing Suffix Mapping Table Entries

The *Suffix Mappings* panel in the *Admin Client* application displays the *WebSTAR* web server's current mapping, action, and MIME type entries.

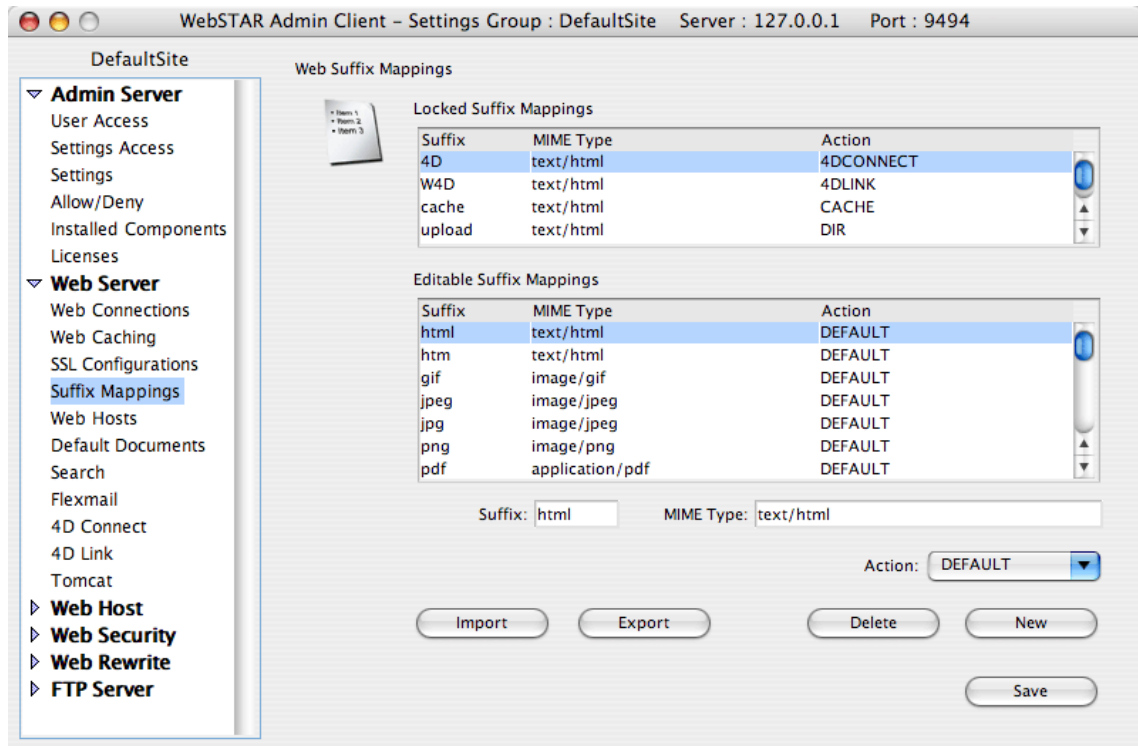


Figure 18.1 The Suffix Mappings panel

The top table, *Locked Suffix Mappings*, contains suffix mappings used by the plug-ins that ship with *Kerio WebSTAR 5*. These mappings cannot be edited.

The bottom table contains the suffix mappings that you can modify. *Kerio WebSTAR 5* ships with an extensive list of suffix mappings that cover all the file types commonly encountered on the web. Many users will never need to edit suffix mappings.

The default suffix mappings cover text, graphics, CGIs, image-map files, encoding and compression, common application files, multimedia, and Java. For descriptions, see the section 18.7.

You can edit these entries or add new entries as you add third-party plug-ins to *Kerio WebSTAR*. You can also change the order in which the suffix mappings appear in the tables. Remember, the server will use the action belonging to the first matching entry in the table that it finds.

Before doing extensive modifications, it is wise to export your suffix mappings to disk, in case you ever need to re-create them. Click the *Export* button to export the contents of the *Editable Suffix Mappings* table to disk as a text file.

Suffix Mappings are administered at the server level and, therefore apply globally to all virtual hosts. You must have full admin privileges to access and modify these settings.

Many third-party plug-ins automatically install *Suffix Mapping* entries in this table. Consult their documentation for details.

To add a new suffix mapping:

1. Click the *Suffix Mappings* item in the *Web Server* group in the *Admin Client* browser. The *Suffix Mappings* panel appears.

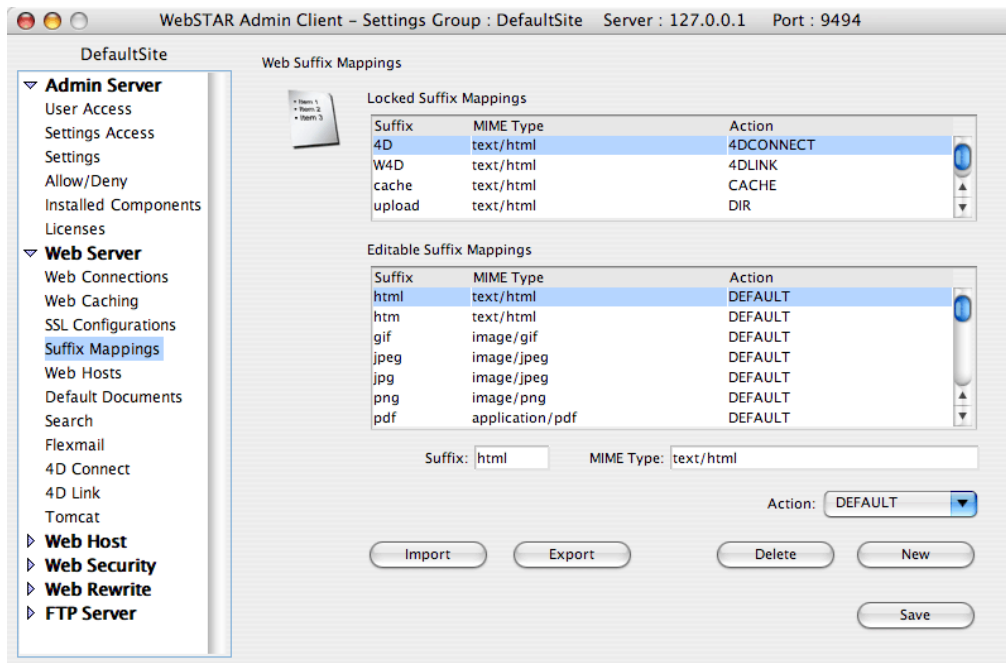


Figure 18.2 The Web Suffix Mappings panel

Each entry has a line in this table, and includes the most common suffixes. These defaults are listed in the section 18.7.

2. To add an entry to the *Editable Suffix Mappings* table, click *New*.
A blank row appears in the table. You may need to scroll down to see it.
3. Enter the suffix and MIME type, and choose an action from the *Action* pop-up menu.
4. Repeat this process, as desired.

To modify an existing mapping:

1. Click twice in the *Suffix* or *MIME type* columns in the *Editable Suffix Mappings* table to get an insertion point.
2. Edit the contents of these columns, as desired.
3. If necessary, choose another action from the *Action* pop-up menu.

To reorder the entries in either *Suffix Mapping* table:

1. Click on the row to be moved in either *Suffix Mapping* table.

The pointer changes to a hand pointer.

2. Drag the row up or down.

As you drag, a red line indicates the place in the table where the row will be dropped when you release the mouse button.

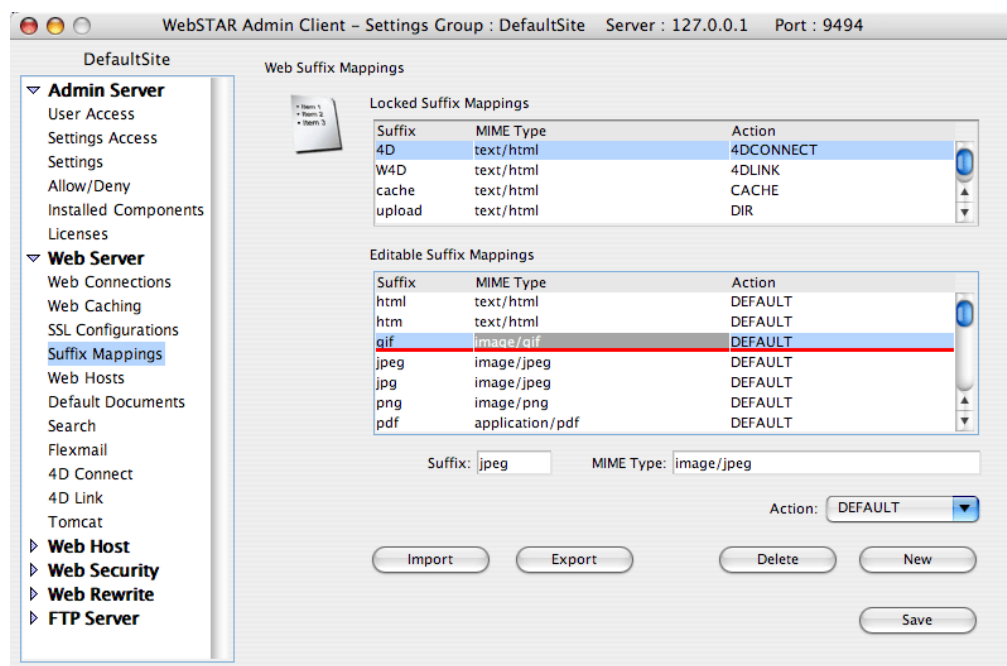


Figure 18.3 A row in the Suffix Mapping table being dragged

3. When the row reaches the desired location, release the mouse button.

18.6 Suffix Mapping Example: Using SSI for All HTML Files

Perhaps you'd like to include the modification date or another SSI feature on your pages, but you don't want to change the suffix from `.html`. You can use the *Suffix Mapping* Table to instruct the web server to use the *WebSTAR SSI* plug-in for all `.html` files.

To make this change, you must have full admin privileges, not host admin privileges:

To use SSI for html files:

1. In the *Admin Client* application, click the *Suffix Mappings* item in the *Web Server* group.
2. Select the entry for `.html` in the *Editable Suffix Mappings* table.
3. Select the *Action* popup menu and choose SSI:

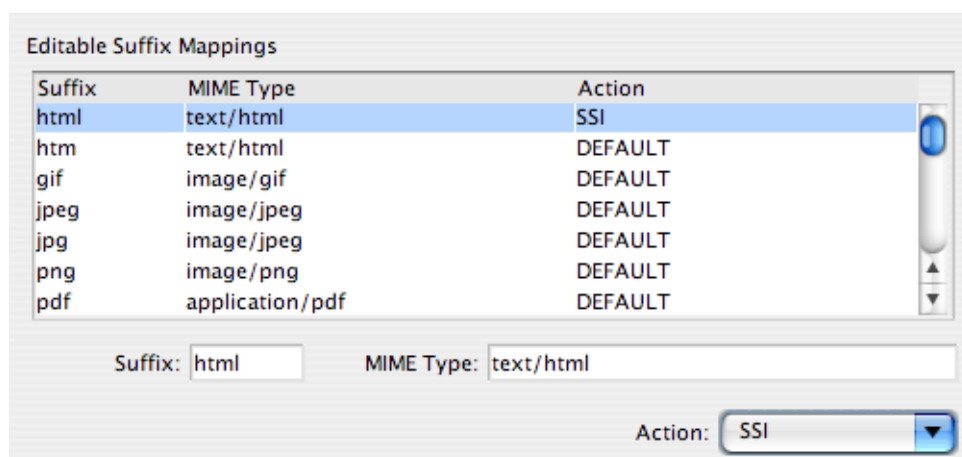


Figure 18.4 Modifying the Action for `.html` files

4. Click *Save* to save the change on your server.

From now on, all HTML files will be sent through SSI before they are returned to the browser. *WebSTAR SSI* will replace the SSI commands in the file with the appropriate data (such as the date the file was changed), and the browser will display these as standard text.

18.7 Suffix Mapping Defaults

Here is the list of locked (uneditable) suffix mappings.

Suffix	MIME type	Action	Description
4D	text/html	4DConnect	Plug-in for dynamic web sites via 4th Dimension.
W4D	text/html	4DLink	Plug-in for dynamic web sites via 4th Dimension.
cache	text/html	Cache	WebSTAR data cache (files you wish to cache).
upload	text/html	DIR	WebSTAR file upload (linked to .upload files).
flx	text/html	Flexmail	WebSTAR Flexmail.
map	application/ximage-map	MAPPY	Image map plug-in.
raw	text/html	RAW	RAW files, served as binary.
search	text/html	SEARCH	WebSTAR Search, used for indexing sites.
ssi	text/html	SSI	WebSTAR SSI, used for processing by the SSI plug-in.
shtml	text/html	SSI	WebSTAR SSL, used for secure transactions.

Table 18.1 Locked suffix mappings

Here is the default suffix mappings table.

Suffix	MIME Type	Action	Description
html	text/html	DEFAULT	HTML marked up text.
htm	text/html	DEFAULT	HTML marked up text.
gif	image/gif	DEFAULT	GIF image files.
jpeg	image/jpeg	DEFAULT	JPEG image files.
jpg	image/jpeg	DEFAULT	JPEG image files.
pdf	application/pdf	DEFAULT	Adobe portable document format files.
txt	text/plain	DEFAULT	Plain (unmarked) text.
sit	application/x-stuffit	DEFAULT	Stuffit (compressed) files.
js	application/x-javascript	DEFAULT	Javascript files.
hqx	application/mac-binhex40	DEFAULT	Binhex (compressed) files.
css	text/css	DEFAULT	Style sheet files.
rtf	text/rtf	DEFAULT	Rich Text Format files.
xsl	text/xml	DEFAULT	XML file
xml	text/xml	DEFAULT	XML file
dtd	text/xml	DEFAULT	Document Type Definition file.
sgml	text/sgml	DEFAULT	SGML file.
mpeg	video/mpeg	DEFAULT	MPEG video file.
word	application/msword	DEFAULT	MS Word file.
doc	application/msword	DEFAULT	MS Word file.
xl	application/vnd.ms-excel	DEFAULT	MS Excel file.
xls	application/vnd.ms-excel	DEFAULT	MS Excel file.
ppt	application/vnd.ms-powerpoint	DEFAULT	MS Powerpoint file.
tar	application/x-tar	DEFAULT	.tar (Unix) file
gz	application/x-gzip	DEFAULT	.gz (Unix) file.
zip	application/zip	DEFAULT	.zip (compressed) file.
pict	image/pict	DEFAULT	PICT file.
wsdl	text/xml	DEFAULT	text/xml file.
mov	image/quicktime	DEFAULT	Quicktime file.
qti	image/x-quicktime	DEFAULT	Quicktime file.
qtif	image/x-quicktime	DEFAULT	Quicktime file.
pntg	image/x-macpaint	DEFAULT	MacPaint file.
qtll	application/x-quicktimeplayer	DEFAULT	QuickTime Player app.

Table 18.2 Built-in editable suffix mappings

Chapter 19

Automatic Localization

If your web site serves multiple languages, you can use the *WebSTAR Language* plug-in to automatically route requests for your home page to the correct (i.e., localized) default *.html* file. The *Language* plug-in looks for the language token that the browser sends and automatically selects the matching version of the requested *.html* document.

To use the *Language* plug-in, you need to:

- Verify that it is installed,
- Create and install localized versions of your default page at the root level of your *Virtual Host* folder.
- Enable automatic localization using the *Admin Client* application.

19.1 Installation

The *Kerio WebSTAR Language* plug-in is installed as part of the *Easy Install* option. You can verify that is installed by viewing the *Installed Components* panel in the *Admin Server* group.

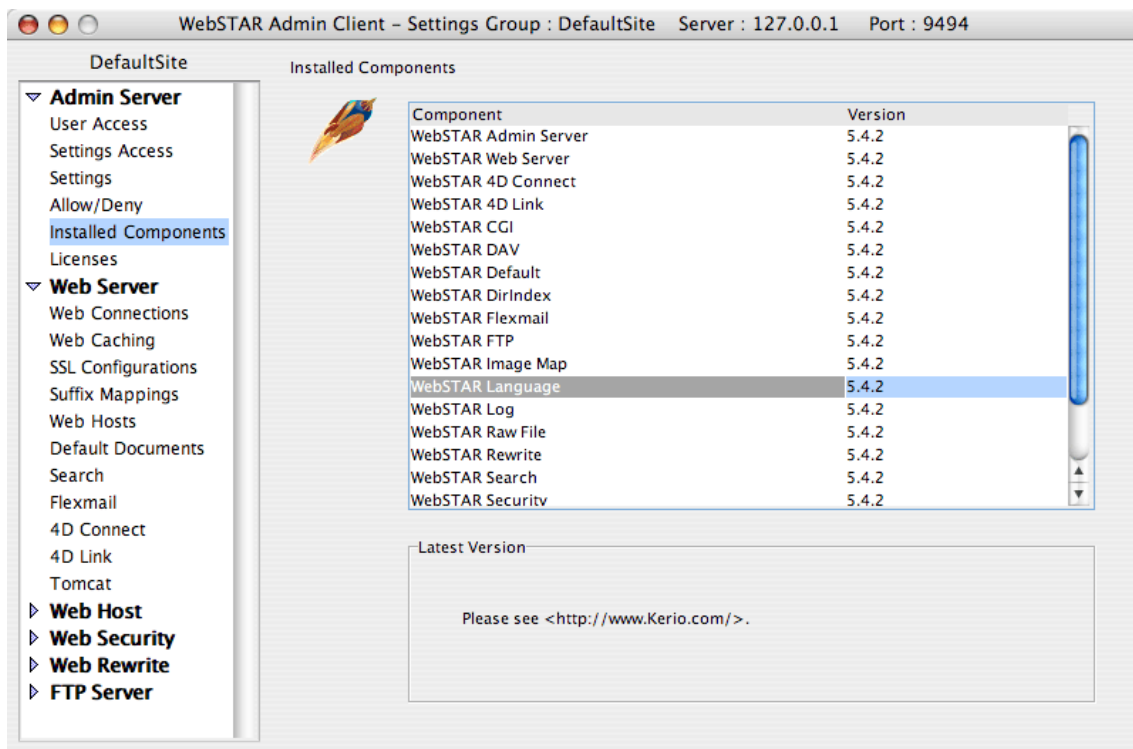


Figure 19.1 The Language plug-in installed

19.2 Enabling and Configuring Localization

The *Language* plug-in adds the *Localization* panel to the *Web Host* group in the *Admin Client* application. It uses a fairly straightforward interface, which consists of a checkbox and a *Save* button. To enable automatic language negotiation, click the checkbox and the *Save* button:

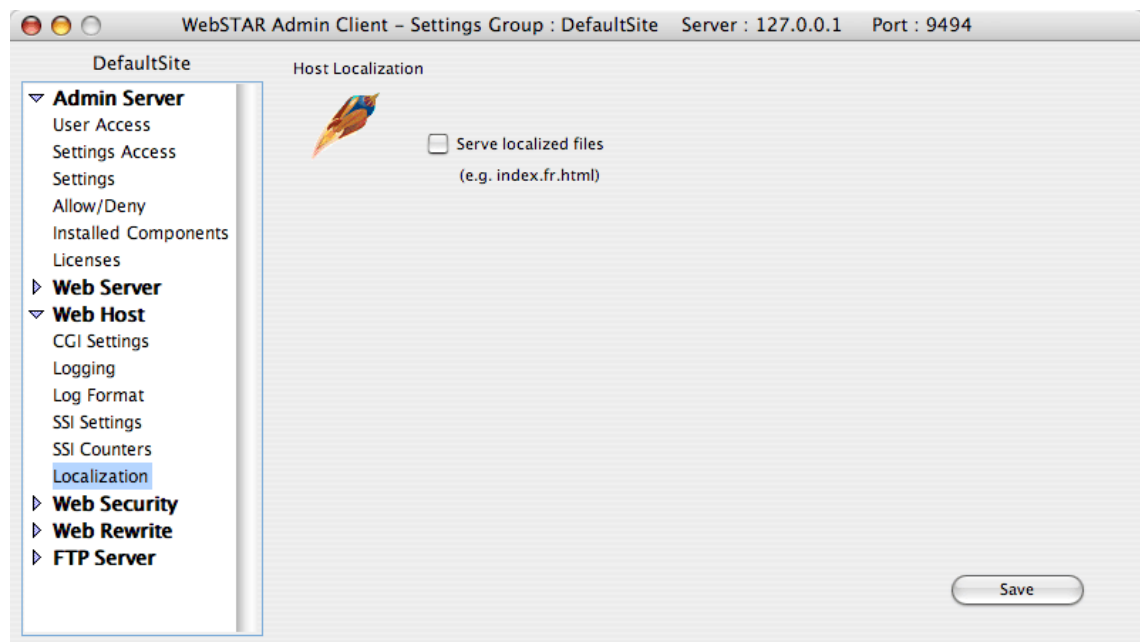


Figure 19.2 Enabling automatic localization

Installing localized documents

The last step is to create, name, and install localized documents for each language that you support. Place your localized `default.html` files at the root level of the *Virtual Host* folder. You may want to create separate subfolders for each language that you support. For example, if your default page is named `index.html` and you serve English (US and UK), French, and German versions of your site, you create the following `.html` files:

- `index.html`
- `index.fr.html`
- `index.de.html`
- `index.en-us.html`
- `index.en-gb.html`

The *Language* plug-in will automatically choose the first one on the list of language tokens your browser sends (it's a priority list). If no match is found, it will serve the version of the default page with no language suffix.

One scenario is that each localized default page contains hardcoded links to the set of documents specific to that language. In this manner, the *Language* plug-in is involved only choosing the default document.

However, you can also let the *Language* plug-in automatically choose the correct localized document for any request. The *Language* plug-in will function on any filename in the directory, including filenames such as `default.en-gb.php` (you may want to use suffix-mapped actions on localized files) and filenames with only one suffix, such as `de.html` or `en-us.html`, though you must provide the exact language substring you wish to match in naming each of your documents.

