Serving WebObjects

Before you can get your own WebObjects applications running, there are some system administrative tasks you should perform. If you encounter problems, this document will help you troubleshoot them. The following sections discuss the details of WebObjects system administration

- Installing and Deploying WebObjects concerns getting WebObjects, installing it on your Web server host, and installing WebObjects applications on hosts across your network.
- Checking Platform Dependencies lists specific features of Windows NT, Solaris, and NEXTSTEP that affect WebObjects installation and performance.
- Understanding WebObjects Files presents the workhorse of WebObjects, the WebObjects Adaptor, along with the **WebObjects.conf** configuration file, the **DefaultApp** script driver, and a specialized WebObjects Adaptor that integrates with the Netscape Commerce Server.
- Tuning, Testing, and Debugging shows you how to manually start a WebObjects application, load-balance multiple WebObjects applications on multiple host computers, inspect the **CGIMessenger.log** file, and follow a step-by-step test plan.

Installing and Deploying WebObjects

It's easy to get WebObjects, and it's easy to set up WebObjects applications distributed across your network.

- Get WebObjects for free from NeXT Software, Inc.
- Set up WebObjects on your Web server host.
- Deploy WebObjects applications on your Web server host and across your network.

Getting WebObjects

To get WebObjects

To get information, documentation, or the latest version of WebObjects for free, go to http://www.next.com/WebObjects

To get WebObjectsPro and WebObjectsEnterprise

WebObjectsPro adds the ability to run compiled applications and customize the WebObjects interface with your Web server.

WebObjectsEnterprise adds the ability to access a variety of relational databases.

To learn more about these more powerful members of the WebObjects family, go to http://www.next.com/WebObjects and choose Products.

You can telephone call 1-800-TRY-NEXT (outside the U.S. call 1-415-599-5239).

You can also email trynext@next.com.

To get more information about WebObjects Products

To keep up to date with changes, check WebObjects Release Notes by going to http://www.next.com/Pubs/Documents/WebObjects/ReleaseNotes.html

Setting up WebObjects

Once you have installed WebObjects on your system, you must set it up to work with your Web server. To do this, you need to know where to find the various WebObjects files and directories and what to do with them.

Note: If you have installed an upgrade version of WebObjects, see Installing a WebObjects upgrade. .

The <NeXT_Root> directory

WebObjects resides under the <*NeXT_Root*>**/NextLibrary** directory.

- On NEXTSTEP, the <*NeXT_Root*> directory is the root directory of the startup disk.
- On Solaris, the <*NeXT_Root*> directory is the /usr directory.
- On Windows NT, you use the WebObjects SETUP utility to specify a <*NeXT_Root*> directory. By default, the SETUP utility creates the **C:\NeXT** directory for <*NeXT_Root*>. See Windows NT platform dependencies for more information.

Remember to check NeXT Software's WebObjects Release Notes regularly for updated information.

The WebObjects directory

The WebObjects directory resides under the <*NeXT_Root*>**/NextLibrary** directory. It contains a **ReadMe.html** file with installation instructions as well as these directories

- <*NeXT_Root*>**/NextLibrary/WebObjects/cgi-bin** contains the WebObject Adaptor
- <*NeXT_Root*>**/NextLibrary/WebObjects/Examples** contains a ReadMe file and example WebObject programs
- <*NeXT_Root*>**/NextLibrary/WebObjects/Executables** contains the **DefaultApp** program

On NEXTSTEP or Solaris you may have an additional WebObjects directory to which the live WebObjects directory links. For example

- <*NeXT_Root*>/NextLibrary/WebObjects-Beta
- <*NeXT_Root*>/NextLibrary/WebObjects-C

This scheme lets you store several versions of WebObjects. You link whichever you choose to the live WebObjects directory. By default, the latest version of WebObjects is linked to the WebObjects directory.

Server-specific directories

On UNIX-based systems, whether or not you received a Web server from NeXT, WebObjects provides a WebServer directory which resides under the <*NeXT_Root*>**/NextLibrary** directory. It contains two directories

- <*NeXT_Root*>**/NextLibrary/WebServer/cgi-bin** contains the WebObjects Adaptor.
- <*NeXT_Root*>**/NextLibrary/WebServer/htdocs/WebObjects** contains an Examples directory that is a link to the examples provided under the <*NeXT_Root*>**/NextLibrary/WebObjects** directory.

If you received a Web server from NeXT Software, Inc., then this Web server resides under the <*NeXT_Root*>**/NextLibrary/WebServer** directory, and these links have been made for you.

On Windows NT systems, WebObjects' setup installation utility asks you to enter information about the location of your <*NeXT_Root*> directory as well as your Web server's *cgi-bin* and *htdocs* directories. The Setup program copies the appropriate files to these locations. See the description of Windows NT platform dependencies for more information.

On NEXTSTEP or Solaris systems, you must create links from the corresponding directories of your Web server.