If you are using a default language in your non-localized document (as in `index.html` above) it is a good idea to duplicate that document with the appropriate substring. As an example, if your default language is English, then you'll need to have `index.html` and a duplicate of the file named `index.en.html`.

Please note that if you add a localized file while *WebSTAR* is running, the change won't take effect until *WebSTAR*'s cache is flushed. You can flush *WebSTAR*'s cache manually by clicking the *Flush Cache* button on the *Web Cache* panel in the *Web Server* group. The new file will also become available when your designated cache flushing period elapses and *WebSTAR* flushes the cache itself. For more information on the cache, see the section [10.1](#).

Chapter 20

Using 4D Link

4D Link is a *Kerio WebSTAR* plug-in that allows you to use *Kerio WebSTAR 5* as the web server for a *4th Dimension* database.

With *4D Link*, you can use *4th Dimension* to serve dynamic pages and *Kerio WebSTAR* to serve the static pages in your web site. When requests that contain a *Kerio Link* directive are received from a browser, *4D Link* passes the request on to the 4D web server. 4D then returns the requested information to *Kerio WebSTAR*, which is served to the browser.

4D Link allows *Kerio WebSTAR* to use 4D like a CGI. Although *Kerio Link* is not really a CGI in the strict meaning of the term, the behavior of *4D Link* is similar to a CGI.

Kerio Connect (chapter 21) provides essentially the same functionality, but uses a template-based approach.

20.1 4D Link Requirements

Dynamic web sites using *4D Link* can be set up on either one computer or different computers for the web server and the database (or databases). For optimum performance, install *4th Dimension* and *Kerio WebSTAR* on separate computers. If one computer is used, *4th Dimension* and *Kerio WebSTAR* must use different TCP ports.

In *4th Dimension*, the TCP port is set on the *Web Server Publishing* page of the *Database Properties* dialog box. The *Publish Database at Startup* preference should also be turned on unless you want to start web publishing manually at a later time. You can start web publishing from the *User EnvironmentWebPages* by choosing *Web Server → Start Web Server* or with the `START WEB SERVER` command.

The 4D application that provides web services must have *4D Web Extension* or *4D Server Web Extension* installed. *4D Link* requires *4th Dimension* version 6.7 or above.

The *4D Link* plug-in is installed by default and its settings are available in the *Web Server* group in the *Admin Client* application. This means that you must have full admin privileges to configure *4D Link*.

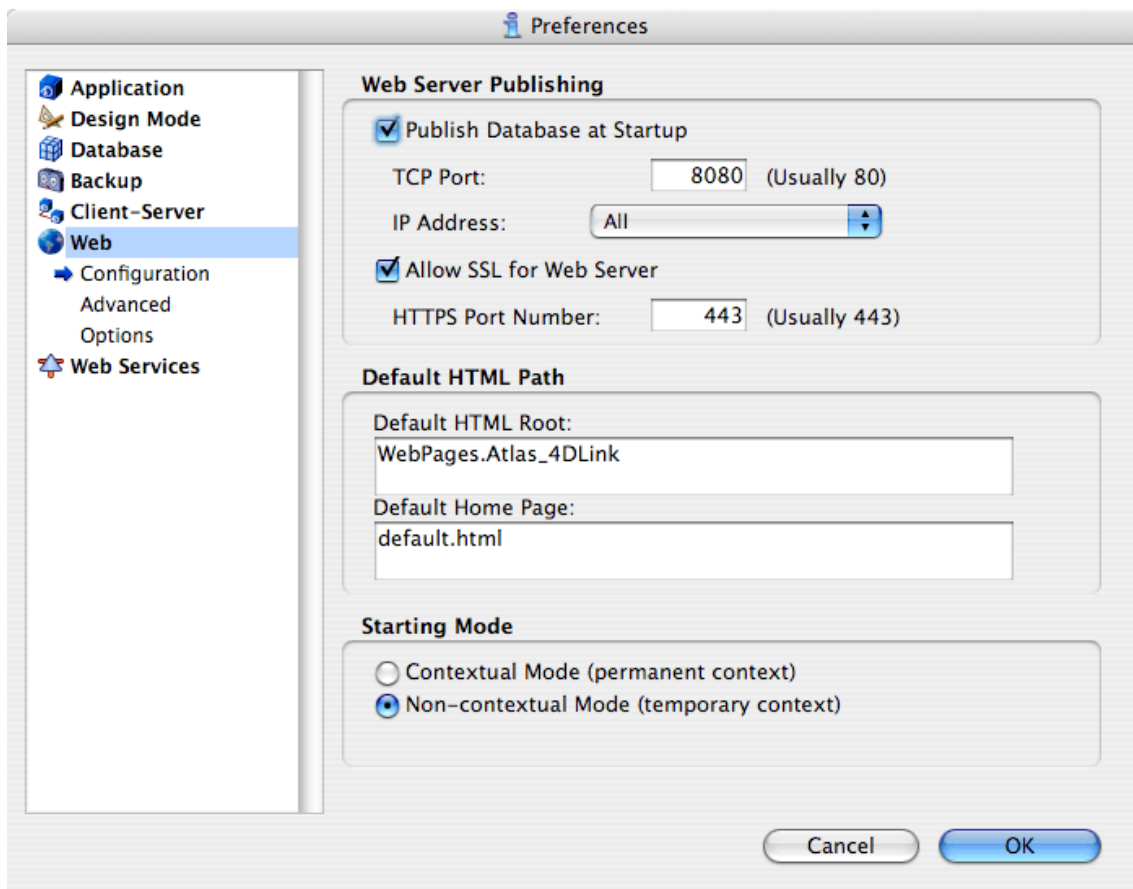


Figure 20.1 The Web Server Publishing panel of the Database Properties dialog

Using 4D and Kerio WebSTAR on the same computer

If you want to use 4D and *Kerio WebSTAR* on the same computer, change the TCP port in the *Web Server Publishing* page in *Database Properties* to another value (i.e., 8080) and configure the settings in the *4D Link* panel in the *Admin Client* application accordingly. *Kerio WebSTAR* will continue to serve on the default port (80) and the plug-in will call 4D on the other port.

20.2 Configuring 4D Link

Once you have the 4D database (or databases) set up, you need to open the *Admin Client* application and set a few configuration parameters:

- The URL or IP address of the machine that is running the database,
- The Port the database uses (default is 80),
- The Timeout value (seconds) used by *4D Link*. If 4D does not respond within the time set by the Timeout parameter, a timeout error message will be sent to the browser.

If you are using more than one database, you also specify an ID parameter for each database. With more than one database, requests are identified by this ID and are automatically directed to the database with the matching ID. For example, requests that begin with */KerioLink2/* will be sent to the database with ID=2.

To configure *4D Link*:

1. Highlight the *4D Link* item in the *Web Server* group in the *Admin Client* application.

The following screen appears.

It includes a default row that you need to modify.

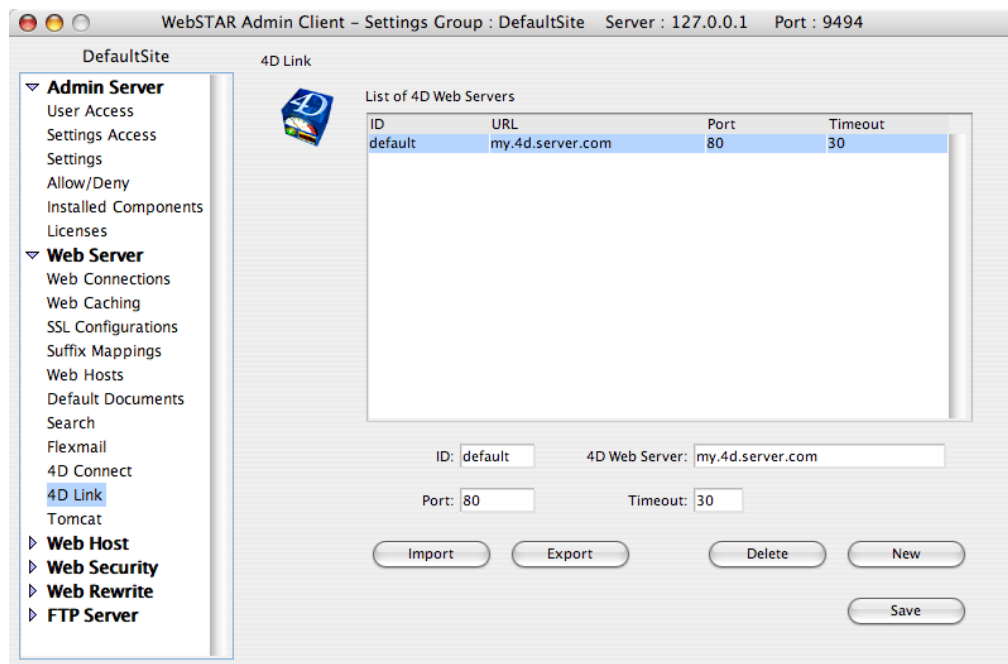


Figure 20.2 The 4D Link panel in Admin Client

2. Click twice in the entry in the URL column and replace the default URL `my.Kerio.server.com` with either the URL or IP address of the machine that is running the database.
3. If necessary, edit the default *Port* entry of 80 to match your setting in the *Database Properties* dialog in the database.
4. If desired, modified the default timeout of 30 seconds.

5. (Optional) If you are using more than one database, set the ID for this database to a number and Click *New* to add the parameters for the next database.

If you're using one database, you don't need to change this setting. Use different IDs for each database and edit calls to *4D Link* within the HTML to match these IDs.

6. Click *Save* to save your changes to the web server.
7. (Optional) Open the *Database Properties* dialog box in 4D and verify that the TCP ports you designated in the *4D Link* panel agree with the settings.
8. (Optional, where applicable) Open the TCP/IP control panel on all computers and verify that TCP/IP is set to *Active* and the option to *Load* only when needed is not selected.

These options are available by clicking the *Options* button in the TCP/IP control panel. If the *Options* button is not on your TCP/IP control panel, choose *Edit* → *User mode* and select the *Advanced* setting.

9. (Optional) Verify that *4D Web Services* are active and running.

You are now ready to use *Kerio WebSTAR* to serve information in 4D databases.

20.3 Overview

The *4D Link* plug-in is designed to pass requests embedded in HTML to 4D for processing. 4D returns the requested information to *Kerio WebSTAR*, which serves the HTML to the browser.

The general scenario is as follows:

- When *Kerio WebSTAR* starts, the *4D Link* plug-in is initialized and tells *Kerio WebSTAR* that it will manage all requests that start with `/4DLINK/` (or, if you are serving more than one database, `/4DLINKx/`, where x is an ID that is defined in the *4D Link* panel).
- When the browser connects to *Kerio WebSTAR*, it serves the page that is defined as the default HTML page. In the examples included with *4D Link*, the home page is

`default.html`. The home page contains one or more URLs that begin with `/4DLINK/`, followed by a 4D tag and any number of optional parameters.

- (Optional) If you have specified more than one database in the *4D Link* panel, then the ID appended to the `/4DLINK/` tag tells *4D Link* which instance of 4D to send the request to.
- 4D receives the URL. The database must recognize, parse, and respond to the information contained in the request. If the 4D instance is not available or does not exist, an error message will be returned to the browser. If references to 4D tags in the request are undefined, an error will be returned in the HTML page.
- *4D Link* will wait for a new HTML page from 4D. If a time-out occurs, an error will be sent to the browser.
- If *4D Link* receives the new HTML page within the time specified by the Timeout parameter on the relevant `.prefs` line, it will be served to the browser.
- The connection between 4D and *Kerio WebSTAR* will be terminated; the web process created on 4D will be closed.

You will generally use the 4DCGI, 4DACTION, and/or 4DSCRIPT tags to make requests.

- 4DCGI executes the On Web Authentication and On Web Connection database methods.
- 4DACTION executes the On Web Authentication database method and the method passed by 4DACTION as the first parameter.
- 4DSCRIPT executes the On Web Authentication database method and the method passed by 4DSCRIPT as a parameter.
- When using 4DACTION, 4DSCRIPT, or 4DCGI, you can pass additional information to 4D in the URL. This information will be placed into \$1. However it is the responsibility of the called method to recognize, parse, and respond to the additional information.

Your 4D code will use commands such as SEND HTML FILE, SEND HTML BLOB, or SEND HTML TEXT to return information to *Kerio WebSTAR*.

In compiled databases, declare `C_TEXT($1;$2;$3;$4;$5;$6)` in the On Web Connection and On Web Authentication methods.

Declare `C_TEXT ($0;$1)` in a method that will be called by a 4DSCRIPT tag. Declare `C_TEXT($1)` in any method that will be called by a 4DACTION tag.

Set `$0:=True` in the On Web Authentication method to accept the connection or `False` to deny this connection.

When 4D receives the 4D tag, it is processed as described in the “Web Services” section of the 4D 6.7 Language Reference. See the Language Reference for complete information on 4D HTML tags.

The following four examples illustrate this basic approach. In the supplied examples, it is assumed that only one instance of 4D is being used.

20.4 4D Link Examples

The *DefaultSite* virtual host includes worked examples of both *4D Link* and *4D Connect*. The *4D Database Integration* folder in the *Examples* folder contains the example databases that are described in this chapter and chapter 21. The folder *WebSTAR 4D Link* contains the examples described here and the folder *Extra Examples* contains one database that illustrates both *4D Link* and *4D Connect* simultaneously. It is a modification of the *Atlas* example that includes the code from this chapter and chapter 21.

To try out the examples, set up a virtual host for each example. Follow the configuration instructions in the `read me` file for each database.

Example 1: Basic Communication

This example serves to illustrate the communication process between *Kerio WebSTAR* and 4D. The default HTML page contains one hyperlink.

Installing the Example

Copy the file `Page1.html` in the *ToCopyOnWebSTAR* folder to the same directory as *Kerio WebSTAR*. Create an alias to this file and rename it `default.html`, replacing the default page in the virtual host.

On the 4D machine, copy the file `Page2.html` to the same folder as the structure file.

Running the Example

Launch *WebSTAR* and 4D. Use a browser to log onto the *WebSTAR* machine. The following page will appear:

Click the link to tell *4D Link* to send the request to 4D. The following screen will appear:

The `default.html` page contains one hyperlink:

```
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=iso-
8859-1">
```

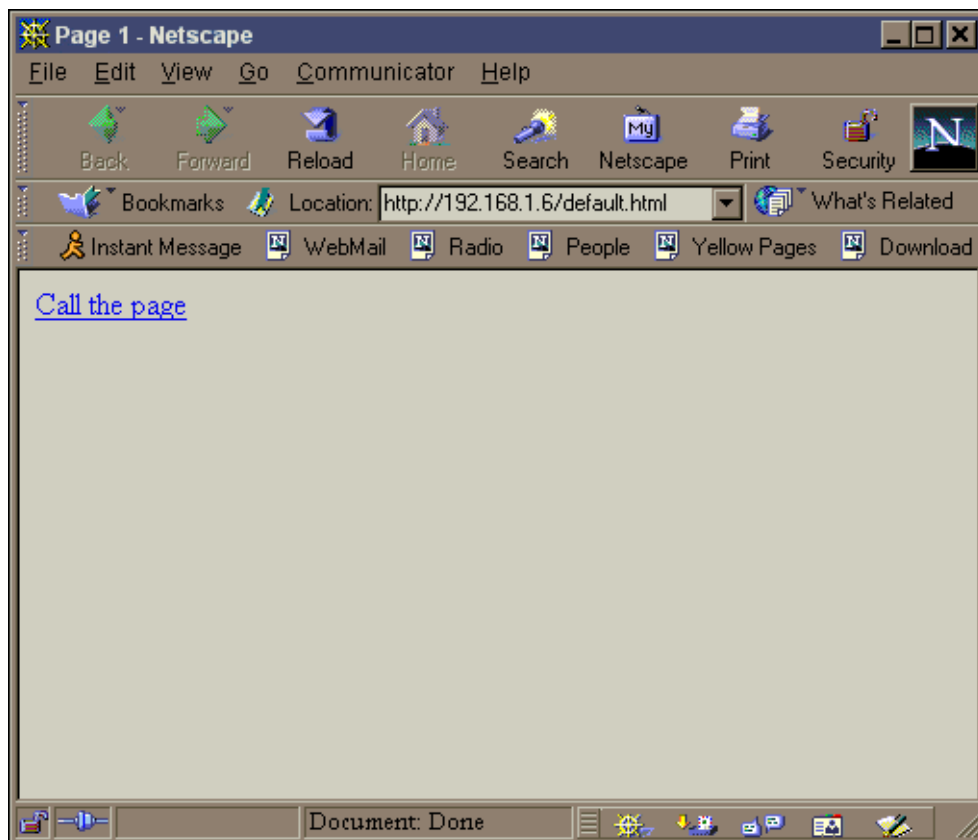


Figure 20.3 The page 1

```

<title>Page 1</title>
<meta http-equiv="pragma" content="no-cache">
</head>
<body>
<a HREF="/4DLINK/4DACTION/WebStar_Call">Call the page</a>
</body>
</html>

```

The link uses the 4D Tag 4DACTION that requests 4D to execute WebStar_Call. This method consists of the line:

```
SEND HTML FILE(Page2.html)
```

The Page2.html document is:

```

<html>
<head>
  <meta http-equiv="content-type" content="text/html;
charset=iso-8859-1">
  <title>Page 2</title>
  <meta http-equiv="pragma" content="no-cache">

```

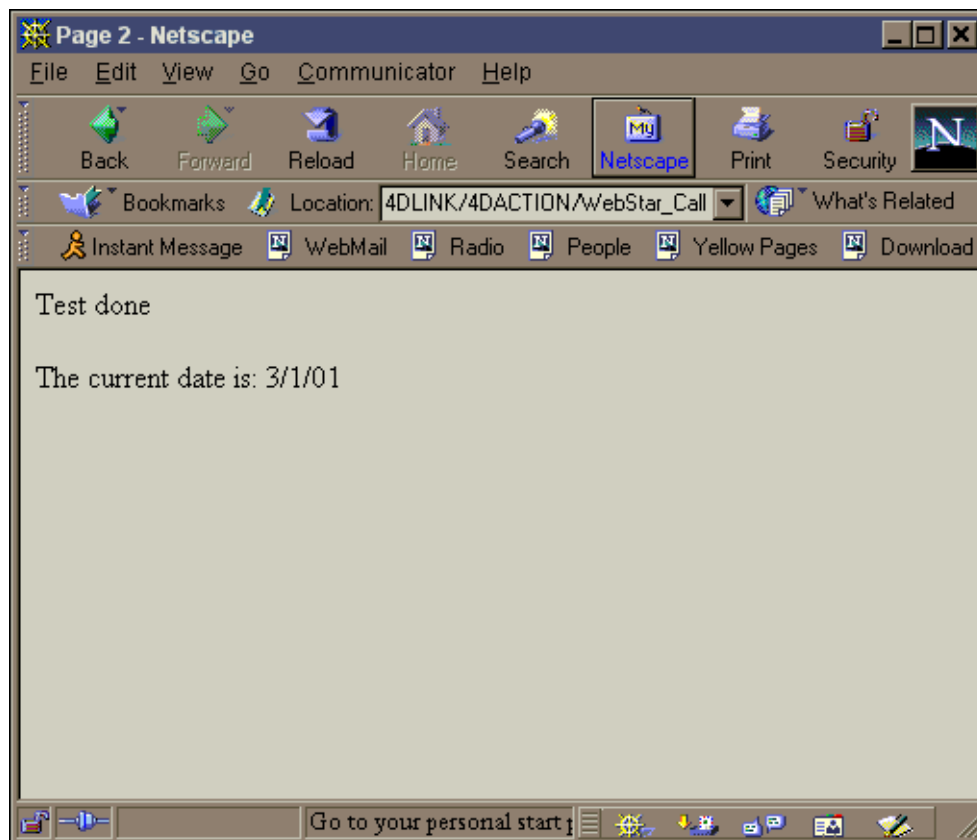


Figure 20.4 The page 2

```
<!--4DSCRIPT/My4DMethod-->
</head>
<body >
<p>Test done</p>
<!--4DVAR MyVariable-->
<!--4DVAR current date-->
</body>
</html>
```

It uses the 4DSCRIPT tag to call My4DMethod, which is:

```
MyVariable:="The current date is:"
```

The two 4DVAR tags in Page2.html return the contents of MyVariable and the value returned by the 4D function Current Date.

Thus, the returned page contains static text from Page2.html, the value of a 4D variable, and the value of a built-in 4D function.

Note: Using 4D functions in your HTML is not recommended because of localization issues. Although 4D allows you to use 4D functions in your HTML, you must be sure that the version of 4D uses the same language that is used in your HTML.

This example requires a US/International version of 4D. If you hardcode the names of any built-in 4D functions in your HTML, you must update the names before opening the database in a foreign-language version of 4D. For example, if you need to provide a French version of this Web site, the French version of 4D will not recognize the string `current date` as a 4D function. The string `current date` would be treated as a variable's name. To avoid the problem, set the value of the function to a variable within 4D, i.e., `MyVariable:="The current date is "+String(Current Date)`. If you know you will never need to localize the Web site, there is no disadvantage to this technique.

Example 2: Basic Forms

This example receives user input via a simple form, does a calculation, and returns the results to the user.

Installing the Example

Copy the file `Page1.html` to a virtual host folder on the *Kerio WebSTAR* machine and create an alias. Rename the alias `default.html`. Copy the file `MyHTMLPage.html` to the same folder as the copy of 4D.

Launch both *Kerio WebSTAR* and 4D.