- Link the <*NeXT_Root*>**/NextLibrary/WebServer/cgi-bin/WebObjects** executable (the WebObjects Adaptor) into your Web server's *cgi-bin* directory.
- Link the <*NeXT_Root*>**/NextLibrary/WebServer/htdocs/WebObjects** folder into your Web server's *DocumentRoot* directory.

Note: Check the documentation for your Web server to confirm the locations of its *cgi-bin* and *DocumentRoot* directories.

Installing a WebObjects upgrade

If you're installing an upgraded version of WebObjects, be sure to clean up the following

- Make sure that you do not have any old **/tmp/WebObjects.*** files left.
- On NEXTSTEP and Solaris systems, verify the time-stamp of the WebObjects file in your Web server's *cgi-bin* directory.

  If you are using your own Web server and not the supplied one, do not forget to remove any old WebObjects file from the HTTP server's *cgi-bin* directory.

  Add in the Web server's *cgi-bin* directory a link to WebObjects
  ```
  ln -s /NextLibrary/WebObjects/Executables/cgi-bin/WebObjects
  WebObjects
  ```
- On Windows NT, you may need to rename your **WebObjects**, **DefaultApp**, and any compiled WebObjects applications with a **.exe** extension, depending on the Web server you are using.

WebObjects Application Deployment

A WebObjects application may reside on the same computer as the Web server or on a remote computer.

To install a WebObjects application on the computer that hosts your Web server, WebObjects applications must reside under the Web server's *DocumentRoot***/WebObjects** directory.

To install a WebObjects application on a remote computer, be sure that computer has the <*NeXT_Root*>**/NextLibrary/WebObjects** directory and its subdirectories, then locate your WebObjects application anywhere under the **WebObjects** directory.

WebObject applications may be scripted or compiled. To start up WebObjects applications

- Start up a scripted application with the command `./DefaultApp WebObjectsAppName`
- Start up a compiled application with the command `./WebObjectsAppName`

The URL that requests a WebObject application is identical in either case.

The WebObjects Adaptor invokes **DefaultApp** to run a scripted WebObjects Application.

The WebObjects Adaptor will autostart either a scripted or a compiled application only on the computer that hosts the Web server.

The WebObjects Adaptor assumes that all instances of WebObjects applications are running. You should create an **rc** script that activates WebObjects applications at boot time. Also you should create a shell script that lets you start up WebObjects applications from the command line.

If the WebObjects Adaptor receives a URL requesting a WebObjects application that is not currently running, the WebObjects Adaptor will try to autostart that WebObjects application.

Note: On Windows NT, determine whether your Web server requires the **.exe** extension for your compiled WebObjects application name. If so, and if your WebObjects applications are misnamed, then the WebObjects Adaptor may skip your intended compiled WebObjects applications and continue its autostarting routine with unpredictable results.

Checking Platform Dependencies

WebObjects works with any HTTP server on any of the following platforms. Note administration differences

- Windows NT doesn't allow links across a filesystem, has no **ps** command, and uses the TEMP environment variable, among other characteristics.
- UNIX systems include Solaris and NEXTSTEP.

Windows NT platform notes

See WebObjects Release Notes for updated information on WebObjects running on Windows NT systems.

Windows NT presents some important differences from UNIX-based operating systems such as

- WebObjects' Setup installation utility has no counterpart in UNIX.
- Windows NT uses a backslash path delimiter. Note that some Web servers require the UNIX-style forward slash in building URLS.
- The DOS command shell requires that programs have the .EXE extension, but some Web servers require no .EXE extension.
- You must continually reopen the log file, as Windows NT has no counterpart to the **tail -f** command.
- Application resources must reside directly under the directory for the application, rather than in subdirectories thereunder.

• A Windows NT-based Web server won't access WebObjects applications on UNIX platforms.

Using WebObjects' Setup Installation utility

If you're using Windows NT, you should use WebObjects' setup installation utility and enter information about the location of your <*NeXT_Root*> directory as well as your Web server's *cgi-bin* and *DocumentRoot* directories. The setup program will copy the appropriate files rather than make links. By default, the setup utility creates the **C:\NeXT** directory for <*NeXT_Root*>.

During installation, the WebObjects setup installation program asks you to specify a name for the software group on NT. Don't use parenthesis characters in the name, or the setup installation program will crash. Either use the default group suggested by the installer, or, if you need to specify another group name, use alphanumeric characters only.

Creating URLs on Windows NT

In building URLs, use forward slashes as opposed to a backslashes.

The Web server uses the **WebObjects.conf** ApplicationDirectory argument to build URLs, so use forward slashes there, too.

With or without the .EXE extension

NetScape HTTP servers on NT do not accept CGI programs with an .EXE extension. For the CGI WebObjects Adaptor to work with a NetScape server, the CGI WebObjects Adaptor can't have any extension. (Other NT HTTP servers may require the .EXE extension; if so, add the .EXE extension to the WebObjects program).