Running the Example

When the user logs onto the *Kerio WebSTAR* machine, the following form appears:

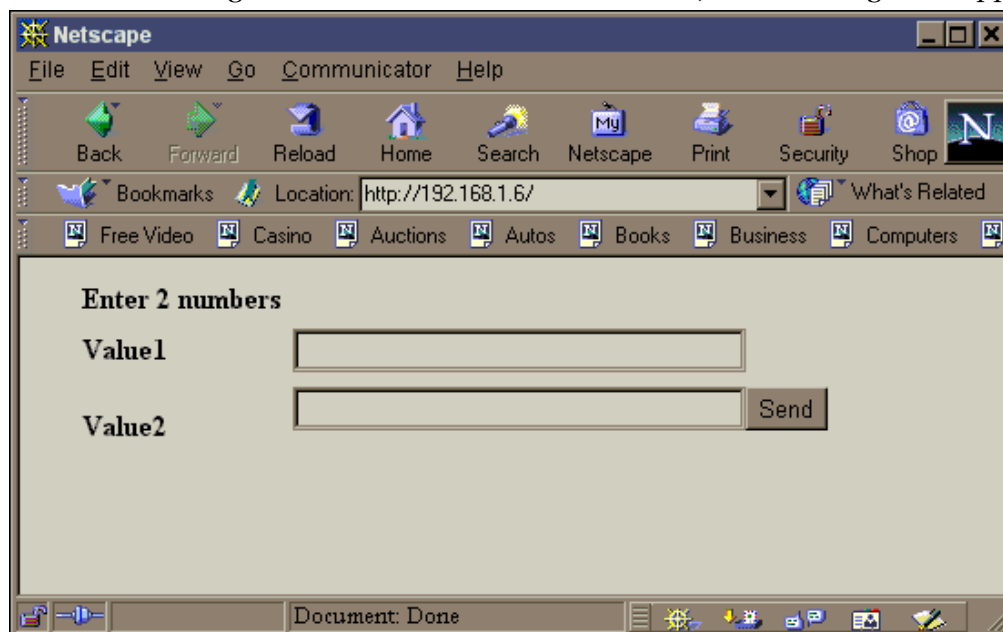


Figure 20.5 Running the Example

When the user enters two numbers and presses *Send*, the results appear in the browser along with the name of the URL that was sent to 4D:

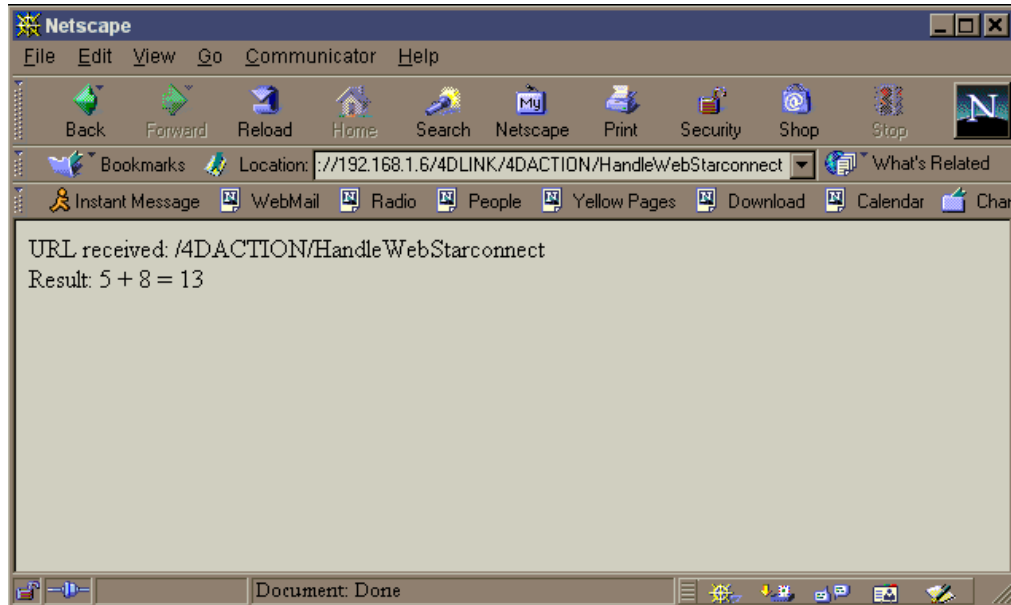


Figure 20.6 Result the Example

The calculation was done by 4D and returned to the user via *Kerio WebSTAR*.

Behind the Scenes

The example was constructed in the following way:

The default `t.html` page is as follows:

```
<html>
<head>
  <meta http-equiv="content-type" content="text/html;
charset=iso-8859-1">
  <meta http-equiv="pragma" content="no-cache">
</head>

<body>
<table>
  <tr>
    <td width="15"></td>
    <td><b>Enter 2 numbers</b></font></td>
    <td></td>
  </tr>
  <form method="POST" action="/4DLINK/4DACTION/
    HandleWebStarconnection">
  <tr>
```

```

        <td width="15"></td>
        <td><b>Value1</b></td>
        <td><input type="text" name="Value1" size="30"></td>
    </tr>
    <tr>
        <td width="15"></td>
        <td><b>Value2</b></td>
        <td><input type="text" name="Value2" size="30"><input
type="submit" name="Submit" value="Send"></td>
    </tr></form>
</table>
</body>
</html>

```

When *Kerio WebSTAR* sends this request to 4D, the On Web Authentication method is called automatically, which extracts the URL of the web page received and puts it into a variable:

```
C_TEXT($1;$2;$3;$4;$5;$6)
```

```
URL_received:=$1
```

```
$0:=True 'accept the connection
```

The `HandleWebStarConnection` method takes the contents of `Value1` and `Value2`, defined in `default.html`, and adds them. The `My4DResult` variable also contains the URL sent to 4D via *Kerio WebSTAR* in the variable `URL_Received`. The last line sends the `MyHTMLPage.html` page to *WebSTAR*.

```

    ' HandleWebStarConnection
    C_TEXT($0;$1)
    C_TEXT(My4DResult;Value1;Value2)
    C_REAL($Compute)
    $Compute:=Num(Value1)+Num(Value2)
    My4DResult:="URL received: "+URL_Received+"<BR>"
    My4DResult:=My4DResult+"Result: "+Value1+" + "+Value2+" =
    "+String($Compute)SEND HTML FILE("MyHTMLPage.html")

```

The `MyHTMLPage.html` file contains a `4DHTMLVAR` tag that returns the contents of the `My4DResult` variable:

```

<HTML>
<HEAD>
<meta http-equiv="pragma" content="no-cache"/>
</HEAD>
<BODY>
<!--4DHTMLVAR My4DResult-->

```

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

Example 3: 4D HTML Tags

Example 3 is an example of the use of the 4D HTML tags used to send requests to 4D. It also illustrates the 4DINCLUDE, 4DIF, and 4DLOOP tags.

Installing the Example

Copy the folder *WebPages.4DTag_4DLink* in the *ToCopyonWebSTAR* folder to the a virtual host folder on the *Kerio WebSTAR* machine. Create an alias to the document *4DTag_4DLink.html* and rename the alias *default.html*. Move the *default.html* file to the same level as the *Kerio WebSTAR* application.

Copy the file *HTML2.txt* and the folder *WebPages.4DTag_4DLink* to the same level as 4D. The latter folder contains the folder *Pages* that contains the files *ActionOK.html*, *PasswordOK.html*, and *HTML3.html*.

Launch the *Kerio WebSTAR* and 4D applications.

Running the Example

When the browser logs onto the *Kerio WebSTAR* machine, the following screen appears:

The *default.html* page contains all the calls to *4D Link*:

```
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
charset=iso-8859-1">
    <meta name="generator" content="Adobe GoLive 4">
    <title>4D Tags for 4D Link</title>
    <meta http-equiv="pragma" content="no-cache">
  </head>
  <body>
    <br>
    <br>
    <a href="/4DLink/4DCGI/M_CGI/parameter1/parameter2">Password
system via 4D CGI</a><br>
    <p><a
href="/4DLink/4DACTION/M_ACTION/parameter1/parameter2"
>HTML Page via 4D ACTION</a><br>
    </p>
    <p><a href="/4DLink/4DStats">4DStats</a><br>
    </p>
```

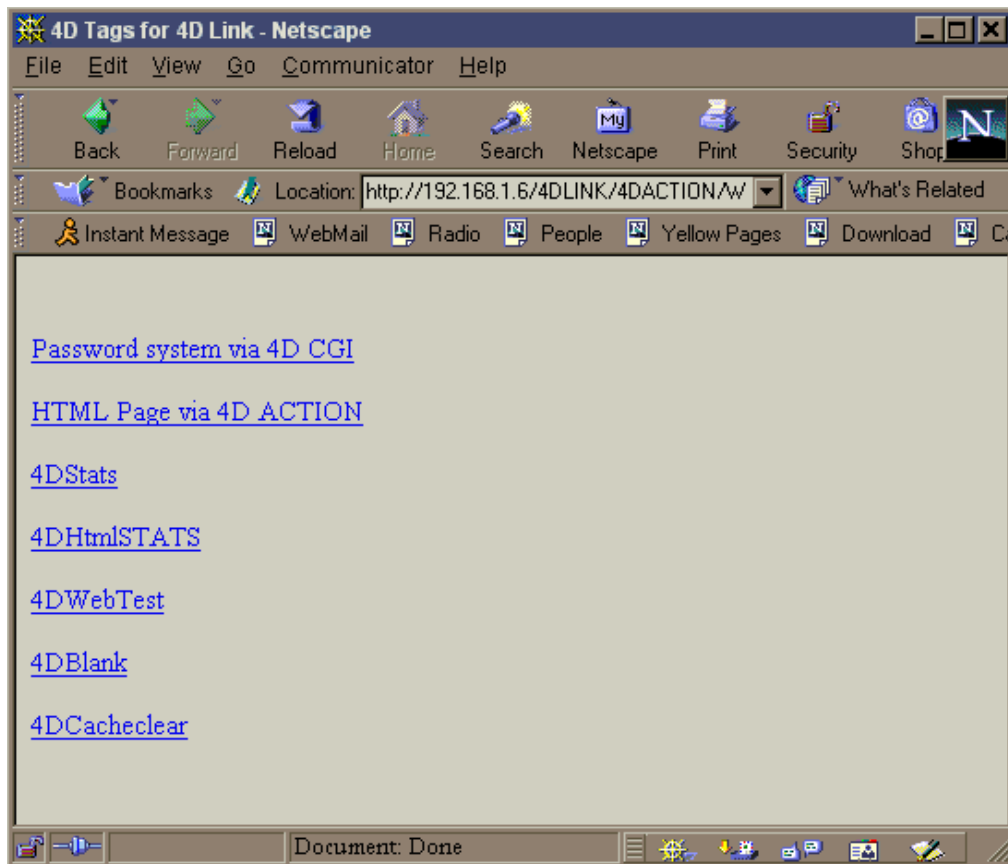


Figure 20.7 Running the Example

```

<p><a href="/4DLink/4DHtmlStats">4DHtmlSTATS</a><br>
</p>
<p><a href="/4DLink/4DWebtest">4DWebTest</a><br>
</p>
<p><a href="/4DLink/4DBlank">4DBlank</a><br>
</p>
<p><a href="/4DLink/4DCacheClear">4DCacheClear</a><br>
</body>
</html>

```

When the user clicks on the HTML Page via 4DACTION link, the following screen appears:

The 4DACTION URL calls M_Action which serves the ActionOK.html page:

```

' M_Action
C_TEXT($1;$2;$3;$4;$5;$6)
' Called by a /4DACTION
' In $1, you can retrieve the 2 parameters

```

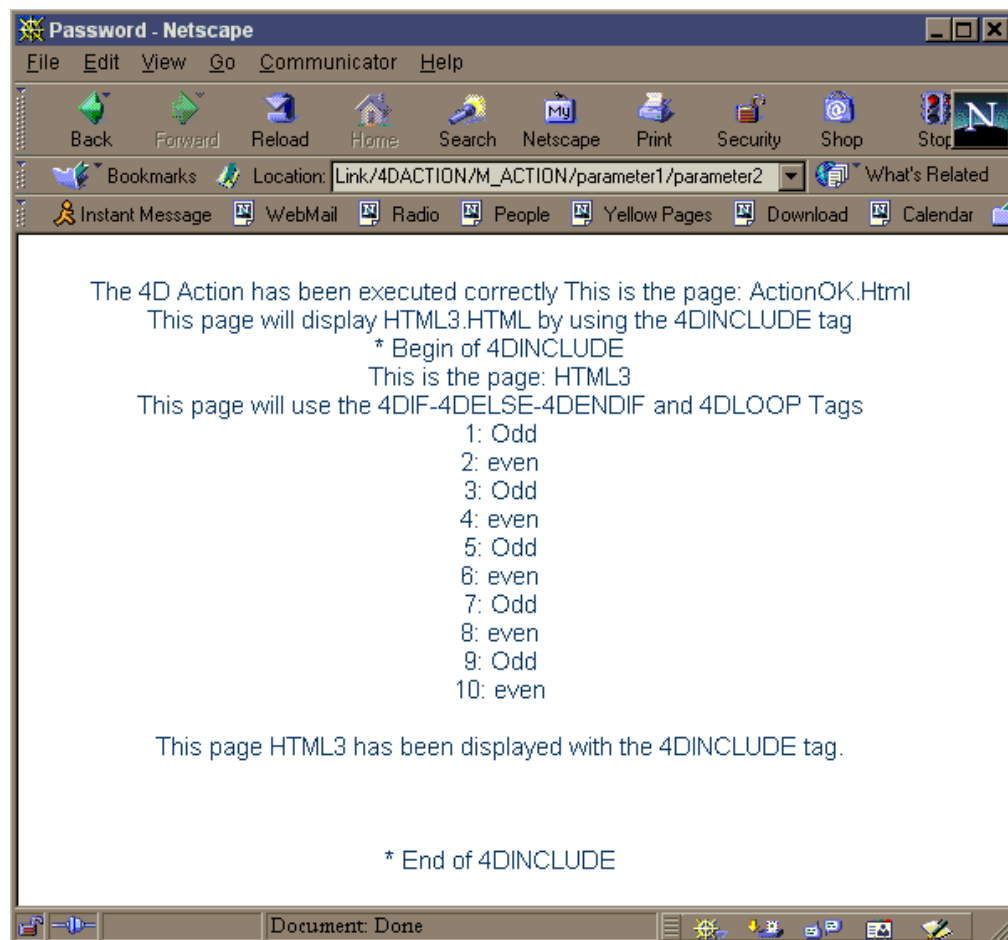


Figure 20.8 Result of Example

```
MyVarBodyHTML1:=""
Loadtext ("HTML2.txt";->MyVarBodyHTML1)
ARRAY INTEGER(arr_Integer;10)

' Array used in HTML3.html
For ($i;1;10)
  Arr_Integer{$i}:=$i
End for

SEND HTML FILE("pages/ActionOK.html")
The ActionOK.html file uses the 4DINCLUDE tag to call HTML3.html.
<html>
<head>
  <meta http-equiv="content-type" content="text/html;
```

```

charset=iso-8859-1">
  <title>Password</title>
  <meta http-equiv="pragma" content="no-cache">
</head>

  <body background="/WebPages.4DTag_4DLink/pages/Images/
background.gif" bgcolor="white">
  <center>
    <br>
    <font face="Arial,Helvetica,Geneva,Swiss,SunSans-Regular"
color="#003366">The 4D Action has been executed correctly

```

```

This is the page: ActionOK.html<br>
This page will display HTML3.HTML by using
the 4DINCLUDE tag<br>
* Begin of 4DINCLUDE<br>
<!--4DINCLUDE HTML3.HTML-->
* End of 4DINCLUDE<br>
</font></center>
</body>
</html>

```

The HTML3.html page demonstrates the use of the 4DLOOP and 4DIF...4DELSE...4DENDIF tags. The 4DIF statement determines whether the ith element in Arr_Integer is even or odd and then returns the value of the ith element and either the word Even or Odd.

```

<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
charset=ISO-8859-1">
    <meta name="generator" content="Adobe GoLive 5">
    <title>Welcome to Adobe GoLive 5</title>
  </head>
  <body bgcolor="#ffffff">
    This is the page: HTML3<br>
    This page will use the 4DIF-4DELSE-4DENDIF and 4DLOOP
    Tags<br>
    <!--4DLOOP Arr_Integer-->
    <!--4DIF Mod(Arr_Integer{Arr_Integer};2)=0--><!--4DVAR
Arr_Integer{Arr_Integer}-->: even<br>
    <!--4DELSE--><!--4DVAR Arr_Integer{Arr_Integer}-->:

```

```
Odd<br>
<!--4DENDIF-->
<!--4DENDLOOP-->
<br>
This page HTML3 has been displayed with the 4DINCLUDE
tag.<br>
<br>
<br>
<br>
</body>
</html>
```

Thus, the page that is returned includes static text from `ActionOK.html` and the output from `HTML3.html`.

Example 4: 4D Atlas

This is a more realistic example that supports user queries from the browser, extracts the requested data from the database, and serves the requested data via *Kerio WebSTAR*. Pages are a mixture of static information served directly by *Kerio WebSTAR* and dynamic information that is served on demand via 4D.

Installing the Example

Copy the contents of the folder *WebPages.Atlas_4DLink* in the *ToCopyOnWebSTAR* folder to a virtual host folder. Inside this folder, the *Pages* folder contains the default page. Create an alias of `Default.html`, move the alias to the root of the virtual host folder on and rename it `Default.html`.

On the 4D machine, move the folder *WebPages.Atlas_4DLink* to the same folder as the 4D. This folder contains a *Pages* folder that contains the following files:

- `WebPages.Atlas_4DLink\Pages\SearchCountry.html`
- `WebPages.Atlas_4DLink\Pages\NoRecords.html`
- `WebPages.Atlas_4DLink\Pages\InvalidContext.html`
- `WebPages.Atlas_4DLink\Pages>EditCountry.html`
- `WebPages.Atlas_4DLink\Pages\HeaderHTML3.txt`
- `WebPages.Atlas_4DLink\Pages\FooterHTML3.txt`
- `WebPages.Atlas_4DLink\Pages\ListRecordsCountry.html`

When this example displays a record for a country, it includes a picture field that shows the country's flag. The `SEND HTML BLOB` command is used to serve the image.

Launch *Kerio WebSTAR* and 4D.

Running the Example

When the user logs onto the *Kerio WebSTAR* server, the following screen appears:

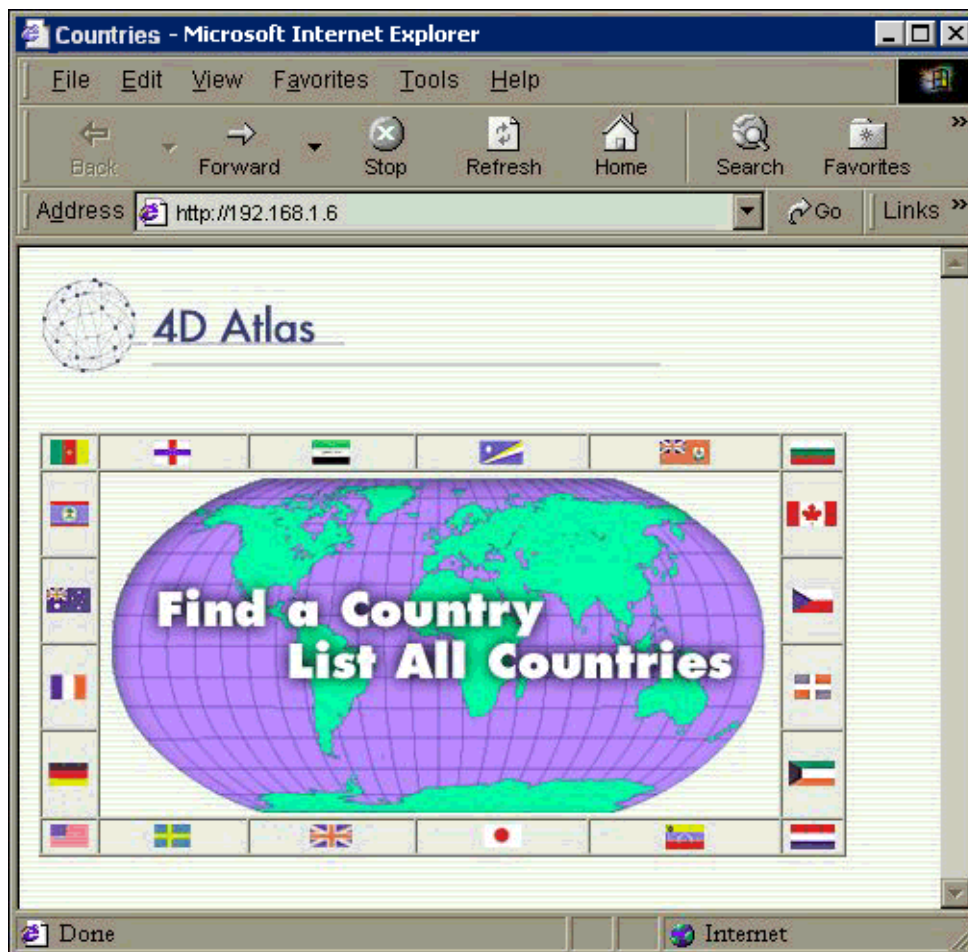


Figure 20.9 Running the Example

The images are served directly by *Kerio WebSTAR* without any involvement of *4D Link*.

The key lines in the default.html page are:

```
href="/4DLINK/4DACTION/HandleWebSTARConnection/4DNC
&WEB_AllRecords&1&A"
```

and

```
href="/4DLINK/4DACTION/HandleWebSTARConnection/4DNC
&WEB_InitSearch&1"
```

for the List All Countries and Find A Country links, respectively.

Each request is done using a URL that begins with /4DLINK/, the 4DACTION tag, the name of the 4D method to execute, and several parameters.

When you pass extra parameters in the URL, it is your responsibility to parse the string and extract the parameters. In this database, On Web Authentication contains a call to *IsNewConnection*, which parses the request.

The HandleWebSTARConnection method executes the method passed as the first parameter.

```
'HandleWebSTARConnection
C_TEXT($0;$1)

C_TEXT(MyVarHeaderHTML;MyVarFooterHTML;MyVarBodyHTML1;
      MyVarBodyHTML3)
MyVarFooterHTML:=""
MyVarBodyHTML3:=""
If (IndiceContext>0) 'ContextID
  Case of
    : (MethodToCall="WEB_InitSearch") ' A search has been
      requested WEB_InitSearch
    : (MethodToCall="WEB_SearchRecords") ' A search has been
      submitted WEB_SearchRecords
    : (MethodToCall="WEB_ShowRecord") ' A record needs to
      be edited WEB_ShowRecord
    : (MethodToCall="WEB_NewOrder") ' A subselection has
      been requested WEB_NewOrder
    : (MethodToCall="WEB_AllRecords") ' All records
      requested WEB_AllRecords
  End case
Else

  ' The contextID is not valid
  For ($i;1;Count tables)
    REDUCE SELECTION(Table($I)->;0)
  End for

  ' Display banner + error message
  INTERN_Fill_VarHTML ("WebPages.Atlas_4DLink"+SepDir+"Pages"+
  SepDir+"HeaderHTML3.txt";->MyVarHeaderHTML)
  INTERN_Fill_VarHTML ("WebPages.Atlas_4DLink"+SepDir+"Pages"+
  SepDir+"InvalidContext.txt";->MyVarBodyHTML1)
  SEND HTML FILE("pages/InvalidContext.html")
End if
```

For a new connection, the ContextID is set to 4DNC. The ContextID is used to manage lengthy selections of records that must be returned to the browser in successive subsets. 4D can store the selection in named selections or sets. The problem is that, after a page is returned, the connection is killed and the selection is lost. The

ContextID allows us to keep the selection on the web server and allows the same browser to retrieve successive groups of records from that selection. Each time a browser logs in with a particular ContextID, it will get its selection.

With *4D Link*, you cannot use the IP address as the ContextID since all requests come from the *Kerio WebSTAR* machine. In this database, the ContextID is built from random numbers in the *IsNewConnection* method. Each time that web browser connects to the site, that browser's ContextID is provided in the request. This enables 4D to use the selection associated with that browser.

The following code in *IsNewConnection* manages the ContextID:

```
If (ContextID="4DNC") ' A new connection
  ContextID:=String(Random)+String(Random) ' An ID will be
  created
  ' Define a semaphore to add element into this array
  INTERN_SetSemaphore ("Arrays")
  IndiceContext:=Size of array(TContextes)+1
  ' Keep this context ID in memory for further connections
  INSERT ELEMENT(TContextes;IndiceContext;1)
  TContextes{IndiceContext}:=ContextID
  INSERT ELEMENT(TContextSets;IndiceContext;1)
  INTERN_ReleaseSemaphore ("Arrays")
Else ' The context already exists
  IndiceContext:=Find in array(TContextes;ContextID)
End if
```

In *WEB_SearchRecords*, a 4D set is given a temporary name using the *Random* function. This name is stored in the *DebileName* variable.

The name of this selection is then stored in an array (*TContextSets*) with the same ID of the contextID (*TContextes*).

```
' Name of the selection of the current connection
DebileName:="<>S"+String(Random)+String(Random)
INTERN_SetSemaphore ("Arrays")
$asd:=Size of array(TContextSets{IndiceContext})+1
' Will keep it until next query
INSERT ELEMENT(TContextSets{IndiceContext};$asd;1)
TContextSets{IndiceContext}{$asd}:=DebileName
INTERN_ReleaseSemaphore ("Arrays")
```

Once the selection has been created, a set is created from this selection. When a connection is performed, *IsNewConnection* checks the index of the contextID in the *TContextes* array. If this ContextID exists, this will give us the index of the set to be used. Then, a *USE SET* is executed to retrieve this selection. Saving and retrieving

the current selection is done in WEB_SearchRecords.

Depending on what the browser has requested, HandleWebSTARConnection calls another method that either presents a search screen or lists all records in groups. In either event, the HTML is returned by a call to SEND HTML FILE within that method.

Find A Country

If the user clicked on Find A Country, WEB_InitSearch sends the page SearchCountry.html. The SearchCountry.html file contains the html to draw the *Search* dialog:

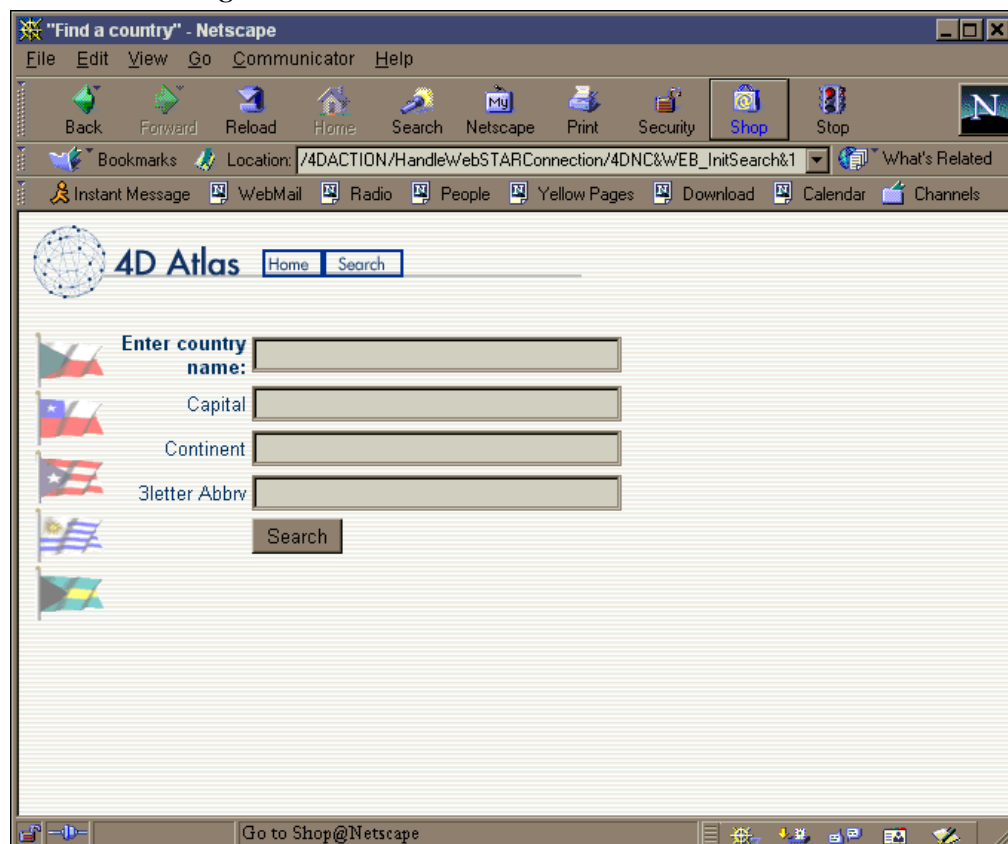


Figure 20.10 Find A Country dialog

```
'Web_InitSearch
```

```
Case of
```

```
: (WebTable=1)
```

```
  ' header of the page
```

```
    INTERN_Fill_VarHTML ("WebPages.Atlas_4DLink"+SepDir+"Pages"+  
    SepDir+"HeaderHTML3.txt";->MyVarHeaderHTML)
```

```
    SEND HTML FILE("Pages/SearchCountry.html")
```

```
End case
```

When the user validates that screen, the `WEB_SearchRecords` method is called via a URL:

```
action="/4DLINK/4DACTION/HandleWebSTARConnection/ <!--4DVAR
ContextID-->&WEB_SearchRecords&<!--4DVAR WEBTable-->&--">
WEBSearchRecords performs the requested query, loads the selected records into a
set, and displays the first group of records using ListRecords.html:
```

```
'WEB_SearchRecords
Web_InitCommonVar
pTable:=Table(WebTable)
We will create a set (DebileName) that will be attached to the Context ID.
DebileName:="<>S"+String(Random)+String(Random)
INTERN_SetSemaphore ("Arrays")
$asd:=Size of array(TContextSets{IndiceContext})+1
INSERT ELEMENT(TContextSets{IndiceContext};$asd;1)
TContextSets{IndiceContext}{$asd}:=DebileName
INTERN_ReleaseSemaphore ("Arrays")
```

Case of

```
: (WebTable=1) ' Search a selection of countries
WebTitle:="Searching a country"
ALL RECORDS([Countries])
If (tCountry# "")
    QUERY SELECTION([Countries];[Countries]Country=tCountry)
End if
If (tContinent# "")
    QUERY SELECTION([Countries];[Countries]Continent=
    tContinent)
End if
If (tCapital# "")
    QUERY SELECTION([Countries];[Countries]Capital=tCapital)
End if
If (tSigne# "")
    QUERY SELECTION([Countries];[Countries]Signe=tSigne)
End if
```

End case

```
CREATE SET(pTable->;DebileName)
```

```
LRecordsinSelection:=Records in selection(pTable->)

' Display a bunch of records
If (LRecordsinSelection>0)
  Case of
  : (WebTable=1)
    If (TVariables{1}="-")
      ORDER BY([Countries];[Countries]Country)
      TVariables{1}:=[Countries]Country<1>
      QUERY SELECTION([Countries];[Countries]Country=
        TVariables{1}+"@")
    Else
      If (TVariables{1}#"*")
        QUERY SELECTION([Countries];[Countries]Country=
          TVariables{1}+"@")
      End if
    End if
    LStartSel:=Records in selection([Countries])
    WEB_FillArrays_Countries
  End case
Else

  ' No records
End if
Case of
: (WebTable=1)
  If (LRecordsinSelection>0)
    SEND HTML FILE("pages/ListRecordsCountry.html")
    INTERN_Fill_VarHTML ("WebPages.Atlas_4DLink"+SepDir+"Pages"+
      SepDir+"FooterHTML3.txt";->MyVarFooterHTML)
  Else
    SEND HTML FILE("pages/Norecords.html")
    ' No records
  End if
End case
INTERN_Fill_VarHTML ("WebPages.Atlas_4DLink"+SepDir+"Pages"+
  SepDir+"HeaderHTML3.txt";->MyVarHeaderHTML)
The ListRecordsCountry.html file contains the html for returning a list of coun-
tries to the browser. A 4DLOOP...4DENDLOOP is used to list the countries:
```

```

<html>
  <head>
    <meta http-equiv="content-type"
content="text/html; charset=iso-8859-1">
    <title><!--4DVAR WebTitle--></title>
    <meta http-equiv="pragma" content="no-cache">
  </head>

  <body bgcolor="white" link="#003399" alink="red"
vlink="#ff00ff"
background="/WebPages.Atlas_4DLink/pages/images/
background.gif">
    <!--4DHTMLVAR MyVarHeaderHTML-->

    <table border="0" cellpadding="0" cellspacing="0" width="400">
      <tr>
        <td width="56" align="top" valign="left"></td>
        <td width="344" align="top" valign="left">
          <table border="0" cellpadding="0" cellspacing="0"
width="100%">

            <td align="left" valign="top"><font
face="Arial,Helvetica, Geneva,Swiss,SunSans-Regular" size="2"
color="#003366"><b><br>
              </b><b>Records Shown <!--4DVAR LStartSel-->
              of <!--4DVAR LRecordsInSelection-->
              Record<!--4DIF (LRecordsInSelection>1)-->
              s <!--4DENDIF-->
              Found.</b></font>
            <p></td>

            <p>
              <!--4DLOOP atNameDisp-->
            <TR>
              <TD>
                <A HREF="/4DLINK/4DACTION/HandleWebSTARConnection/
<!--4DVAR ContextID-->&WEB_ShowRecord&<!--4DVAR WEBTable-->&
<!--4DVAR atIDDisp{atNameDisp}-->">
                <!--4DVAR atNameDisp{atNameDisp}--></A><P>

```

```

        </TD>
    </TR>
<!--4DENDLOOP-->

    <p>
    <tr>
        <TD>
        <table border="0" cellspacing="0" cellpadding="3">
        <tr>
        <!--4DLOOP atLetterOrder-->
        <td>

            <A HREF="/4DLINK/4DACTION/HandleWebSTARConnection/<!--4DVAR
ContextID-->&WEB_NewOrder&<!--4DVAR WebTable-->&<!--4DVAR
DebileName-->&<!--4DVAR atLetterOrder{atLetterOrder}-->"><!--4DVAR
atLetterOrder{atLetterOrder}--></A>
        </td>
        <!--4DENDLOOP-->
        </tr>
        </table>
        </TD>
    </tr>
    </table>
</td>
</tr>
</table>
</body>
</html>

```

A typical screen looks like this:

If the user clicks a country, it is displayed via WebShowRecord:

```

    ' WebShowRecord
pTable:=Table(WebTable)
Case of
    : (WebTable=1)
    WebTitle:="Displaying a country"
    QUERY([Countries];[Countries]ID_Country=Num(TVariables{1}))
    If (Record number([Countries])=0)
        ' Record not found
        SEND HTML FILE("pages/Norecord.html")
    Else

```

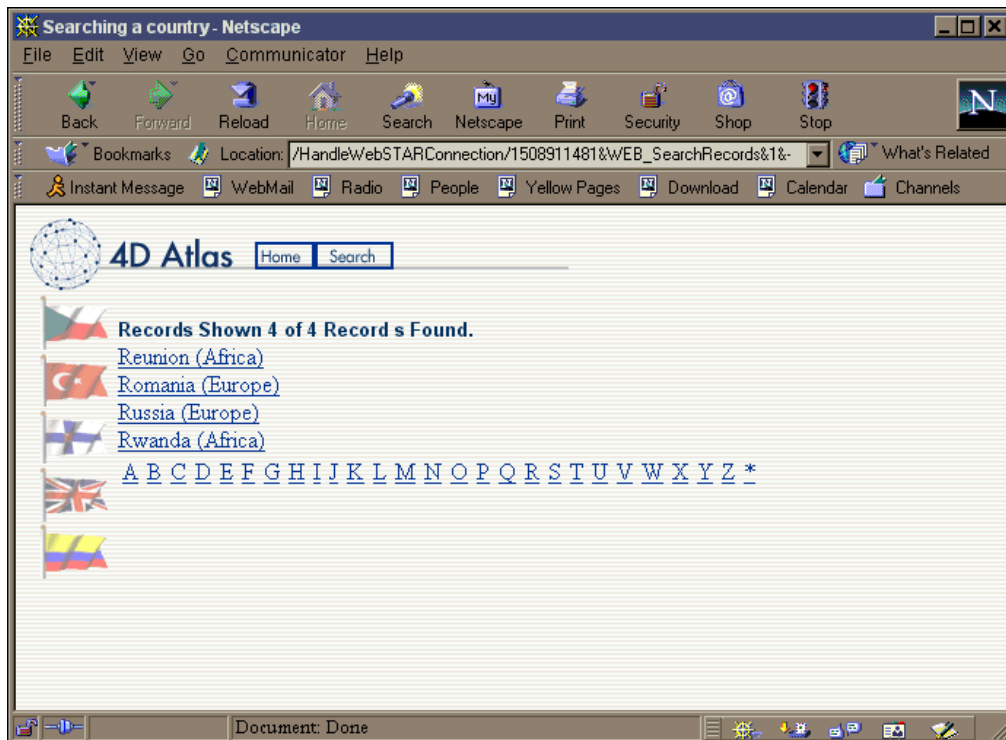


Figure 20.11 4D Atlas

```

' Record found
INTERN_Fill_VarHTML ("WebPages.Atlas_4DLink"+SepDir+"Pages"
+SepDir+"HeaderHTML3.txt";->MyVarHeaderHTML)
SEND HTML FILE("pages/EditCountry.html")
End if
End case
The EditCountry.html file contains the html to display a single record:

```

```

<html>
  <head>
    <meta http-equiv="content-type"
content="text/html; charset=iso-8859-1">
    <title><!--4DVAR WebTitle--></title>
    <meta http-equiv="pragma" content="no-cache">
  </head>

  <body bgcolor="white" link="#003399" alink="red"
vlink="#ff00ff" background="/WebPages.Atlas_4DLink
/pages/images/background.gif">
  <!--4DHTMLVAR MyVarHeaderHTML-->

```

```
<table border="0" cellpadding="0" cellspacing="0" width="400">
  <tr>
    <td width="56" rowspan="6" align="left" valign="top">
</td>
    <td colspan="3"><font
face="Arial,Helvetica,Geneva,Swiss,SunSans-Regular"
color="#003366"><b>Country Information</b></font></td>
  </tr>
  <tr>
    <td width="187"><font size="2"
face="Arial,Helvetica,Geneva,Swiss,SunSans-Regular"
color="#003366">Country</font></td>
    <td width="79"><font size="2"
face="Arial,Helvetica,Geneva,Swiss,SunSans-Regular"><!--4DVAR
[Countries]Country-->
    </font></td>
    <td width="78"><font size="2"
face="Arial,Helvetica,Geneva,Swiss,SunSans-Regular">
    </font></td>
  </tr>
  <tr>
    <td width="187" ><font size="2"
face="Arial,Helvetica,Geneva,Swiss,SunSans-Regular"
color="#003366">Continent</font></td>
    <td colspan="2"><font size="2"
face="Arial,Helvetica,Geneva,Swiss,SunSans-Regular"><!--4DVAR
[Countries]Continent-->
    </font></td>
    <td width="78"></td>
  </tr>
  <tr>
    <td width="187"><font size="2"
face="Arial,Helvetica,Geneva,Swiss,SunSans-Regular"
color="#003366">Capital city</font></td>
    <td colspan="2"><font size="2"
face="Arial,Helvetica,Geneva,Swiss,SunSans-Regular"><!--4DVAR
```

```

[Countries]Capital-->
    </font></td>
    <td width="78"></td>
</tr>
<tr>
    <td ><font size="2"
face="Arial,Helvetica,Geneva,Swiss,SunSans-Regular"
color="#003366">Main cities</font></td>
    <td colspan="2"><font size="2"
face="Arial,Helvetica,Geneva,Swiss,SunSans-Regular"><!--4DVAR
[Countries]Main cities-->
    </font></td>

    </tr>

    <tr>
    <td width="187"><font size="2"
face="Arial,Helvetica,Geneva,Swiss,SunSans-Regular"
color="#003366">3letter Abbrv</font></td>
    <td colspan="2"><font size="2"
face="Arial,Helvetica,Geneva,Swiss,SunSans-Regular"><!--4DVAR
[Countries]Signe-->
    </font></td>
    <td width="78"></td>
</tr>
</table>
</body>
</html>

```

HTML that sends the image in the [Countries]Flag field calls WEB_DisplayPicture which uses SEND HTML BLOB after first converting the image to a GIF. If you want to send an image using a 4DVAR tag (as with the other fields in this example) you need to have *4D Connect* installed. Both *4D Connect* and *4D Link* can be installed simultaneously. Each plug-in is configured with its own .prefs file.

‘Project Method: WEB_DisplayPicture

‘Sends the contents of [Countries]Flag using SEND HTML BLOB

C_TEXT(\${1})

C_PICTURE(\$gPict)

C_BLOB(\$xGif)

Case of

```
: (WebTable=1)
```

```
WebTitle:="Displaying a country"
```

```
QUERY([Countries];[Countries]ID_Country=Num(TVariables{1}))
```

```
PICTURE TO GIF([Countries]Flag;$xGif)
```

```
SEND HTML BLOB($xGif;"image/gif")
```

End case

The following screen shows a randomly selected country record:

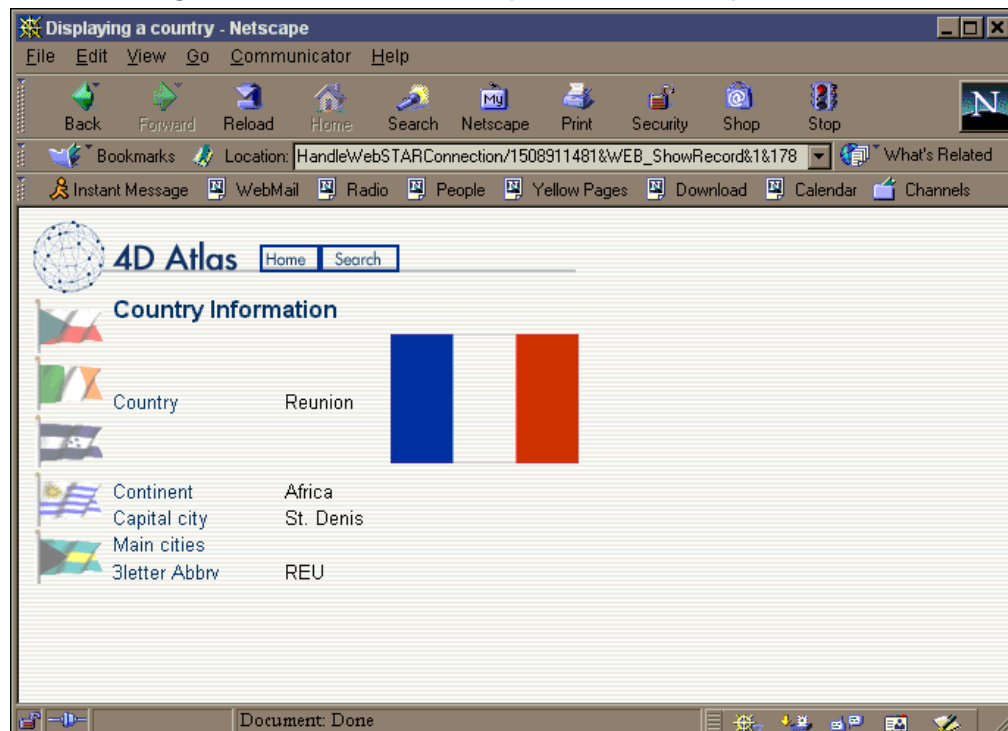


Figure 20.12 A randomly selected country record

The URL in the browser shows the values of the parameters passed to 4D.

List All Records

If the user clicks the List All Records link, HandleWebSTARConnection calls WEB_AllRecords, which sets all the search fields to @ and calls WEB_SearchRecords. WEB_SearchRecords uses the logic that was just reviewed to display all records in groups using ListRecords.html. The user can click any record to view the country information using EditCountry.html.