DefaultApp can't be run from the DOS shell. Windows NT expects executables to have the .EXE extension when they are executed from the DOS shell. The workaround is to copy **DefaultApp** to **DefaultApp.exe** before you execute it in a DOS shell.

Inspecting the CGIMessenger file

In order to enable debugging you should create the **logWebObjects** file in the directory specified by the TEMP environment variable. Use Notepad or a similar text editor to view the resulting **CGIMessenger.log** file. Remember that the file is changed during each request, so you'll need to continually reopen it to monitor adaptor activity.

Application resource location

Resources cannot be located in nested directories. Keep resources immediately underneath the application directory. For instance, the DodgeLite example keeps its database dictionary in
*DocumentRoot***/WebObjects/Examples/DodgeLite/DodgeData.dict**
instead of in
*DocumentRoot***/WebObjects/Examples/DodgeLite/Database/DodgeData.dict**
as it would on a UNIX-based system.

Mixed Platforms

Load balancing doesn't work with a Web server running on Windows NT and the WebObject applications running on UNIX platforms. Avoid hybrid configurations.

UNIX platforms

Webobjects runs on Solaris and NEXTSTEP UNIX systems.

Solaris platform notes:

- Under Solaris, the <*NeXT_Root*> directory is the **/usr** directory. After you've downloaded WebObjects, you'll find WebObjects resides under the **/usr/NextLibrary** directory.

- Using the **ps** command.

  To check which scripted WebObjects applications are running on a Solaris system use
  ```
  ps -eaf | grep DefaultApp
  ```

  To check which compiled WebObjects applications are running on a Solaris system use
  ```
  ps -eaf | grep <ApplicationName>
  ```

NEXTSTEP platform notes

- Under NEXTSTEP, the <*NeXT_Root*> directory is the root (**/**) directory. After you've downloaded WebObjects, you'll find WebObjects resides under the **/NextLibrary** directory.

- Using the **ps** command.

  To check which scripted WebObjects applications are running on a NEXTSTEP system use
  ```
  ps -aux | grep DefaultApp
  ```

  To check which compiled WebObjects applications are running on a NEXTSTEP system use
  ```
  ps -aux | grep <ApplicationName>
  ```

Understanding WebObjects Files

The behavior of your WebObjects application depends on its interaction with your Web server and some WebObjects intermediary files, which include the WebObjects Adaptor, the **DefaultApp** program, and the **WebObjects.conf** configuration file.

- The WebObjects Adaptor is a CGI-based software interface between a Web server and your WebObjects applications that allows WebObjects applications to be Web server independent.

- **DefaultApp** provides an interface between the WebObjects Adaptor and scripted WebObjects applications.

- The **WebObjects.conf** configuration file is a text file that you create in your Web server's *cgi-bin* directory that lets you institute load-balanced distribution of WebObjects applications across your network.

- The Netscape Interface Adaptor, based on the Netscape API, can be integrated with the Netscape Commerce Server. This is part of the WebObjects Pro and WebObjects Enterprise products.

The WebObjects Adaptor

The WebObjects Adaptor provides a flexible interface between a variety of HTTP servers and WebObjects applications. The WebObjects Adaptor consists of two parts, a Web server-side part and a WebObjects application part.

The application part consists of special classes that are part of every WebObjects application.

The server-side part is a stand-alone CGI program named **WebObjects**. It should be installed under your Web server's *cgi-bin* directory.

Note: The WebObjects Adaptor must reside on the NEXTSTEP, Solaris, or Windows NT computer that hosts your Web server.

Note: On Windows NT, some Web servers (such as Microsoft's IIS server) require that you rename the WebObject Adaptor to **WebObjects.exe**.

The WebObjectsPro product provides additional WebObjects Adaptor components:

- The NSAPI WebObjects Adaptor integrates with the Netscape Commerce Server, giving better security and faster performance.
- C source code for both the CGI WebObjects Adaptor and the NSAPI WebObjects Adaptor which lets you customize the WebObjects Adaptor to work with a Web server that resides on a platform that is not NEXTSTEP, Solaris, or Windows NT.

WebObjects Adaptor Behavior

The WebObjects Adaptor is started by each new request coming into the server and exits as soon as it has transmitted the corresponding response back. Each time the WebObjects Adaptor wakes, it makes attempts to contact a WebObjects application.

The WebObjects Adaptor attempts to make a connection with a currently-running WebObjects application. If the WebObjects application is not running, the WebObjects Adaptor attempts to autostart the application. If the WebObjects Adaptor does not connect with the WebObjects application, the WebObjects Adaptor fails, returning an error message.

If the WebObjects Adaptor connects to the WebObjects application, it trades information with the WebObjects application using the HTTP protocol, passes HTTP responses containing HTML pages back to the Web server, then exits.

How the WebObjects Adaptor connects

Whenever the Web server receives a URL with `/cgi-bin/WebObjects...` in it, it awakens the WebObjects Adaptor. (Note that you can change the name of the *cgi-bin* directory; for example, with the Microsoft IIS Server on NT, the URL contains `/Scripts/WebObjects.exe` in it.)

The Web server sends the WebObjects Adaptor information found in the URL requests, according to CGI protocol, using environment variables and the stdin() character string. The WebObjects Adaptor uses two configuration files as it begins the process of locating a particular WebObjects application:

- A private **WebObjects.conf** file, on UNIX systems located in the **/tmp** directory, on Windows NT systems located in whatever directory is specified by the *TEMP* environment variable. WebObjects applications add their port number information to the private WebObjects.conf file at the time they start up.

- A public **WebObjects.con**f file located in the *cgi-bin* directory, or in the **ns-home/httpd-80/config** directory in the case of the NSAPI WebObjects Adaptor. The public **WebObjects.conf** file is optional, but if it exists and if it lists WebObjects applications, then it overrides the private WebObjects.conf file, for a particular WebObjects application.

If the WebObjects Adaptor fails to connect with a requested WebObjects application, it returns a single error message: "Did not received any response from application." The anonymity of this message helps protect the details of your site from unwanted user inspection. To get details on the cause of failure, you should inspect the WebObjects Adaptor's **CGIMessenger.log** file. The exceptions to this one-error-message rule are error messages related to URL syntax.

The WebObjects Adaptor inspects the information it gets from the Web server and attempts to connect with a WebObjects application in the following ways

- If the information specifies an instance number and host, the WebObjects Adaptor looks for the public **WebObjects.conf** file to find a particular port and instance number for the specified WebObjects application. If successful, the WebObjects Adaptor then tries to connect. If the WebObjects Adaptor cannot find the **WebObjects.conf** file, or if the requested WebObjects application is not listed, or if the connection attempt fails, the WebObjects Adaptor fails, returning its error message.

- If the information does not specify an instance number and a host, the WebObjects Adaptor looks first for the public **WebObjects.conf** file, then for the private **WebObjects.conf** file.

  If there is a public **WebObjects.conf** file, the WebObjects Adaptor reads all the instances of application under the name requested and picks one randomly. Then it tries to connect. If the connection attempt fails, the WebObjects Adaptor fails, returning the standard error message.

  If there is no public **WebObjects.conf** file, or if there is one but the application requested is not listed in it, the WebObjects Adaptor reads the private **WebObjects.conf** file. If the application is listed there, it tries to connect. If the application is not listed there, or if the connection failed, the WebObjects Adaptor will attempt to autostart the WebObjects application requested.

Autostarting a WebObjects application

By design, when the connection process begins, the WebObjects Adaptor first assumes all WebObjects applications are currently running. If, in the connection process, the WebObjects Adaptor finds a particular WebObjects application is not running, the WebObjects Adaptor may try to automatically start up that application.

- The WebObjects Adaptor can only autostart a WebObjects application that resides on the computer hosting the Web server.
- There is no way to use a URL to force autostarting. If the WebObjects application requested is already runnning and accessible following the scheme described above, the WebObjects Adaptor will connect to it and won't autostart a new one.
- The WebObjects Adaptor is intelligent enough to autostart only the applications present on your system.

The WebObjects Adaptor responds to a URL in a sequence of steps, stopping when it makes a connection. For example, given a request that includes the string

```
.../cgi-bin/WebObjects/Examples/myApp
```

The WebObjects Adaptor performs the following sequence:

1. Look for the executable <*NeXT_Root*>**/NextLibrary/WebObjects/Executables/Examples/myApp.app/myApp**

2. if not found, look for the executable *<NeXT_Root>***/NextLibrary/WebObjects/Executables/Examples/myApp.debug/myApp**
3. if not found, look for the executable *<NeXT_Root>***/NextLibrary/WebObjects/Executables/Examples/myApp/myApp**
4. if not found, look for the executable *<NeXT_Root>***/NextLibrary/WebObjects/Executables/Examples/myApp**
5. if not found, look for the executable *<Document_Root>***/WebObjects/Examples/myApp.app/myApp**
6. if not found, look for the executable *<Document_Root>***/WebObjects/Examples/myApp.debug/myApp**
7. if not found, look for the executable *<Document_Root>***/WebObjects/Examples/myApp/myApp**
8. if not found, look for the executable *<NeXT_Root>***/NextLibrary/WebObjects/Executables/EOFDefaultApp**
9. if not found, look for the executable *<NeXT_Root>***/NextLibrary/WebObjects/Executables/DefaultApp**

10.if not found, return the standard error message.

In the first eight cases, as soon as the executable is found, the application will be started with

```
myApp -d <Document_Root>/Examples/myApp
```

The last argument represents the path to the applicatio..

In the last two cases, the WebObjects Adaptor also tries to find the following directories:

- Look for the resources directory *<Document_Root>***/WebObjects/Examples/myApp**
- if not found, look for resources directory *<NeXT_Root>***/NextLibrary/WebObjects/Executables/myApp**
- if not found, return the standard error message.