20.5 Troubleshooting

If the browser can connect to *Kerio WebSTAR* but cannot connect to 4D via *4D Link*, check your TCP/IP configuration on the 4D machine (assuming you are running Mac OS classic). Be sure that the *Load only when needed* checkbox is not checked in the *TCP/IP Control* panels on the 4D machine. To check this point, do the following:

- From the *TCP/IP Control* panel, click the *Options* button in the bottom-right corner of the dialog. If this button is not in the dialog, choose *Edit* → *User Mode* from the *Edit* menu and select the *Advanced mode*.

The *Options* dialog box is shown below:

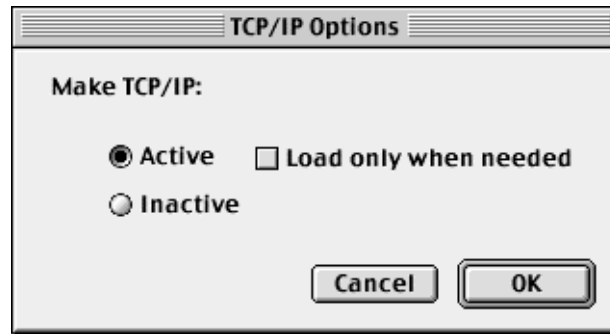


Figure 20.13 The TCP/IP Options dialog box (Mac OS classic)

It must be configured as shown here: *Active* must be selected and *Load only when needed* must be unchecked.

- Be sure that the IP address and port you specified in the *4D Link* panel in *Admin Client* matches the settings in the *Database Properties* dialog in the copy of 4D that you are trying to access.
- Be sure that 4D's web server is running. You can start it automatically by selecting the *Publish Database* at *Startup* preference in *Database Properties*.

You can also start the web server from the user environment by choosing *Web Server* → *Start Web Server*. This menu item is disabled if the web server is running.

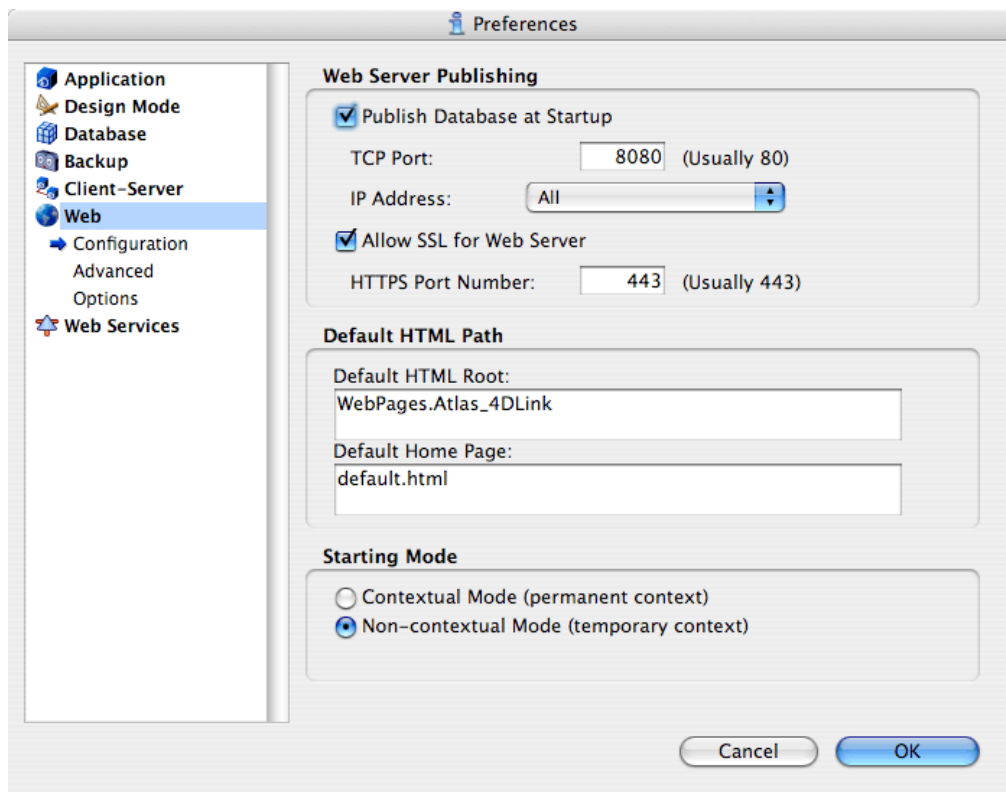


Figure 20.14 Database Properties dialog

Chapter 21

Using 4D Connect

4D Connect is a *Kerio WebSTAR* plug-in that allows you to use *Kerio WebSTAR* as the web server for a 4D database. With *4D Connect*, you can use *4th Dimension* to serve dynamic pages and *WebSTAR* to serve your static pages. When requests that contain a *4D Connect* directive are received from a browser, *4D Connect* passes the request on to the *4th Dimension* database along with an HTML template file. 4D then returns the requested information to *WebSTAR* by replacing the placeholders in the template with values from the database.

4D Connect provides essentially equivalent functionality as *4D Link* (chapter 20), but the latter does not use the template approach. Both *4D Link* and *4D Connect* can be used simultaneously.

21.1 4D Connect Requirements

A dynamic web site that uses *4D Connect* can be implemented on either one or two computers. For optimum performance, install *Kerio Webs STAR* and 4D on separate computers. If one computer is used, 4D and *Kerio WebSTAR* must use different TCP ports.

The 4D application that acts as the web server must have *4D Web Extension* or *4D Server Web Extension* installed. *4D Connect* requires *4th Dimension* version 6.7.1 or above.

One copy of *Kerio WebSTAR* can support several virtual hosts and copies of 4D simultaneously. In this case, you specify the IP addresses and TCP ports of each copy of 4D.

21.2 Installation

The *4D Connect* plug-in is installed automatically by the *Kerio WebSTAR 5 Installer*. It is configured from the *4D Connect* panel in the *Web Server* group in *Admin Client*. This means that you must have full admin privileges.

For each virtual host that uses *4D Connect*, you set:

- The URL or IP address of the machine running the 4D database,
- TCP/IP Port used by the 4D database.

In the database, you set the TCP/IP port on the *Web Server Publishing* page of *Database Properties*. The *Publish Database at Startup* preference should also be turned on (unless you will enable database publishing at a later time) You can start web publishing from the User Environment by choosing *Web Server* → *Start Web Server* or with the START WEB SERVER command.

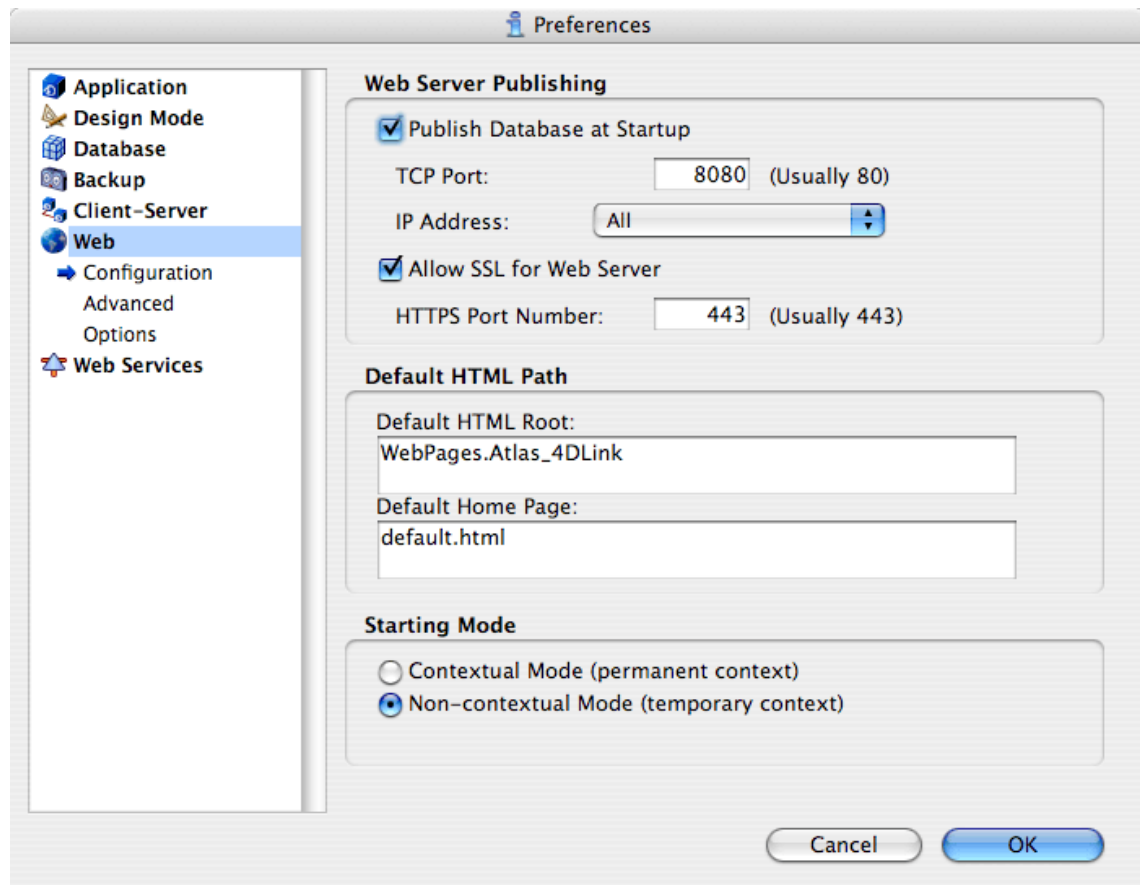


Figure 21.1 The Web Server Publishing page of Database Properties

Installation on one computer

If you want to use 4D and *Kerio WebSTAR* on the same computer, change the TCP port in this dialog to another value (i.e., 8080) and configure *4D Connect* accordingly. *Kerio WebSTAR* will continue to serve on the default port (80) and the plug-in will call 4D on the other port.

Supporting several virtual hosts

You can also configure *4D Connect* so that several virtual hosts and communicate with a different 4D database. The *Admin Client* application lets you configure *4D Connect* separately for each virtual host. If you are using *4D Connect* for several virtual hosts, simply create an entry for each virtual host.

21.3 Configuring 4D Connect

Once you have the 4D database (or databases) set up, you need to open the *Admin Client* application and set two configuration parameters: for each virtual host that uses *4D Connect*:

- The URL or IP address of the machine that is running the database,
- The TCP/IP Port the database uses (default is 80),

To configure *4D Connect*:

1. Highlight the *4D Connect* item in the *Web Server* group in the *Admin Client* application.

The following screen appears, listing all the virtual hosts that you have created.

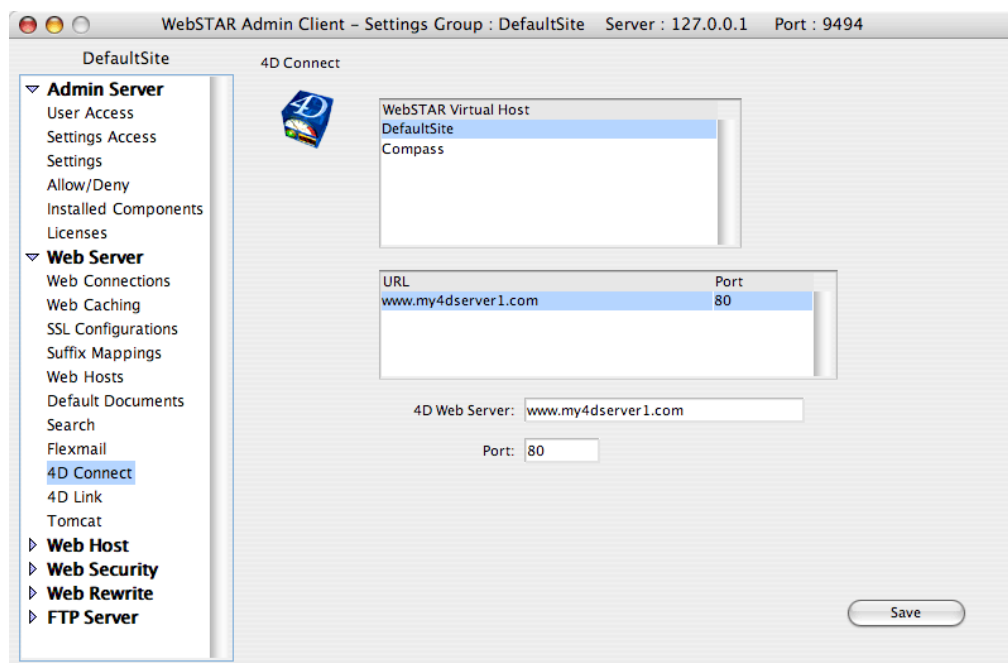


Figure 21.2 The 4D Connect panel in Admin Client

2. Highlight a virtual host that uses *4D Connect* in the *Virtual Hosts* table.
The URL table changes to show the configuration options for that virtual host.
3. Replace the default URL with the URL or IP address of the machine running the 4D database.
4. If necessary, modify the default port of 80.
This setting must match the TCP/IP port set in *Database Properties* in the database.
5. (Optional) If more than one virtual host will use *4D Connect*, repeat this process for each virtual host.
6. Click *Save* to save these specifications to the web server.
If necessary, add additional lines for additional hosts and instances of 4D.
7. (Optional) Open the *Database Properties* dialog box in 4D and verify that the TCP port you designated in the *4D Connect* panel in *Admin Client* agrees with the setting in 4D.
8. (Optional, if applicable) Open the *TCP/IP Control* panel on the relevant Mac OS classic computers and verify that TCP/IP is set to *Active* and the option to *Load only when needed* is not selected.

These options are available by clicking the *Options* button in the *TCP/IP Control* panel. If the *Options* button is not on your *TCP/IP Control* panel, choose *Edit → User mode* and select the *Advanced* setting.
9. Verify that 4D Web Services are active and running.

You are now ready to use *Kerio WebSTAR* to serve information in 4D databases.

21.4 Overview

4D Connect uses a template-based approach to dynamic web serving. When it sends a request for data to 4D, it also sends a template file. This is usually an incomplete HTML page that contains 4DVARs and/or 4DHTMLVARs. 4D parses the template, supplying the requested data by filling in the values of 4D tags on the template.

The general scenario is as follows:

- When the browser connects to *Kerio WebSTAR*, it serves the home HTML page. The home page usually contains one or more URLs to an incomplete page that 4D will use as a template. The template page must be a .4d text document. It is usually an HTML

file that contains 4D tags. However, it could be a text document that contains at least one 4D tag that tells 4D to return HTML, such as

```
<!--4DINCLUDE DefaultPage.html-->
```

In this case, 4D will replace this tag by `DefaultPage.html` that could contain a `4DSCRIPT` that will check the request and replace other 4D tags by specific values.

or

```
<!--4DSCRIPT/HandleConnection-->
```

```
<!--4DHTMLVAR WebHTMLPage-->
```

In this case, the `HandleConnection` method will check the request and initialize the `WebHTMLPage` variable with the correct HTML page to be displayed.

- The user then clicks on a URL that contains a reference to the template page.
- *Kerio WebSTAR* sends the request and the template to 4D.
- 4D receives the request and the template. The database must recognize, parse, and respond to the information contained in the request. If the 4D instance is not available or does not exist, an error message will be returned to the browser. The template will be parsed and all 4D Tags will be replaced by their current values. If references to 4D tags in the request or template are undefined, an error will be returned in the HTML page sent to *Kerio WebSTAR*.

If there are no errors, 4D sends the filled-out HTML template back to *Kerio WebSTAR*, which serves the page to the browser.

- The connection between 4D and *Kerio WebSTAR* will be terminated; the web process created on 4D will be closed.

You will generally use the `4DSCRIPT` tag in requests and 4D tags such as `4DVAR` and `4DHTMLVARs` in your templates to return information to the browser.

`4DSCRIPT` automatically executes the `On Web Authentication` database method and then the method passed by `4DSCRIPT`.

`4DSCRIPT` can pass additional parameters to the called method. It is your responsibility to recognize, parse, and respond to the additional information passed in these parameters.

A request received from *4D Connect* does not execute the `On Web Connection` method by default. A connection could be done and 4D tags in the template could be replaced by their values without running the `On Web Authentication` method. `On Web Authentication` runs when 4D encounters a `4DSCRIPT` tag. Each time that a 4D method is executed

from a 4DSCRIPT tag, the On Web Authentication method will run before the specified method. For example, if the template contains four 4DSCRIPT tags, On Web Authentication executes four times. This would happen if 4DSCRIPT is used in the template header and is also nested in three 4DVAR or 4DHTMLVAR tags in the templates. We suggest you use 4DSCRIPT only in the header of your template so that 4D can quickly parse the template document and respond to *WebSTAR*. There is no time-out between 4D and *Kerio WebSTAR* but there is between the browser and *Kerio WebSTAR*.

In compiled databases, declare C_TEXT(\$1;\$2;\$3;\$4;\$5;\$6) in the On Web Connection and On Web Authentication methods.

Declare C_TEXT (\$0;\$1) in the method that will be called by the 4DSCRIPT Tag.

Declare C_BOOLEAN(\$0) in the On Web Authentication method and set \$0:=True to accept the connection or False to refuse the connection. If \$0 is not defined, the connection is accepted by default.

When 4D receives the 4D tag, it is processed as described in the *Web Services* section of the 4D 6.7 *Language Reference*. See the *Language Reference* for complete information on 4D HTML tags.

21.5 4D Connect Examples

The *DefaultSite* virtual host includes worked examples of both *4D Link* and *4D Connect*. The *4D Database Integration* folder in the *Examples* folder contains the example databases that are described in this chapter and chapter 20. The folder *WebSTAR 4D Connect* contains the examples described here and the folder *Extra Examples* contains one database that illustrates both *4D Link* and *4D Connect* simultaneously. It is a modification of the *Atlas* example that includes the code from this chapter and chapter 20.

To try out the examples, set up a virtual host for each example. Follow the configuration instructions in the read me file for each database.

The following four examples illustrate this basic approach. In the supplied examples, it is assumed that only one instance of 4D is being used.

Example 1: Basic Communication

This example serves to illustrate the communication process between *Kerio WebSTAR* and 4D. The default page contains one hyperlink, which loads the template.

Installing the Example

To install the example, place *Page1.html* and *Page2.4d* in a virtual host folder that you have configured to use *4D Connect*. Create an alias to *Page1.html* and name it *default.html*. Launch *Kerio WebSTAR* 5.

On the machine with 4D, open the Example 1 database. *Database Properties* have already been set to *Publish Database at Startup*.

Running the Example

Open a Web browser and enter the URL of the *Kerio WebSTAR* machine. You will see the following screen:

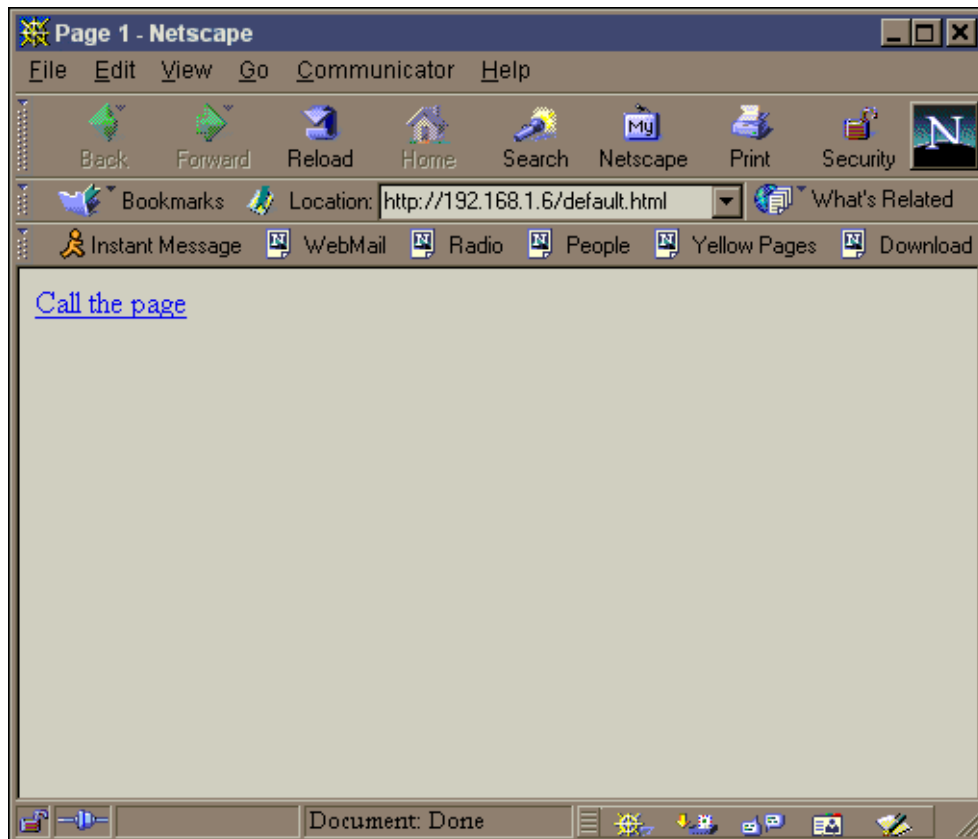


Figure 21.3 Running the Example

When you click the link, the following page appears:

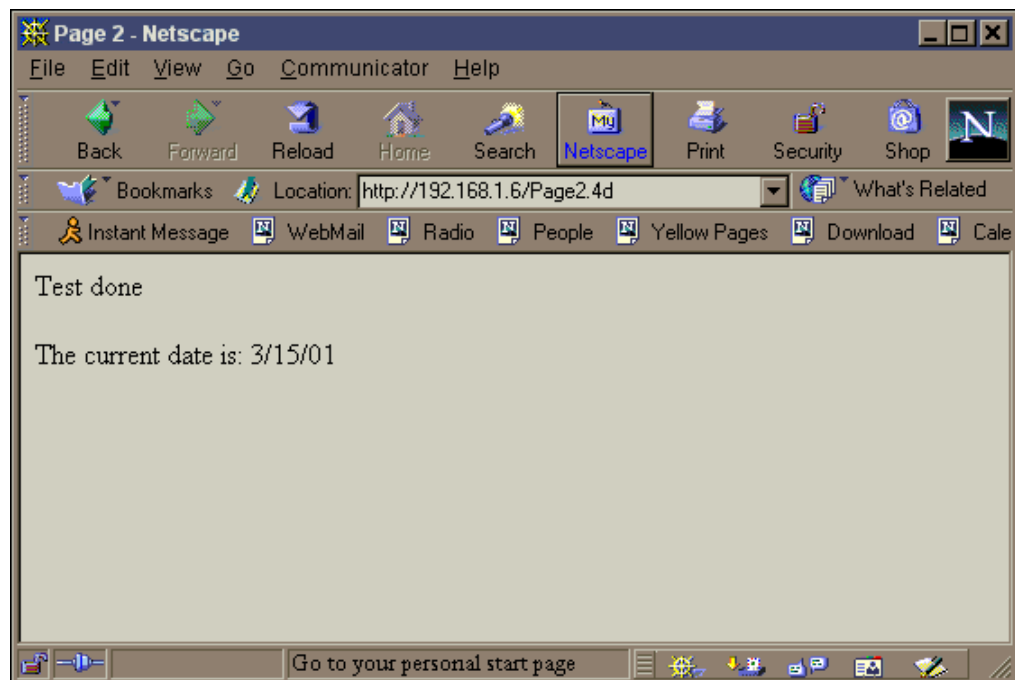


Figure 21.4 Result the Example

Behind the Scenes

Page1.html is very simple HTML that contains the one link to Page2.4d.

```
<html>

  <head>
    <meta http-equiv="content-type" content="text/html;
charset=iso-8859-1">
    <title>Page 1</title>
    <meta http-equiv="pragma" content="no-cache">
  </head>

  <body>
<a HREF="/Page2.4d">Call the page</a>
  </body>
</html>
```

The template page, Page2.4d, contains a 4DSCRIPT tag that calls My4DMethod, some static text, and two 4DVARs:

```
<html>

  <head>
    <meta http-equiv="content-type" content="text/html;
charset=iso-8859-1">
    <title>Page 2</title>
    <meta http-equiv="pragma" content="no-cache">
    <!--4DSCRIPT/My4DMethod-->
  </head>
  <body >
    <p>Test done.</p>

    <!--4DVAR MyVariable-->
    <!--4DVAR current date-->
  </body>
</html>
```

The first 4DVAR is a variable that is assigned a value by the database. The second 4DVAR is the 4D function Current date.

The 4DSCRIPT tag tells 4D to run the My4DMethod when it receives the request. 4D runs On Web Authentication prior to My4DMethod. My4DMethod assigns a value to the MyVariable 4DVAR. It is:

```
‘My4DMethod
```

```
C_TEXT($1;$0)
```

```
MyVar:="The current date is "
```

Just to let you know when On Web Authentication runs, it consists of the line:

```
BEEP
```

When you click the hyperlink, the `page2.4d` template page is sent to 4D with the request. 4D executes the method specified by the 4DSCRIPT command.

Thus, the returned page contains static text from `Page2.html`, the value of a 4D variable, and the value of a built-in 4D function.

Now, open the template document, `Page2.4d`, with a text editor such as *SimpleText* or *BEdit* and remove the line that calls the 4DSCRIPT tag.

It should now look like this:

```
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
charset=iso-8859-1">
    <title>Page 2</title>
    <meta http-equiv="pragma" content="no-cache">

  </head>

  <body >
<p>Test done</p>

<!--4DVAR MyVariable-->
<!--4DVAR current date-->
  </body>
</html>
```

Rerun the example by logging onto the *WebSTAR* machine again. When you click the *Call the Page* link, you will see a screen like this:

Since 4DACTION wasn't around to call `My4DMethod`, 4D was unable to substitute a value for `MyVariable`. However, it recognized `current date` as a 4D function and was able to return its value. Since a 4DVAR referenced in the template was not defined, an error message is returned to the browser. Also, you will not hear the beep on the 4D machine, indicating that On Web Authentication was not executed, even though information was returned by 4D.

Note: Using 4D functions in your HTML is not recommended because of localization issues. Although 4D allows you to use 4D functions in your HTML, you must be sure that the version of 4D uses the same language that is used in your HTML. This example requires a US/International version of 4D. If you hardcode the names of any built-in 4D functions in your HTML, you must update the names

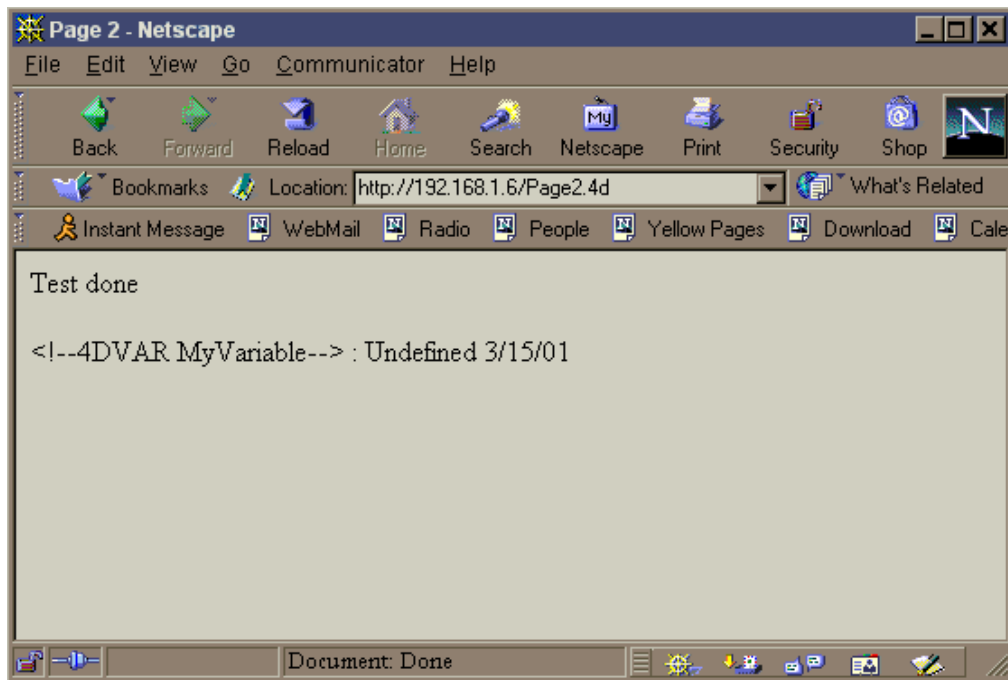


Figure 21.5 Result the Example

before opening the database in a foreign-language version of 4D. For example, if you need to provide a French version of this web site, the French version of 4D will not recognize the string current date as a 4D function. The string current date would be treated as a variable's name. To avoid the problem, set the value of the function to a variable within 4D, i.e., `MyVariable:="The current date is "+String(Current Date)`. If you know you will never need to localize the web site, there is no disadvantage to this technique.

Example 2: Basic Forms

This example receives user input via a simple form, does a calculation, and returns the results to the user.

Installing the Example

Copy the files `Page1.html` and `MyHTMLPage.4d` to a virtual host folder that is configured to use *4D Connect*. Create an alias to `Page1.html` and name it `default.html` and launch *Kerio WebSTAR*.

On the machine with 4D, open the Example 2 database. *Database Properties* have already been set to *Publish Database at Startup*.

Running the Example

When a browser logs onto the *WebSTAR* machine, the following screen appears:

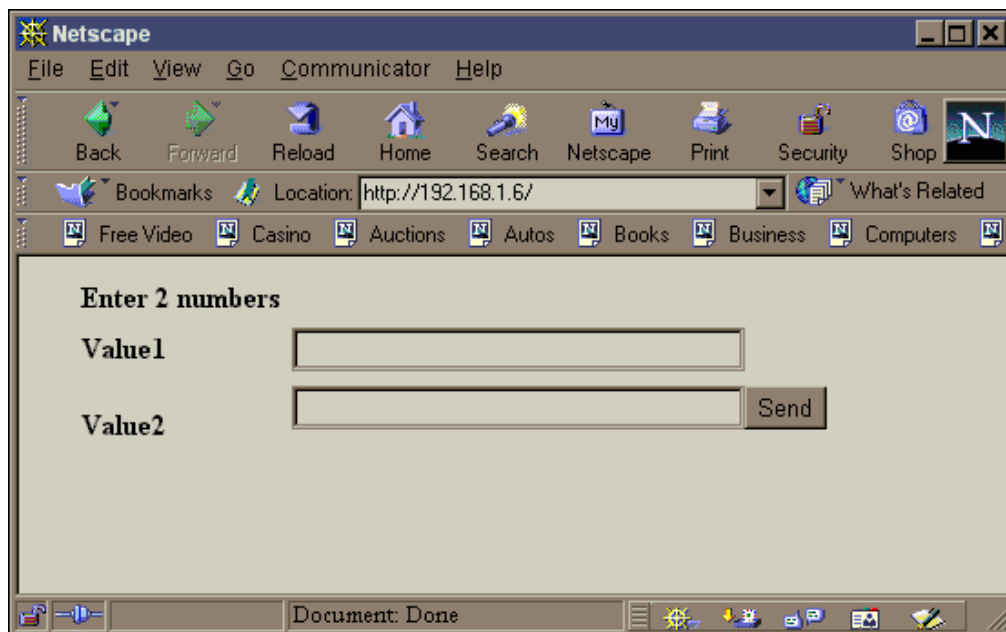


Figure 21.6 Running the Example

When the user enters two numbers and presses *Send*, the results appear in the browser along with the name of the HTML page that processed the request:

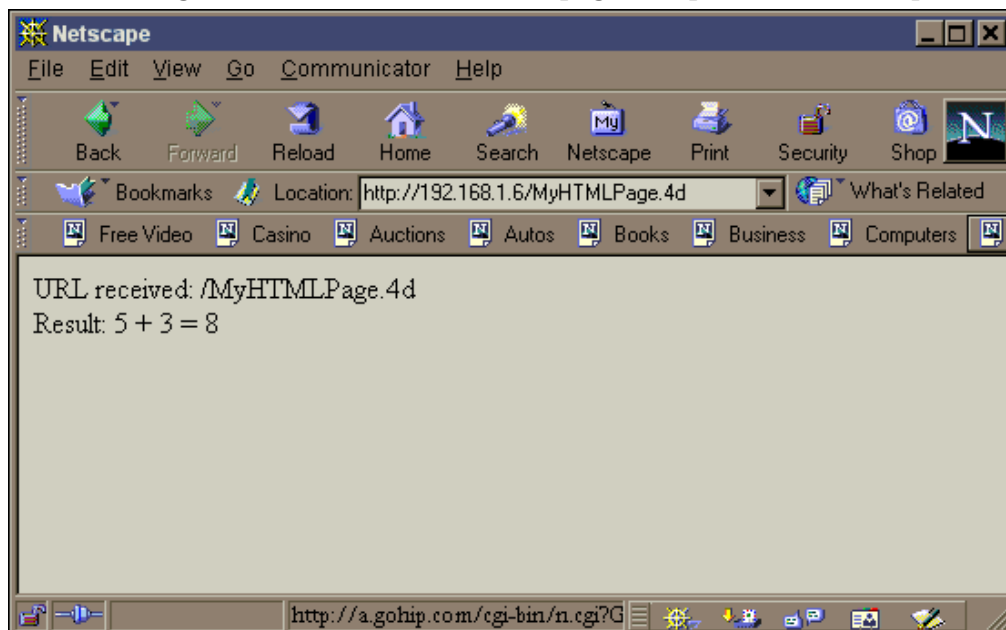


Figure 21.7 Result the Example

The text of the URL sent by *4D Connect* was extracted by the On Web Authentication database method and the calculation was done by the method called by 4DSCRIPT. The URL of the template page is shown in the URL entry area.

Behind the Scenes

The example was constructed in the following way:

The default page is as follows:

```
<html>

  <head>
    <meta http-equiv="content-type" content="text/html;
charset=iso-8859-1">
    <meta name="generator" content="Adobe GoLive 4">
    <meta http-equiv="pragma" content="no-cache">
  </head>

  <body>
    <table>
      <tr>
        <td width="15"></td>
        <td><b>Enter 2 numbers</b></font></td>
        <td></td>
      </tr>
      <form method="POST" action="/MyHTMLPage.4d">
        <tr>
          <td width="15"></td>
          <td><b>Value1</b></td>
          <td><input type="text" name="Value1" size="30"></td>
        </tr>
        <tr>
          <td width="15"></td>
          <td><b>Value2</b></td>
          <td><input type="text" name="Value2" size="30"><input
type="submit" name="Submit" value="Send"></form></td>
        </tr>
      </table>
    </body>
  </html>
```

The template page is `MyHTMLPage.4d`. It contains only one 4D variable, `My4DResult`, and the 4DSCRIPT specifies that the `HandleWebSTARConnection` method will run

when 4D receives the page:

```
<HTML>
<HEAD>
  <meta http-equiv="pragma" content="no-cache"/>
  <!--4DSCRIPT/HandleWebSTARConnection-->
</HEAD>
<BODY>
  <!--4DHTMLVAR My4DResult-->
</BODY>
</HTML>
```

The On Web Authentication database method is executed prior to HandleWebSTAR-Connection and extracts the URL sent to it by *4D Connect*:

```
C_TEXT($1;$2;$3;$4;$5;$6)
C_BOOLEAN($0)
ARRAY TEXT(Array_Header_Names;0)
ARRAY TEXT(Array_Header_Values;0)
GET HTTP HEADER(Array_Header_Names;Array_Header_Values)
URL_received:=Array_Header_Values{2}
```

```
$0:=True
```

The HandleWebSTARConnection method does the calculation and fills in the value of the 4DHTMLVAR:

```
‘ HandleWebStarConnection
C_TEXT(My4DResult;Value1;Value2)
C_TEXT($1;$0)
C_REAL($Compute)
$Compute:=Num(Value1)+Num(Value2)
My4DResult:="URL received: "+URL_Received+"<BR>"
My4DResult:=My4DResult+"Result: "+Value1+" + "+Value2+" =
"+String($Compute)
```

After the 4DSCRIPT has been executed in the template, the page will continue to be parsed. When 4D reaches the 4DHTMLVAR tag it will be replaced by the contents of the My4DResult variable. Then, this parsed and modified document will be sent to *WebSTAR*, which will serve it to the browser.

Example 3: 4D HTML Tags

Example 3 illustrates the use of the 4DINCLUDE, 4DIF, and 4DLOOP tags.

Installing the Example

Copy the contents of the folder ToCopyonWebSTAR to the *WebSTAR* computer. It contains one folder, WebPages.4DTag_4DConnect. Move it to a virtual host folder that is configured to use 4D Connect. It contains a single folder, Pages, that contains the template file, 4DConnect.4d, and a folder called Images that contains a background graphic. The default page is named 4DTag_4DConnect.html. Create an alias to it, rename the alias default.html, and move it to a virtual host that has been configured to use *4D Connect*.

The computer running 4D uses the file HTML2.txt and the folder *WebPages.4DTag_4DConnect*, which contains one file, HTML3.html. This file and folder must be in the same folder as the database.

Running the Example

When a browser connects to the *WebSTAR* machine, the following link appears:

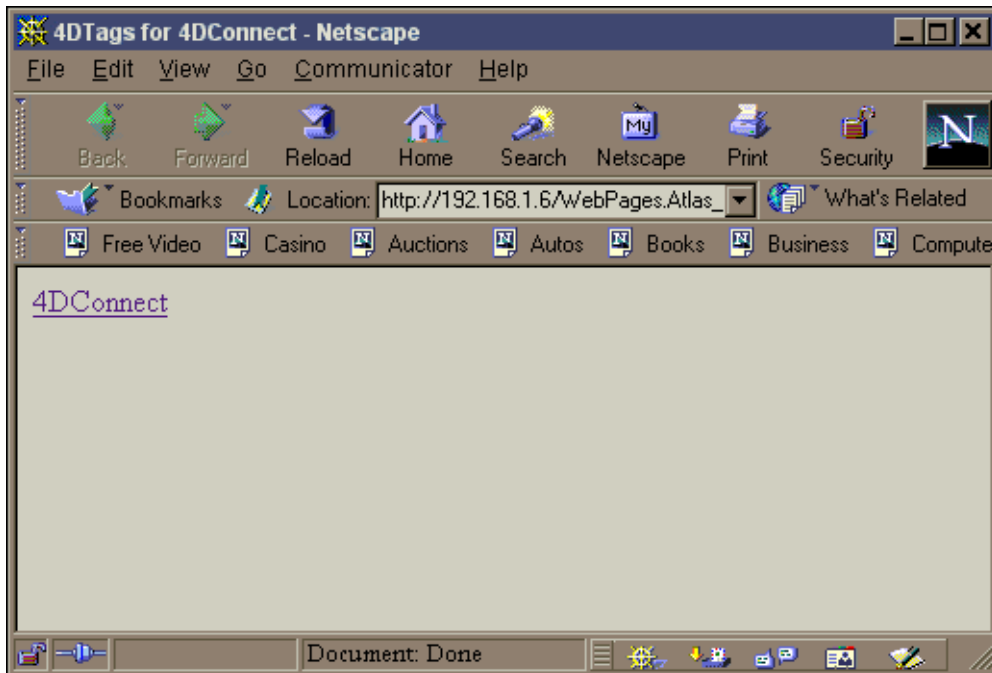


Figure 21.8 Running the Example

When the user clicks the link, the following results are displayed:

Behind the Scenes

The default.html page contains one link which calls the template page:

```
<HTML>
<HEAD>
  <meta http-equiv="pragma" content="no-cache"/>
```

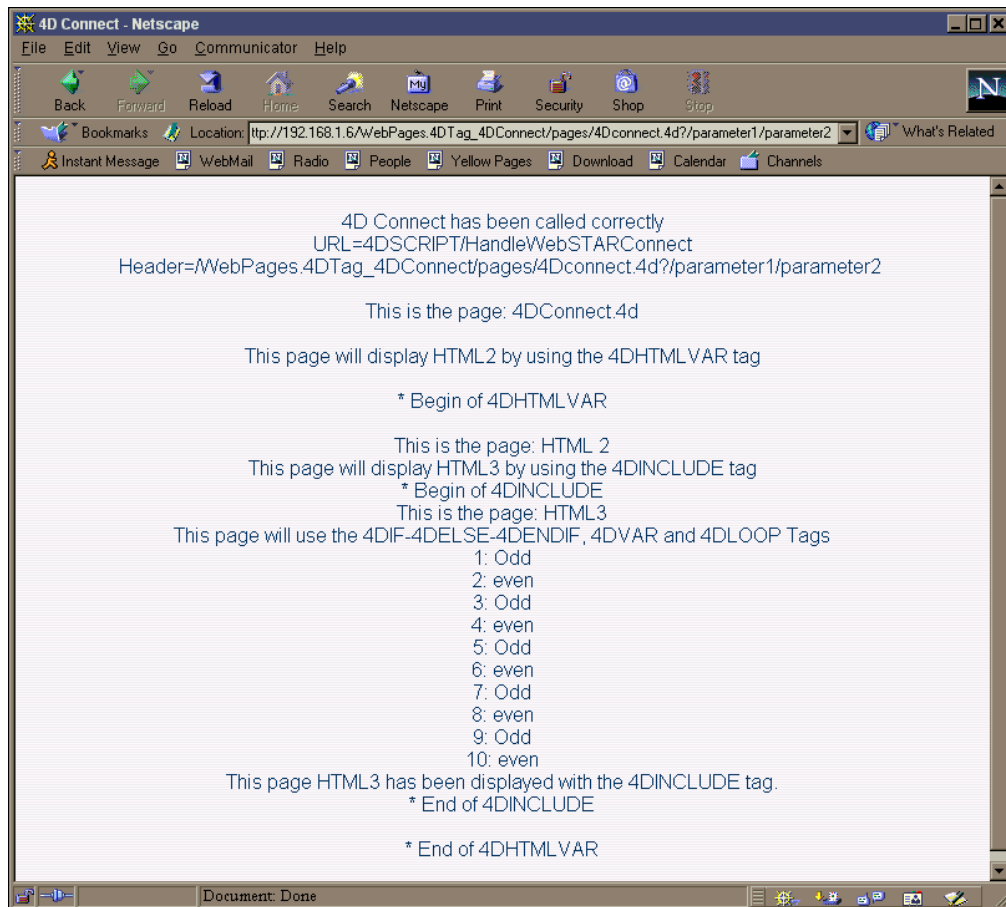


Figure 21.9 Result the Example

```
<TITLE>4DTags for 4DConnect</TITLE>
</HEAD>
<BODY>
  <A
    HREF="/WebPages.4DTag_4DConnect/pages/4Dconnect.4d?/pa
    rameter1/parameter2">4DConnect</A>
</BODY>
</HTML>
```

The template page, 4DConnect.4d is the following:

```
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=iso-8859-1">
    <title>4D Connect</title>
    <!--4DSCRIPT/HandleWebSTARConnection-->
```

```

    <meta http-equiv="pragma" content="no-cache">
</head>
<body
background="/WebPages.4DTag_4DConnect/pages/Images/
background.gif
" bgcolor="white">
    <center>
        <br>
        <font color="#003366"
face="Arial,Helvetica,Geneva,Swiss,SunSans-Regular">
<tr><td>4D Connect has been called correctly</td></tr><br>
<tr><td>URL=<!--4DVAR URL--></td></tr><br><tr><td> Header=<!--
4DVAR vheader--></td></tr>
<p>This is the page: 4DConnect.4d</p>
<p>This page will display HTML2 using the 4DHTMLVAR tag</p>

<p>* Begin of 4DHTMLVAR</p>
<!--4DHTMLVAR MyVarBodyHTML1-->
<p>* End of 4DHTMLVAR</p>

</font></center>
</body>

</html>

```

It contains two 4DVARs, URL and vHeader, and the 4DHTMLVAR variable MyVarBodyHTML1. When the request is sent to the database, the On Web Authentication method runs automatically and fills in the value of the URL:

```

'On Web Authentication
C_TEXT($1;$2;$3;$4;$5;$6)
$0:=True

```

```
URL:=$1 ' 4DVAR URL in 4DConnect.4d
```

The values of the 4DVARs vHeader and MyVarBodyHTML1 are set in the HandleWebSTARConnection method, which is called by the 4DSCRIPT tag in the template:

```

' HandleWebSTARConnection
' 4D Connect has been called
' The method defined in 4DSCRIPT of the 4DConnect.4d
' will be executed

```

```
GET HTTP HEADER(tHeader;tValueHeader)
```

```
C_TEXT($v)
vHeader:=tValueHeader{2} ' 4DVAR vHeader in 4DConnect.4d

MyVarBodyHTML1:=""
Loadtext ("HTML2.txt";->MyVarBodyHTML1)
ARRAY INTEGER(arr_Integer;10)

' Array used in HTML3.html
For ($i;1;10)
Arr_Integer{$i}:=$i
End for
The HTML2.txt document contains a call to HTML3.html:

This is the page: HTML 2<br>
This page will display HTML3 by using the 4DINCLUDE tag<br>
* Begin of 4DINCLUDE<br>
<!--4DINCLUDE HTML3.HTML-->
* End of 4DINCLUDE<br>
The HTML3.html page demonstrates the use of the 4DLOOP and
4DIF...4DELSE...4DENDIF tags. The If statement determines whether the ith
element in Arr_Integer is even or odd and then returns the value of the ith element
and either the word Even or Odd.

<html>
<head>
  <meta http-equiv="content-type" content="text/html;
charset=ISO-8859-1">
  <meta name="generator" content="Adobe GoLive 5">
  <title>Welcome to Adobe GoLive 5</title>
</head>
<body bgcolor="#ffffff">
  This is the page: HTML3<br>
  This page will use the 4DIF-4DELSE-4DENDIF, 4DVAR and 4DLOOP
Tags<br>
  <!--4DLOOP Arr_Integer-->
  <!--4DIF Mod(Arr_Integer{Arr_Integer};2)=0--><!--4DVAR
Arr_Integer{Arr_Integer}-->: even<br><!--4DELSE--><!--4DVAR
Arr_Integer{Arr_Integer}-->: Odd<br><!--4DENDIF-->
  <!--4DENDLOOP-->
  This page HTML3 has been displayed with the 4DINCLUDE
tag.<br>
```

```
</body>
</html>
```

Thus, the page that is returned includes static text from `4DConnect.4d` and `HTML2.html`, plus the output from `HTML3.html`.