As soon as one of these directories is found, the application will be started with:

```
[EOF]DefaultApp -d <Document_Root>/Examples/myApp
```

The last argument represents the path to the application.

URLs for WebObjects

Generally, a WebObjects application responds to a URL placed in a Web page. The format for a WebObjects application URL is

```
protocol://<ServerHost>/cgi-
bin/WebObjects/Application_subpath/AppName:33@<AppHost>
```

For example, the Web server may receive a URL requesting a WebObjects application of the form

```
http://localhost/cgi-bin/WebObjects/Examples/HelloWorld:33@rhino
```

which reads as

- Use the http: protocol.
- Look on this (localhost) computer.
- Run the WebObjects Adaptor program in the cgi-bin directory.
- Pass the application subpath, port, and hostname (`Examples/HelloWorld:33@rhino`) to the WebObjects Adaptor.

The WebObjects Adaptor prepends the application subpath with the path to the Web server's *DocumentRoot*, which in this example is

<*NeXT_Root*>**/NextLibrary/WebServer/htdocs/WebObjects/Examples/HelloWorld**

The /tmp/logWebObjects and /tmp/CGIMessenger.log files

The WebObjects Adaptor generates a **/tmp/CGIMessenger.log** file if you create **/tmp/logWebObjects** file. Inspect the **CGIMessenger.log** file to see a record of the WebObjects Adaptor's activities as it attempts to connect to WebObjects applications.

- Lines beginning with INFO record normal activity.
- Lines beginning with WARNING record problems.

The DefaultApp Program

The **DefaultApp** program is a binary executable located under your WebObjects directory that interprets scripted WebObjects applications. It resides in the <*NeXT_Root*>**/NextLibrary/WebObjects/Executables** directory.

A copy of **DefaultApp** must run for each instance of a scripted WebObjects application.

Using DefaultApp

You can invoke the **DefaultApp** program on the command line. The WebObjectgs Adaptor may call the **DefaultApp** program as part of its autostart routine. **DefaultApp** will not start an application on a remote host.

The usage for DefaultApp is

```
DefaultApp [applicationName] [-c] [-d DocumentRoot] [-v
WebObjectsVersion] [-n InstanceNumber] [-p PortNumber] [-]
ApplicationDirectory
```

where

- `[applicationName]` is the generic name of the application
- `[-c]` is used for caching information about the session in the pages returned
- `[-d DocumentRoot]` need not be specified as long as your Web server complies with CGI specifications and passes its *DocumentRoot* path to the WebObjects Adaptor.
- `[-v WebObjectsVersion]` forces use of a particular WebObjects Adaptor and corresponding **DefaultApp** or compiled WebObjects application; the string `WebObjectsVersion` should match the suffix of the WebObjects Adaptor: the string following (not including) the hyphen character (for example, "Beta" for WebObjects-Beta or "C" for WebObjects-C).
- `[-n InstanceNumber]` is an integer that the WebObjects Adaptor creates using random number generation. If your URL does not specify the instance number, the application is presumed to run on the server machine as a single instance application, as if it had been autostarted.
- `[-p PortNumber]` specifies the socket port (for socket use only).
- [-] escapes ApplicationDirectory names that begin with the hyphen character. For example `DefaultApp - -oddname/myApp` uses the hyphen option to allow the directory name `-oddname`, which would otherwise be treated as the `-o` option followed by `ddname`, with failed results.
- `[ApplicationDirectory]` specifies the path to the application, but is not necessary if the application wrapper is found directly under the document root.

Note: Windows NT users should remember that the Web server uses the `ApplicationDirectory` argument to build URLs, so use forward slashes as opposed to a backslashes.

The WebObjects.conf File

The **WebObjects.conf** configuration file is a text file that you create in your Web server's *cgi-bin* directory.

Note: Note: If you want to use the **WebObjects.conf** file with WebObjects' Netscape Interface (NSAPI) Adaptor, then locate **WebObjects.conf** in your **ns-home/http[d/s-port]/config/** directory rather than in your Web server's *cgi-bin* directory.

The WebObjects Adaptor inspects the **WebObjects.conf** file to resolve URL information or to autostart a WebObjects application.

The way in which you specify WebObjects applications in the **WebObjects.conf** file lets you institute load-balanced distribution of WebObjects applications across your network.

The Structure of WebObjects.conf

The **WebObjects.conf** configuration file is a line-oriented ASCII file that on each line names an application, host, and maximum number of instances to run on the host, using space character separators.

The format of a line in the configuration file **WebObjects.conf** is always

```
[ApplicationDirectory]:[ApplicationNumber]@[hostname]
[ApplicationPort]
```

- `[ApplicationDirectory]` is the path below the WebObjects directory of your Web server's *DocumentRoot* which lets the adaptor find the application's resources.
- `[ApplicationNumber]` lets you identify a particular instance of a WebObjects application.
- `@[hostname]` lets you identify a particular host computer.
- [ApplicationPort] lets you specify a port for a particular instance of a WebObjects application on a particular host.