Example 4: 4D Atlas

This is a more realistic example that supports user queries from the browser, extracts the requested data from the database, and serves the requested data via *Kerio WebSTAR*.

Installing the Example

The *WebSTAR* machine uses the *WebPages.Atlas_4DConnect* folder that is in the *ToCopyonWebStar* folder. Move *WebPages.Atlas_4DConnect* to the same level as the *WebSTAR* application. It contains a single folder, *Pages*. The default page is located in this folder. Create an alias to `default.html` and move the alias to the same level as *WebSTAR* and rename it `default.html`.

The 4D machine uses a different *WebPages.Atlas_4DConnect* folder. It contains a single folder, *Pages* that contains the following files:

- `WebPages.Atlas_4DConnect\Pages\SearchCountry.txt`
- `WebPages.Atlas_4DConnect\Pages\NoRecords.txt`
- `WebPages.Atlas_4DConnect\Pages\InvalidContext.txt`
- `WebPages.Atlas_4DConnect\Pages>EditCountry.txt`
- `WebPages.Atlas_4DConnect\Pages\HeaderHTML.txt`
- `WebPages.Atlas_4DConnect\Pages\FooterHTML.txt`
- `WebPages.Atlas_4DConnect\Pages\ListRecordsCountry.txt`
- `WebPages.Atlas_4DConnect\Pages\ListRecords.txt`

Move this *WebPages.Atlas_4DConnect* folder to the same level as the database structure.

Running the Example

When a browser logs onto the *WebSTAR* server, the following screen appears:

The `default.html` page draws the graphics directly. The key lines in the `default.html` page are:

```
href="WebPages/Pages/request.4d?4DNC&WEB_InitSearch&1"
```

and

```
href="WebPages/Pages/request.4d?4DNC&WEB_AllRecords&1&A"
```

for the *Find a Country* and *List All Countries* links, respectively. Each request is done using a URL that specifies the `request.4d` template page and several parameters which follow the question mark.

The `request.4d` template uses three 4DHTMLVARs for the header, body, and footer of the page:

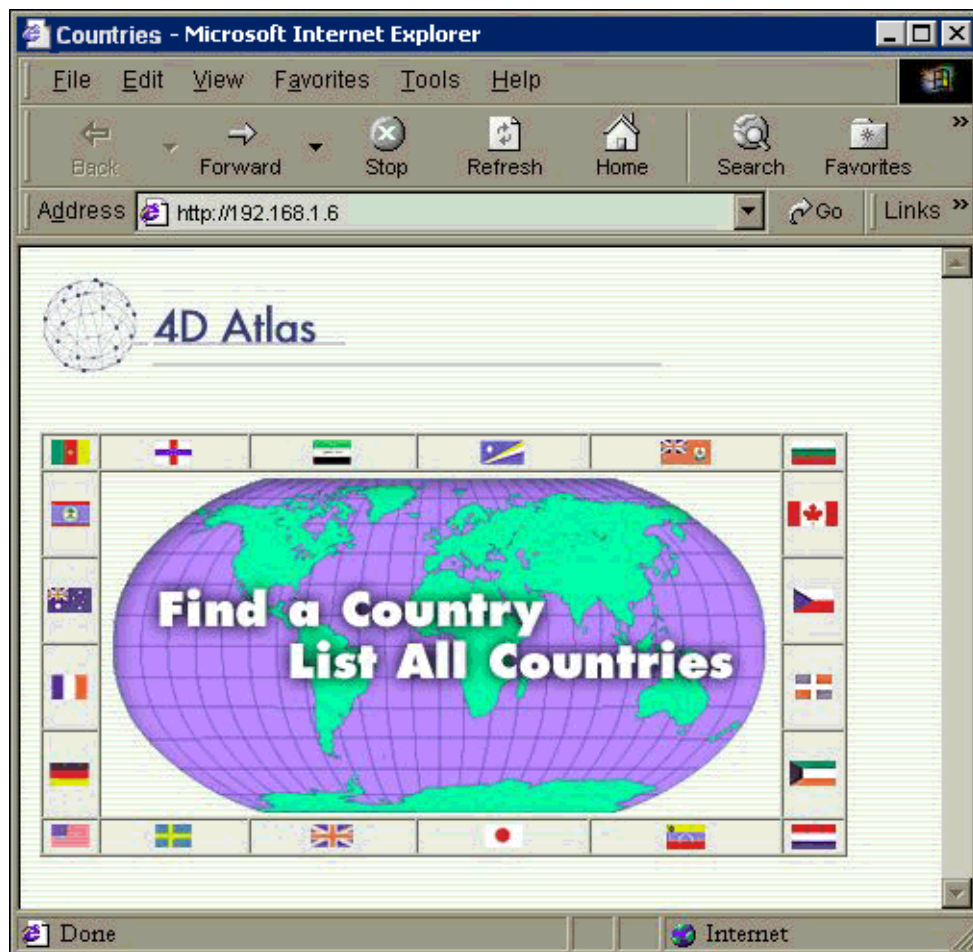


Figure 21.10 Running the Example

```
<html>
<head>
  <meta http-equiv="content-type"
content="text/html; charset=iso-8859-1">
  <meta http-equiv="pragma" content="no-cache">
  <!--4DSCRIPT/HandleWebSTARConnection-->
  <title><!--4DVAR WebTitle--></title>
</head>

  <body bgcolor="white" link="#003399" alink="red" vlink=
"#ff00ff" background="images/background.gif">
  <!--4DHTMLVAR MyVarHeaderHTML-->
  <p>
  <!--4DHTMLVAR MyVarBodyHTML1-->
```

```

<p>
<!--4DHTMLVAR MyVarFooterHTML-->
</body>
</html>

```

The On Web Authentication method runs before HandleWebSTARConnection and calls Connection_is_4DConnect. It extracts the ContextID, the method to call, and the table number from the URL. Optional parameters that follow the table number parameter are parsed here.

```

' On Web Authentication
C_TEXT($1;$2;$3;$4;$5;$6)
$0:=True ' Accept the connection
WebTitle:=""
Connection_is_4DConnect

```

For a new connection, the ContextID is set to 4DNC. For a new connection, Connection_is_4DConnect assigns the ContextID and keeps it in memory for further requests. The ContextID is used to manage lengthy selections of records that must be returned to the browser in successive subsets.

The problem is that, after a page is returned, the connection is killed and the selection is lost. The ContextID allows 4D to keep the selection on the web server and allows the same browser to retrieve successive groups of records from that selection. Each time a browser logs in with a particular ContextID, it will get its selection.

With 4D Connect, you cannot use the IP address of the browser as the ContextID since all requests come from the *WebSTAR* machine. In this database, the ContextID is built from random numbers in the Connection_is_4DConnect method. Each time that web browser connects to the site, that browser's ContextID is provided in the request. This enables 4D to use the selection associated with that browser.

The following code in Connection_is_4DConnect manages the ContextID:

```

If (ContextID="4DNC")
  'It's a new connection from a browser
  ContextID:=String(Random)+String(Random)
  ' Define a semaphore to add element into this array
  INTERN_SetSemaphore ("Arrays")
  IndiceContext:=Size of array(TContextes)+1
  INSERT ELEMENT(TContextes;IndiceContext;1)
  TContextes{IndiceContext}:=ContextID
  INSERT ELEMENT(TContextSets;IndiceContext;1)
  INTERN_ReleaseSemaphore ("Arrays")
Else

```

```
IndiceContext:=Find in array(TContextes;ContextID)
End if
```

In WEB_SearchRecords, a 4D set is given a temporary name using the Random function. This name is stored in the DebileName variable.

The name of this selection is then stored in an array (TContextSets) with the same ID of the contextID (TContextes).

```
' Name of the selection of the current connection
DebileName:="<>S"+String(Random)+String(Random)
INTERN_SetSemaphore ("Arrays")
$asd:=Size of array(TContextSets{IndiceContext})+1
' Will keep it until next query
INSERT ELEMENT(TContextSets{IndiceContext};$asd;1)
TContextSets{IndiceContext}{$asd}:=DebileName
INTERN_ReleaseSemaphore ("Arrays")
```

Once the selection has been created, a set is created from this selection. When a connection is performed, Connection_is_4DConnect checks the index of the contextID in the TContextes array. If this ContextID exists, this gives us the index of the set to be used. Then, a USE SET is executed to retrieve this selection. Saving and retrieving the current selection is done in WEB_SearchRecords.

The HandleWebSTARConnection routine executes the method requested in the URL, i.e., either WEB_AllRecords or WEB_InitSearch.

```
'HandleWebSTARConnection
' Method called by 4D Connect
C_TEXT($0;$1)

' The request.4d file contains 4DHTML variables
C_TEXT(MyVarHeaderHTML;MyVarFooterHTML;MyVarBodyHTML1;
MyVarBodyHTML3)

MyVarFooterHTML:=""
MyVarBodyHTML3:=""
```

```
If (IndiceContext>0)
Case of
: (MethodToCall="WEB_InitSearch") 'A search has been
requested WEB_InitSearch
: (MethodToCall="WEB_SearchRecords") ' A search has been
validated WEB_SearchRecords
: (MethodToCall="WEB_ShowRecord") ' A record has been
```

```

requested WEB_ShowRecord
: (MethodToCall="WEB_NewOrder") ' A selection of records
by name has been requested
WEB_NewOrder
: (MethodToCall="WEB_AllRecords") ' All records requested
WEB_AllRecords
End case
Else
' The context ID does not exist
For ($i;1;Count tables)
  REDUCE SELECTION(Table($I)->;0)
End for
  INTERN_Fill_VarHTML ("WebPages.Atlas_4DConnect"+SepDir+
"Pages"+SepDir+"HeaderHTML.txt";->MyVarHeaderHTML)
  INTERN_Fill_VarHTML ("WebPages.Atlas_4DConnect"+SepDir+
"Pages"+SepDir+"InvalidContext.txt";->MyVarBodyHTML1)
End if

```

If a header will be returned, the 4D method will fill the Header variable. The method will also fill the body variables. If a footer will be returned, the 4D method will fill the Footer variable.

Each request begins with a string of the form:

```
request.4d?4DNC&MethodToBeExecuted&TableNumber&Extra parameters
```

Subsequent requests will be of the form:

```
request.4d?ContextID&MethodToBeExecuted2&
```

```
TableNumber&Extraparameter1&Extra parameter2
```

While executing the request, 4D will check if the ContextID is correct. If it is, it will execute the method passed as the second parameter.

Using the extra parameters, the method will determine which HTML page 4D has to return. In this example, the method will fill the HTML variables defined in the *request.4d* page, allowing you to display the requested information, an error message, another HTML page, or whatever is appropriate.

List All Countries

If the user clicks the *List All Countries* link, the URL calls the WEB_AllRecords routine which sets the *Country* search field to @ and then calls WEB_SearchRecords: After finding all the records and loading it into a set, it loads the names and IDs of the countries into the arrays atNameDisp and atIDDisp, respectively. It then fills in the 4DHTMLVARs specified in the template page using the following lines of code:

```
INTERN_Fill_VarHTML ("WebPages.Atlas_4DConnect"+SepDir+
"Pages"+SepDir+"ListRecords.txt";->MyVarBodyHTML1)
INTERN_Fill_VarHTML ("WebPages.Atlas_4DConnect"+SepDir+
"Pages"+SepDir+"ListRecordsCountry.txt";->MyVarBodyHTML3)
INTERN_Fill_VarHTML ("WebPages.Atlas_4DConnect"+SepDir+
"Pages"+SepDir+"FooterHTML.txt";->MyVarFooterHTML)
```

Each of these lines specifies a text document in the *WebPages.Atlas_4DConnect* folder on the 4D machine. The *ListRecords.txt* document is used to print the header information:

```
<td align="left" valign="top"><font
face="Arial,Helvetica,Geneva,Swiss,SunSans-Regular" size="2"
color="#003366"><b><br>
    </b><b>Records Shown <!--4DVAR LStartSel-->
    of <!--4DVAR LRecordsInSelection-->
    Record<!--4DIF (LRecordsInSelection>1)-->
    s <!--4DENDIF-->
    Found.</b></font>
<p></td>
```

The *ListRecordsCountry.txt* document is used to print the list itself using a 4DLOOP tag:

```
<!--4DLOOP atNameDisp-->
<TR>

<TD>
<A HREF="request.4d?<!--4DVAR ContextID-->
&WEB_ShowRecord&<!--4DVAR WEBTable-->&<!--4DVAR
atIDDisp{atNameDisp}-->">
    <!--4DVAR atNameDisp{atNameDisp}--></A><P>
</TD>
</TR>
<!--4DENDLOOP-->
```

The *FooterHTML.txt* document is used to print the alphabetical TOC at the bottom of the screen:

```
<!--4DLOOP atLetterOrder-->
<td>
    <A HREF="request.4d?<!--4DVAR ContextID-->
&WEB_NewOrder&<!--4DVAR WebTable-->&<!--4DVAR DebileName-->
&<!--4DVAR
atLetterOrder{atLetterOrder}-->"><!--4DVAR
```

```
atLetterOrder{atLetterOrder}--></A>
```

```
</td>
```

```
<!--4DENDLOOP-->
```

The following illustration shows the contents of each 4DHTMLVAR on the template page.

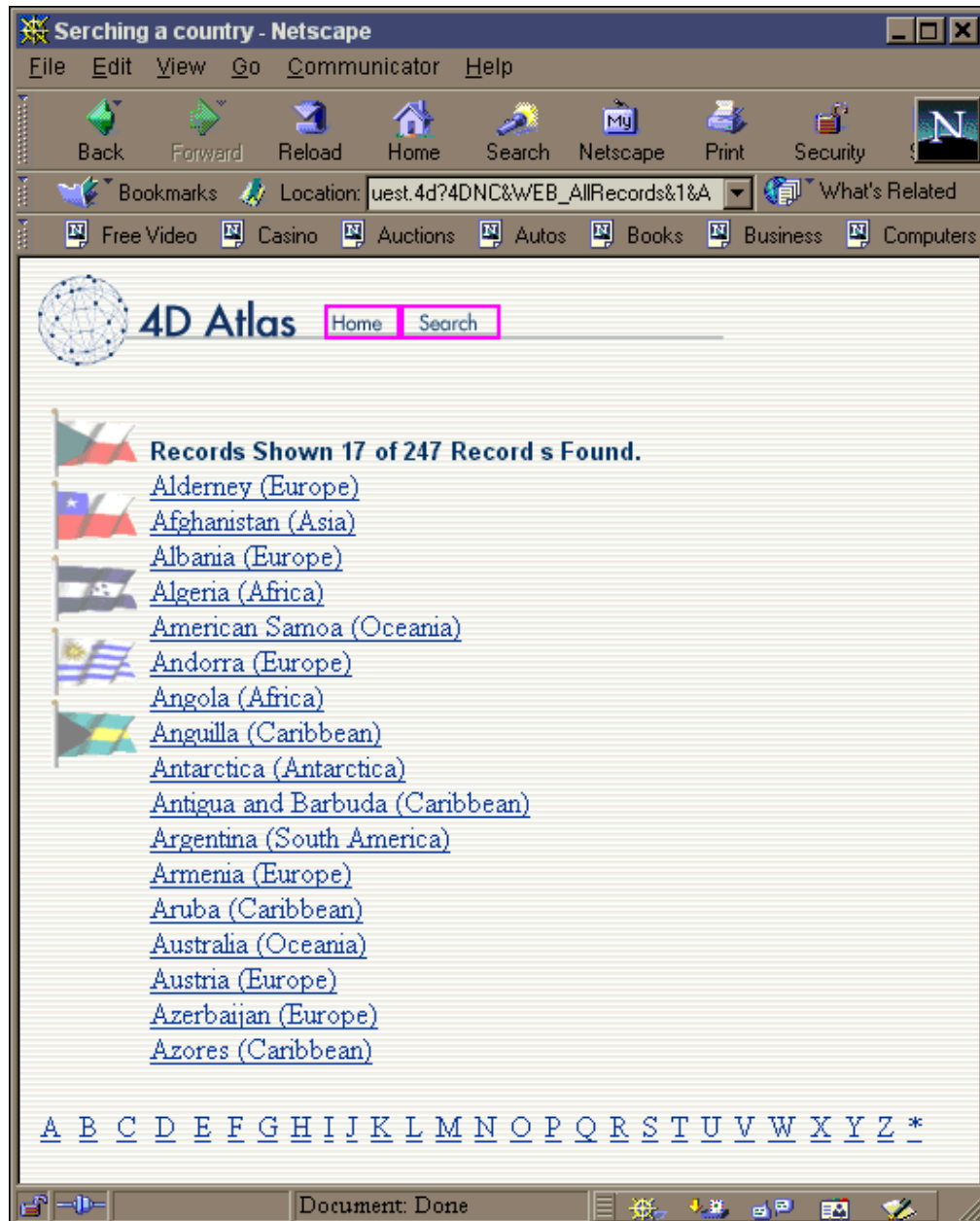


Figure 21.11 List All Countries

Searching for a Country

If the user clicks the *Find a Country* link, the URL directs HandleWebStarConnection to execute WEB_InitSearch, which uses the SearchCountry.txt file to create the search dialog.

```
‘ WebInitSearch
```

```
Case of
```

```
: (WebTable=1) ‘ Will display the search dialog
```

```
INTERN_Fill_VarHTML ("WebPages.Atlas_4DConnect"+SepDir+  
"Pages"+SepDir+"SearchCountry.txt";->MyVarBodyHTML1)
```

```
WebTitle:="Searching a country"
```

```
End case
```

```
‘ Display the banner
```

```
INTERN_Fill_VarHTML ("WebPages.Atlas_4DConnect"+SepDir+  
"Pages"+SepDir+"HeaderHTML.txt";->MyVarHeaderHTML)
```

This sends the following search screen to the browser:

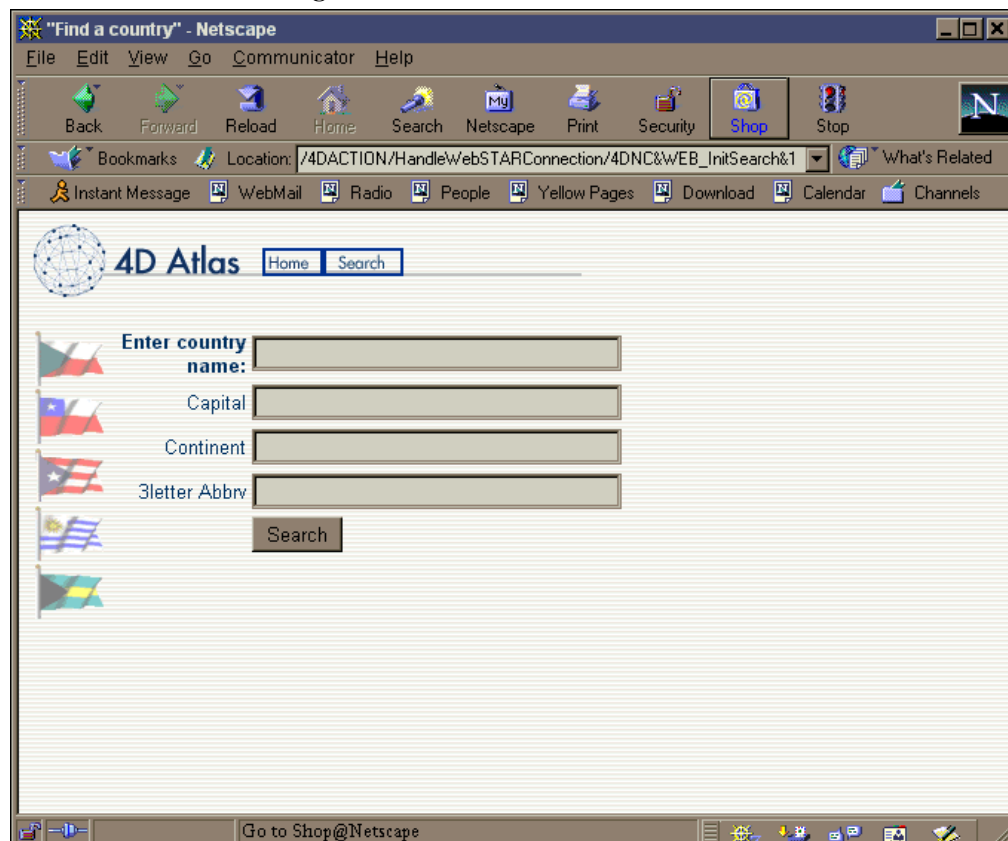


Figure 21.12 Searching for a Country

When the user enters one or more values into the fields and clicks *Search*, the following line in SearchCountry.txt calls WEB_SearchRecords:

```
action="request.4d?<!--4DVAR ContextID-->
&WEB_SearchRecords&<!--
4DVAR WEBTable-->&" >
```

WEB_SearchRecords executes the requested query and returns the requested records via the same code used for displaying all countries.