Note the following additional rules for creating a **WebObjects.conf** file.

- The WebObjects Adaptor skips empty lines in the configuration file.

- You can add comment lines. The WebObjects Adaptor will attempt to parse them, then skip them.
- You can specify up to 256 applications with the same name. The WebObjects Adaptor will ignore any more.
- The WebObjects Adaptor will parse the first 256 characters of each line, skipping subsequent characters.

Here are sample contents of a WebObjects.conf file:

```
Examples/HelloWorld:1@onemachine 3000
Examples/CyberWind:2@twomachine 2002
Examples/CyberWind:1@threemachine 4000
Examples/CyberWind:4@fourmachine 2004
```

Netscape API Interface Adaptor

The standard WebObjects Adaptor in your Web server's *cgi-bin* directory is a CGI executable that handles requests for WebObjects applications.

If you're using WebObjects Pro, and if your system hosts Netscapes Commerce Server, you can install WebObjects' Netscape Interface (NSAPI) Adaptor in place of the standard WebObjects Adaptor. The WebObjects NSAPI Adaptor is designed to be integrated with the Netscape Commerce Server according to the Netscape API, and thus eliminates the overhead of calling a separate CGI process.

The advantages of WebObjects NSAPI Adaptor are that it has no extra CGI processes between the server and the Web Application, and as a consequence, improved robustness and security (if your server machine has been misconfigured). (See the Netscape page with this claim at http://www.netscape.com/newsref/std/nsapi_vs_cgi.html)

The disadvantage is that the WebObjects Netscape Interface Adaptor is harder to put into place (especially on NT).

The WebObjects NSAPI Adaptor works with the Netscape Commerce Server 1.1 with WebObjects version Beta1 and greater on

- Windows NT 3.51 and greater
- Solaris 2.3 and greater

WebObjects NSAPI Adaptor is named **WONetscapeServerAdaptor.so** on Solaris platforms.

Once installed, the WebObjects NSAPI Adaptor's behaviour will be completely transparent to you. URLs are not affected. Always use `.../cgi-bin/WebObjects...` in your URL, even though the request is not going through any CGI program. The string `/cgi-bin/WebObjects` is used as a key by the Netscape Commerce Server to use the linked-in Netscape Interface Adaptor.

Note: The Netscape Commerce Server will work with both kinds of Adaptors: the standard CGI WebObjects Adaptor or the linked-in WebObjects NSAPI Adaptor.

Note: If you want to use the **WebObjects.conf** file with the Netscape Interface Adaptor, then locate **WebObjects.conf** in your **ns-home/http[d/s-port]/config/** directory rather than in your *cgi-bin* directory as for the standard WebObjects Adaptor.

Modifying configuration files

You need to modify the following files in the Netscape Commerce Server's **ns-home/http[d/s-port]/config** directory.

- In **magnus.conf**

```
Init fn=load-modules
shlib=/usr/NextLibrary/WebObjects/NSAPI-bin/WONetscapeServerAdaptor.so
funcs="WONetscapeInterface,WONetscapeInterfaceFindWebObjects"
```

Note: Enter the funcs argument exactly as shown here, with no spaces inside the quotes.

- In **obj.conf**, just before the `NameTrans`

```
from="/cgi-bin" fn="pfx2dir"
dir="/usr/local/etc/ns-insecure/cgi-bin"
name="cgi"
NameTrans from="/cgi-bin/WebObjects" fn="WONetscapeInterfaceFindWebObjects"
name="webobjects"
```

and at the end of **obj.conf**

```
< Object name="webobjects">
Service fn="WONetscapeInterface"
< /Object>
```

NetscapeServerAdaptor.dll installation on Windows NT

In the NT Server Help application, read the chapter *Using the NT Registry* before you begin setting up <*NeXT_Root*>**/NextLibrary/WebObjects/NSAPI-bin/NetscapeServerAdaptor.dll**.

The Netscape Commerce Server on Windows NT doesn't provide any utility program to edit their developer configuration registry. It is not possible to automate this procedure as some customizations are necessary to the installation. Therefore, the following instructions should only be attempted by a Netscape server administrator. Be sure to follow these instructions to the letter, as any error could hang your Netscape server.

Getting started with the Registry Editor

1. To run the Registry Editor, choose `File|Run` and type `REGEDT32.EXE` (you can also type start `REGEDT32` from a command window).

2. The Registry contains information in a hierarchical structure similar to the organization of files and directories on your hard disk. The Netscape Server configuration information is in the `HKEY_LOCAL_MACHINE` folder under `Software | Netscape`.

3. Choose `httpd`, `https`, or `httpd-80` whichever you want to configure, then choose `CurrentVersion`.

Adding a new object

4. At this stage, you should see two folders **Objects** and **Startup**. Select **Objects** to add a new object to the list

5. Choose `Edit|Add Key` and type `Object12` in the `Key Name` field. Select the new object created, choose the `Edit|Add value`. Type `name` in the `Value Name` field, validate it, and type `webobjects` in the `string` field. Your new object `Object12` should now look like `name : REG_SZ : webobjects`, similar to the other objects listed, except it won't show a + sign in its folder, for the moment.