For example, if the search requests one country, a screen such as this appears:

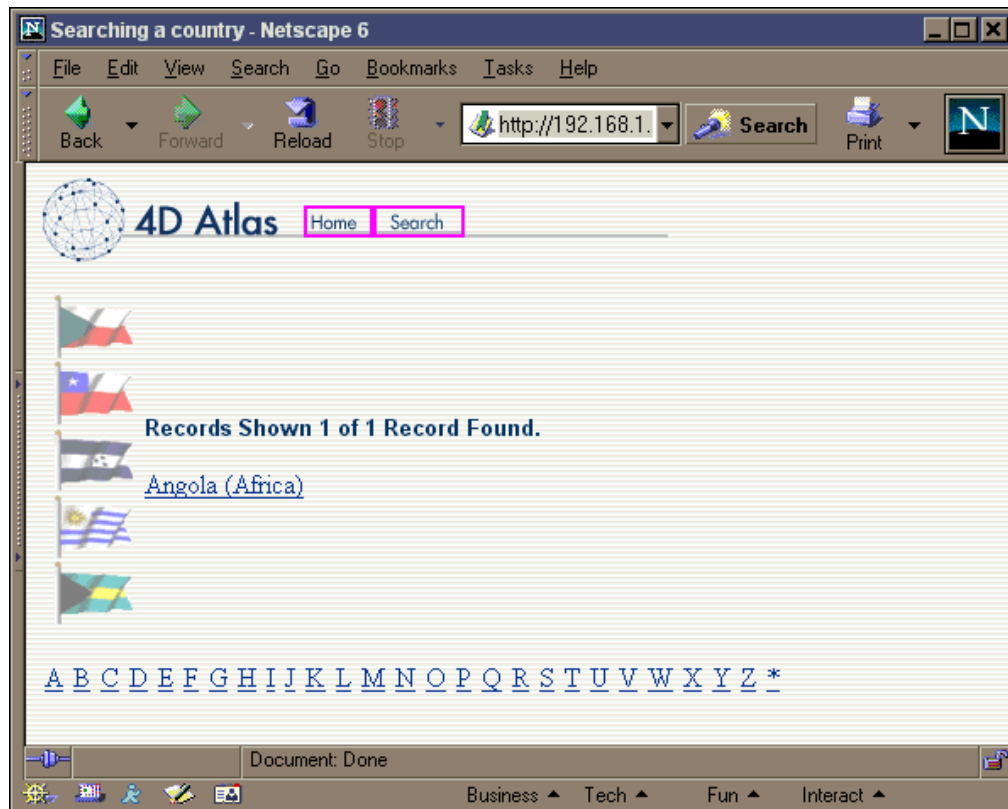


Figure 21.13 Searching for a Country — result

If the user clicks a country, WEB_ShowRecord executes, which uses the EditCountry.txt document.

```
' WEB_ShowRecord}
pTable:=Table(WebTable)
Case of
: (WebTable=1)
WebTitle:="Displaying a country"

QUERY([Countries];[Countries]ID_Country=Num(TVariables{1}))
```

```

If (Record number([Countries])=0)
  INTERN_Fill_VarHTML ("WebPages.Atlas_4DConnect"+SepDir+
  "Pages"+SepDir+"Edit_Record_Not_Found.txt";->MyVarBodyHTML1)
  Imagette:=[Countries]Flag
Else
  INTERN_Fill_VarHTML ("WebPages.Atlas_4DConnect"+SepDir+
  "Pages"+SepDir+"EditCountry.txt";->MyVarBodyHTML1)
End if
End case
' Display the header
INTERN_Fill_VarHTML ("WebPages.Atlas_4DConnect"+SepDir+
  "Pages"+SepDir+"HeaderHTML.txt";->MyVarHeaderHTML)

```

This presents a screen such as the following:

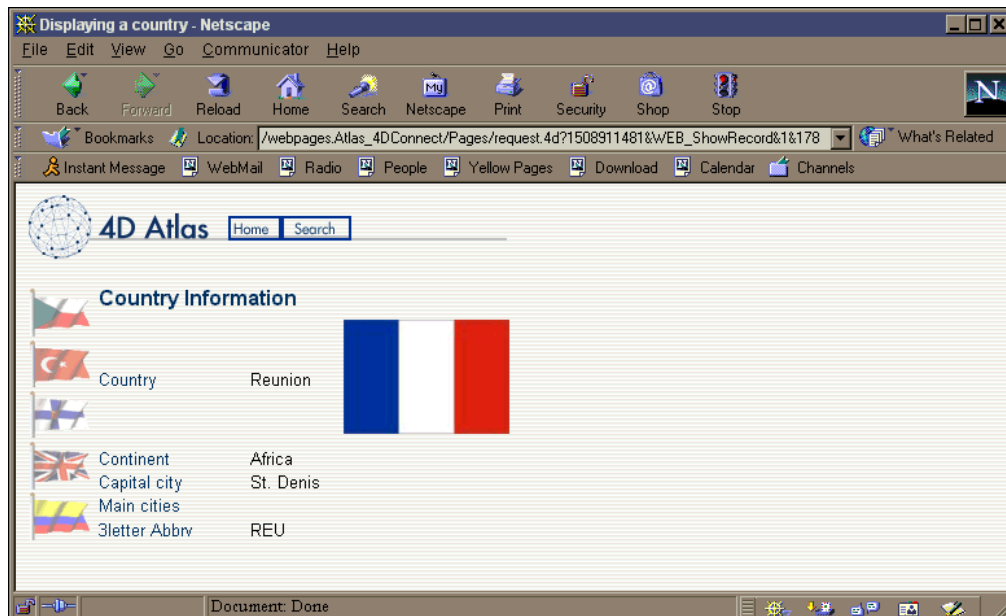


Figure 21.14 Searching for a Country — result uses the EditCountry.txt document

The HeaderHTML.txt contains links to redisplay the default.html page or the search screen. Either process starts a new connection with a new ContextID.

21.6 Troubleshooting

If the browser can connect to *Kerio WebSTAR* but cannot connect to 4D via *4D Connect*, check your TCP/IP configuration. Be sure that the *Load only when needed* checkbox is not checked in the *TCP/IP Control* panels on the 4D server machine (running classic Mac OS). To check this point, do the following:

1. From the *TCP/IP Control* panel, click the *Options* button in the bottom-right corner

of the dialog. If this button is not in the dialog, choose *Edit* → *User Mode* from the *Edit* menu and select the *Advanced* mode.

The *Options* dialog box is shown below:

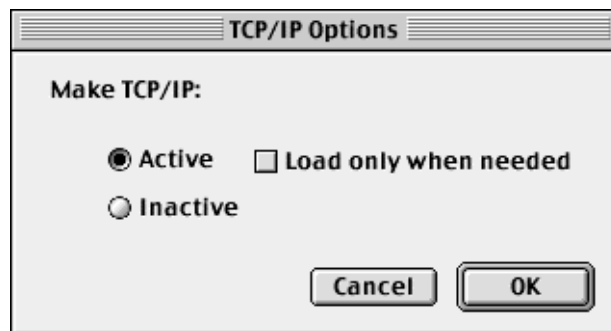


Figure 21.15 The TCP/IP Options dialog box

It must be configured as shown here: *Active* must be selected and *Load only when needed* must be unchecked.

- Be sure that the IP address and port you specified in the *4D Connect.* panel in *Admin Client* matches the settings in the *Database Properties* dialog in the copy of 4D that you are trying to access. Double-check that you have configured *4D Connect* for the correct virtual host.
- Be sure that 4D's web server is running. You can start it automatically by selecting the *Publish Database at Startup* preference in *Database Properties*.

You can also start the web server from the User environment by choosing *Web Server* → *Start Web Server*. This menu item is disabled if the web server is running.

- Be sure that *4D Connect* was initialized when you launched *Kerio WebSTAR*. Check the *WebSTAR Status* window to verify that *4D Connect* appears in the list of initialized plug-ins.

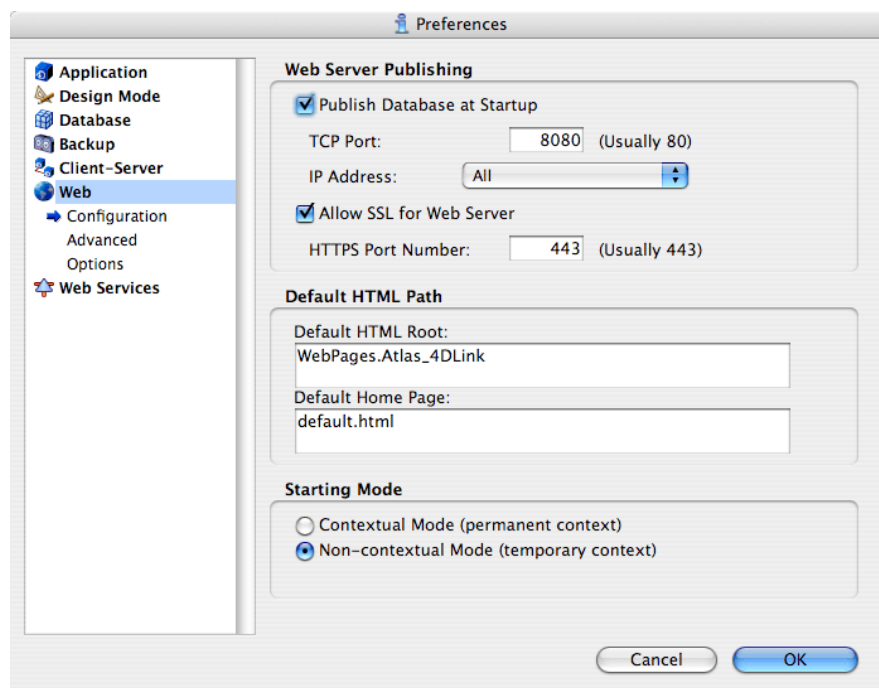


Figure 21.16 The Database Properties

Chapter 22

Domain Name Servers

To link host names and IP addresses, the Internet uses the Domain Name System (DNS) protocol. DNS servers keep databases of these name-address combinations and work together to keep the system up to date. The Domain Name System works as the Internet's phone book. When you visit a web site, DNS translates the domain name in the URL (into a numeric IP address).

Your ISP or network administrator is probably running a DNS server. To enter your *Kerio WebSTAR* server into this system, you should request that your ISP or the network administrator make sure your server host name or names and IP address is in their DNS database. They will broadcast this information to other DNS servers across the Internet, and everyone will be able to find you shortly.

Many ISPs offer user-friendly ways of supporting users who want to host their own web or mail servers that eliminate the need for you to deal with records in the DNS database directly. You may need to only fill out an online questionnaire or send an email with information on your domain names. Contact customer support for your ISP for more information.

On the other hand, you may want to run your own DNS server. In that case, you will be responsible for configuring it so that mail and http requests are routed properly.

For technical information, see RFC 1034. The most authoritative book on DNS is *DNS and Bind*, published by O'Reilly.

22.1 A Name Records

The A (Address) record is the vital link between the host name and the IP address. All Internet host machines must have an A name record somewhere in the DNS system.

The A name record contains the host name, class IN (Internet), the letter A, and the IP address of the host machine. Here are some examples:

domain.com	IN	A	192.0.1.1
mail.domain.com	IN	A	192.0.1.2
www.domain.com	IN	A	192.0.1.1

Table 22.1 A Name Records

In the example above, the `domain.com` and `www.domain.com` host names both point at the same IP address, while the `mail.domain.com` host name points at another address. To distribute your server requests among machines (if your DNS server performs load sharing), you can direct one domain name to several IP addresses.

22.2 CNAME Records

The CNAME (Canonical Name) record is an alias, much like a Macintosh alias file: it points to another host name. The DNS administrator may use CNAME records to define virtual host names.

A CNAME record is just an alias, so don't try to add any more entries to that host name record, but put the other information in the A Name records instead. They are generally used in the case where you might be pointing to a host on another network that has an A record in another DNS system, for example, if your ftp server was actually located on a different network and some other DNS administrator was apt to change the A record for that host without notice.

The CNAME name record contains the host name which is the alias, class IN (Internet), CNAME, and the real host name for the alias to point at. For example, this entry:

ftp.domain.com	IN	CNAME	www.domain.com
----------------	----	-------	----------------

Table 22.2 CNAME Records

directs all requests for the host name `ftp.domain.com` to the machine that hosts `www.domain.com`.

22.3 MX Records

MX (Mail Exchange) records specify whether the host is running an email server or messages for this host should go to another mail server machine.

22.4 Secondary DNS Services

If your DNS server goes offline for any reason, links to your site will fail and email will bounce. Therefore, the domain registries generally require you to register at least one alternate DNS server. Preferably, the secondary server should be located on another network (so local problems will not affect them) and have all current information. Your ISP or institutional DNS administrator can help you to update this information.

22.5 Checking Your DNS Entries

To make sure that your DNS entries are correct, you should check them periodically. The *Men & Mice* company provides a form at the following URL:

<http://us.mirror.menandmice.com/cgi-bin/DoDig>

22.6 Running Your Own DNS Server

If you are independent or have a cooperative network administrator, you can run your own DNS server. This allows you to add Virtual Domains quickly, and to alias several IP Addresses to one domain name, for load balancing, mirroring, and other purposes. There are several Mac OS DNS servers available.

Chapter 23

Specifying Pathnames

When you specify pathnames in *WebSTAR Admin Client*, you always use Unix-style path syntax — even though the application is running on a Macintosh. For example, when you create virtual hosts, you need to specify the folder that contains the files for each virtual host and the virtual host's error folder.

Although you may have never used a Unix computer, chances are good that you have been using Unix-style syntax as URLs in web browsers for years. Unix uses the slash (/) rather than the colon as the delimiter in pathnames.

The / all by itself is used to specify the root, or top, level of the file hierarchy. For example, the path /Applications/Utilities/ProcessViewer is the full path (or absolute path) to your *Process Viewer* application that you can use to verify that your server is running. The initial slash specifies the root, and each subsequent slash moves you down one level in the hierarchy until you get to the file itself.

When the current directory is below the root, you use the dot to specify another directory's position relative to the current directory.

Use the slash to move down to a lower level relative to the current directory. Thus ./DefaultSite is a relative path from the server to the folder that contains the default virtual host.

Use two dots to move up one level from the current directory. This moves you to the parent of the current directory. This syntax, however, is not needed in *WebSTAR Admin Client*.

Index

A

- 3DES [176, 194](#)
- 4D Connect [120, 259](#)
 - configuring [261](#)
 - installation [259](#)
 - overview of dynamic web service [262](#)
 - requirements [259](#)
 - troubleshooting [286](#)
- 4D HTML tags [234](#)
- 4D Link [120, 228, 234](#)
 - configuring [230](#)
 - overview [232](#)
 - requirements [228](#)
 - troubleshooting [257](#)
- 4D WebSTAR 4.x
 - update certificate [199](#)
- absolute path [292](#)
- access control [197](#)
 - firewall [198](#)
 - limit FTP access [197](#)
 - remove FTP access [197](#)
 - server monitor application [197](#)
- Action
 - 4D Connect Action [217](#)
 - 4D Link [217](#)
 - ACGI [216](#)
 - BINARY [216](#)
 - CGI [216](#)
 - MAPPY [217](#)
 - RAW [217](#)
 - SEARCH [217](#)
 - TEXT [216](#)
- Admin Client [7](#)
 - Admin Server menu [35](#)
 - File menu [34](#)
 - import and export [34](#)
 - Interface [20](#)
 - menu bar [34](#)
 - overview [16](#)
 - pop-up menus [30, 31](#)
 - reordering table rows [33](#)
 - sorting tables [33](#)
 - tables [28](#)
 - Tools menu [35](#)
 - View menu [35](#)
 - Windows menu [35](#)
- admin user [37](#)
- Allow/Deny panel [65](#)
- Allow/Deny table [66, 128](#)
 - controlling access [41](#)
- anonymous FTP [134](#)
- authenticator [61](#)

B

- Built-In User List [60](#)
- Built-in User Lists [36](#)

C

- cache
 - auto flush cache [114, 111](#)
 - cache size [113](#)
 - data cache [112](#)
 - flushing the cache [114](#)
 - manual flush cache [114](#)
 - updating caches [114](#)
- Certificate
 - creation [179](#)
- certificate
 - downloading [189](#)

Certificate
 formats [178](#)
 generate a key [179](#)
 installation [178](#)
certificate
 obtaining a test certificate [190, 191](#)
Certificate Authority
 Thawte [186](#)
 VeriSign [184](#)
CGI plug-in [13](#)
ciphers
 encryption [193](#)
Console application [13](#)
.css file [206](#)

D

DefaultSite folder [9](#)
Directory Indexer [202](#)
distinguished name [187](#)
DNS
 CNAME records [290, 289](#)
 MX records [290](#)
 name records [289](#)
 RFC 1034 [289](#)
documentation [8](#)
Domain Name System [289](#)

F

Flexmail
 configuration [148, 148](#)
 form creation [150](#)
 forms [149](#)
 MIME type [155](#)
 required fields [151](#)
 sending messages [153, 154](#)
.flx file [150](#)
_footer file [211](#)
FTP
 ASCII [141](#)
 binary [141](#)
 file security [140](#)

 folder security [140](#)
 MacBinary [141](#)
 monitoring FTP use [141](#)
 multiple hosts [141](#)
 password security [140](#)
 ports [142](#)
 security [140](#)
 specification [142](#)
 troubleshooting [142](#)
 URLs [141](#)

G

Goliath [70](#)

H

_header file [211](#)
host admin user [40](#)
HTTP [174](#)
HTTPS [174](#)

I

image mapping [171](#)
Image Map Plug-In [171](#)

K

Kerio WebSTAR Admin Client [7](#)
Kerio WebSTAR Admin Client [7](#)
Kerio WebSTAR Admin Server [7](#)
Kerio WebSTAR Launcher [7, 10](#)
Kerio WebSTAR Monitor [7](#)
Kerio WebSTAR Rewrite
 comparison types [87](#)
 conditional rules [84](#)
 interface [83, 78](#)
 modes [79](#)
 order of execution [82](#)
 rules creation [90](#)
 subject parameters [86](#)

L

localization

- configuration [226](#)
- documents [226](#)
- installation [224](#), [224](#)

M

_message file [206](#), [211](#)

MIME type [217](#)

Monitor [7](#)

P

Permissions [63](#)

Permissions List [60](#), [62](#)

Permission Table Configuration [63](#)

Process Viewer [13](#)

Process Viewer application [13](#)

R

realm

- creation [204](#)
- setting up [66](#)

regular expressions

- alternation operator [99](#)
- character classes [93](#)
- definition [92](#)
- greediness [98](#), [92](#)
- repetition chracters [96](#)
- submatches [100](#)
- subpatterns [100](#)
- wildcards [93](#)

relative path [292](#)

requirements

- client machine requirements [9](#)
- server machine requirements [8](#)

Rewrite [78](#)

rewrite modes

- modify [79](#)
- proxy [81](#)
- redirect [81](#)

routing table [55](#)

S

search

- boolean operators [164](#)
- crawler [159](#)
- DefaultSite [160](#)
- forms [164](#)
- grouping [164](#)
- queries [163](#), [118](#), [156](#)
- templates [168](#)
- troubleshooting [170](#)
- variables [169](#)

server-wide settings [103](#)

server certificate

- installation [192](#)

SSL

- 3DES [175](#)
- access [175](#)
- certificate authority [177](#)
- certificate chains [176](#)
- configuration [177](#)
- configuration panel [115](#)
- DES [175](#)
- DES-40 [175](#)
- encryption levels [174](#)
- IDEA-128 [175](#)
- intermediate certificates [176](#)
- MAC-only [175](#)
- protocol [174](#)
- RC4-40 [175](#)
- RC4-128 [175](#), [174](#)
- troubleshooting [199](#)
- wildcard certificates [177](#)
- SSL Handshake failed [200](#)
- Suffix Mapping
 - defaults [222](#)
 - MIME type [217](#), [214](#)

T

TCP/IP Control panel [286](#)

TLS [174](#)

Index

Tomcat

configuration [145](#)

quick start [147](#), [145](#)

U

User List Configuration [60](#)

V

virtual host

configuration [51](#)

creation [50](#), [50](#)

virtual host-specific settings [121](#)

W

WebDAV

client applications [70](#)

setting up [71](#), [70](#)

WebServer [7](#)