6. Select your new object, `Object12`, choose `Edit|Add Key` and type `Directive10` in the `Key Name` field. Select the new directive created and choose `Edit|Add Value`. Type `DirectiveName` in the `Value Name` field, validate it, and type `Service` in the `string` field.

7. Select your new directive, `Directive10`, choose `Edit|Add Key` and type `Function10` in the `Key Name` field. Select the new function created and choose `Edit|Add Value`. Type `fn` in the `Value Name` field, validate it, and

type `WONetscapeInterface` in the `string` field. This completely specifies the `webobjects` object.

Modifying the default object

8. In `Object10`, select `Directive10`, and choose `Edit|Add Key`. Type `Function109` in the `Key Name` field. Select the new function created and choose `Edit|Add Value`. Type `fn` in the `Value Name` field, validate it, and type `WONetscapeInterfaceFindWebObjects` in the `string` field. Choose `Edit|Add Value` and type `from` in the `Value Name` field, validate it, and type `/cgi-bin/WebObjects` in the `string` field. Choose `Edit|Add Value` and type `name` in the `Value Name` field, validate it and type `webobjects` in the `string` field. You should now have a set of parameters like the following

```
fn : REG_SZ : WONetscapeInterfaceFindWebObjects
from : REG_SZ : /cgi-bin/WebObjects
name : REG_SZ : webobjects
```

Important: The registry name `Function109` is not the important thing here: your new function must be the one immediately before the function defined by the following parameters

```
dir : REG_SZ : c:/NETSCAPE/cgi-bin
fn : REG_SZ : pfx2dir
from : REG_SZ : /cgi-bin
name : REG_SZ : cgi
```

This example assumed an uncustomized Netscape server, you had to insert your new function between `Function10` and `Function11`, which meant you add to pick the name `Function109`. The instructions may not apply if you have a different Netscape server configuration.

Modifying the startup config

9. Now select the **StartUp** registry which is at the same level as the **Objects** registry you have been working on for the moment.

10. Choose `Edit|Add Key` and type `InitFunction11` in the `Key Name` field. Select the new function created, choose `Edit|Add Value`, and type `fn` in the `Value Name` field, validate it, and type `load-modules` in the `string` field. Choose `Edit|Add Value` and type `shlib` in the `Value Name` field, validate it, and in the `string` field enter the path on your machine to the **NetscapeServerAdaptor.dll** file (for example, **C:/<:NeXT_Root>/NextLibrary/WebObjects/NSAPI-bin/NetscapeServerAdaptor.dll**.

11. Choose `Edit|Add Key` and type `InitFunction11` in the `Key Name` field. Select the new function created, choose `Edit|Add Value`, and type `funcs` in the `Value Name` field, validate it and type `WONetscapeInterface,WONetscapeInterfaceFindWebObjects` in the `string` field. You should now have a set of parameters like this:

```
  fn : REG_SZ : load-modules
  shlib : REG_SZ :
  C:/NeXT/NextLibrary/WebObjects/NSAPI-bin/NetscapeServerAdaptor.dll funcs :
REG_SZ : WONetscapeInterface,WONetscapeInterfaceFindWebObjects
```

Important: The registry name `InitFunction11` is not the important thing here, but your new function must be located between the init function(s) with (`fn : REG_SZ : load-types`) and the init function(s) with (`fn : REG_SZ : init-clf`). The order among the init functions with `fn : REG_SZ : load-modules`, if there are others than the one you just created, is irrelevant.

This example assumes an uncustomized Netscape server, so you must insert your new function between `Function1` and `Function2`, which means you have to pick the name `Function11`. The exact name may vary if your Netscape server is differently configured.

12. Restart your server

Tuning, Testing, and Debugging

If you have problems connecting with your WebObjects applications

- Manually start WebObjects applications by using `./DefaultApp` on the command line.
- Load balance WebObjects applications through specifications in the **WebObjects.conf** file.
- Check the behavior of the WebObjects Adaptor by inspecting the WebObjects Adaptor's **CGIMessenger.log** file.
- Use the WebObjects Adaptor Test Plan to follow a step-by-step procedure that tests four important WebObjects Adaptor features.

Manually starting a WebObjects application

For debugging or deployment reasons, you may want to start your WebObjects application by hand.

There are two ways of doing this from **Terminal.app**'s command window. Both use the **DefaultApp** executable that's in the <*NeXT_Root*>**/NextLibrary/WebObjects/Executables** directory.

Starting a single instance of a WebObjects application

This feature is only available on the computer that hosts your Web server, and only if WebObjects has been installed on it. As an example, to manually start **HelloWorld**

- Change directories to <*NeXT_Root*>/NextLibrary/WebObjects/Executables and enter
  ```
  ./DefaultApp Examples/HelloWorld
  ```

This will start a single instance of the WebObjects application **HelloWorld** located in your *DocumentRoot***/Examples/HelloWorld** directory.

This is the simplest way to start an application by hand, with the limitation that you can only have one application with this name running on the server host.

Starting multiple instances of a WebObjects application

This feature is available on any computer with WebObjects installed on it, including possibly the computer that hosts the Web server. It is done in two steps.

1. On a computer called <*myHost*>, change directories to <*NeXT_Root*>**/NextLibrary/WebObjects/Executables** and start the application in **Terminal.app**'s command window by entering
   ```
   ./DefaultApp -p 3000 -n 1 Examples/HelloWorld
   ```

This will start **instance1** of **HelloWorld** on port 3000 of <*myHost*>.

2. Go to the computer hosting your Web server <*myServerHost*> and create or edit the **WebObjects.conf** file in your Web server's *cgi-bin* directory by entering the following line in the file
   ```
   Examples/HelloWorld:1@myHost 3000
   ```

   Save the file. This lets the WebObjects adaptor know that an application

HelloWorld is available on the computer named <*myHost*> at port 3000 for serving requests.

Now try contacting your application with the same URL as before:

```
http://<myServerHost>/cgi-bin/WebObjects/Examples/HelloWorld
```

All requests for **Examples/HelloWorld** will be forwarded on <*myHost*> to this WebObjects application.

You can also force the Web server to use one particular application running on a host by typing in your browser

```
http://<myServerHost>/cgi-
bin/WebObjects/Examples/HelloWorld:1@myHost
```

Remember always to pick ports greater than 1024 as ports under 1024 are reserved on UNIX platforms.

Load balancing WebObjects applications

The WebObjects Adaptor includes a load balancing feature to ensure that WebObjects application requests are spread evenly across your network.

The example in this section assumes that WebObjects is installed on the computer hosting your Web server and on two other computers.

- <*WebServerHost*> hosts the Web server and one instance of **HelloWorld**.
- <*FirstHost*> hosts two instances of **HelloWorld**.
- <*SecondHost*> hosts one instance of HelloWorld and one instance of **CyberWind**.

The following instructions show you how to use **Terminal.app**'s command window to start up the WebObjects applications, modify the **WebObjects.conf** configuration file, and test the results.

1. From <*FirstHost*>, in a command window, change directories to <*NeXT_Root*>**/NextLibrary/WebObjects/Executables** and start the two **HelloWorld** applications by entering
   ```
   ./DefaultApp -p 3000 -n 1 Examples/HelloWorld ./DefaultApp -p
   4000 -n 2 Examples/HelloWorld
   ```
2. From <*SecondHost*>, in a command window, change directories to <*NeXT_Root*>**/NextLibrary/WebObjects/Executables** and start **HelloWorld** and **CyberWind** by entering
   ```
   ./DefaultApp -p 3000 -n 1 Examples/HelloWorld ./DefaultApp -p
   4000 -n 1 CyberWind
   ```

3. From <*ServerHost*>, in a command window, change directories to <*NeXT_Root*>**/NextLibrary/WebObjects/Executables** and start **HelloWorld** by entering
   `./DefaultApp -p 5000 -n 33 Examples/HelloWorld`

4. On <*ServerHost*>, edit the file **WebObjects.conf** in your Web server's *cgi-bin* directory to look like this

```
Examples/HelloWorld:1@<myHost> 3000 Examples/HelloWorld:2@<myHost> 4000
Examples/HelloWorld:1@<mySecondHost> 3000 CyberWind:1@<mySecondHost> 4000
Examples/HelloWorld:33@<myServerHost> 5000
```

5. Using the same URL as before, try contacting your application
   `http://<ServerHost>/cgi-bin/WebObjects/Examples/HelloWorld`

The WebObjects Adaptor will pick one of the four HelloWorld applications it knows about and try to contact it. If the application is busy serving someone else, the WebObjects Adaptor will wait. If the application died or was not started, the WebObjects Adaptor will fail the request and return its standard error message.

If your WebObjects application carries state for the client between requests, the generated HTML pages will contain the complete path to return to the exact application that served the given client in his previous request, with links such as

```
http://<myServerHost>/cgi-
bin/WebObjects/Examples/HelloWorld:33@<myServerHost>
```

and no load-balancing will be performed as long as the session lasts.

If your WebObjects application does not maintain state, the application will automatically return HTML pages with links such as

```
http://<myServerHost>/cgi-bin/WebObjects/Examples/HelloWorld
```

so that the WebObjects Adaptor can perform load balancing on subsequent requests, possibly contacting a different application.

As there is a single instance of the application **CyberWind**, all requests for **CyberWind** will always go to this same application. If you experience too much traffic for this single instance, you can deploy it on other hosts as you did with **HelloWorld**.

The WebObjects Adaptor will contact both autostarted applications and load-balanced applications (started by hand) at all times, as long as they don't have the same names. When names conflict, the WebObjects Adaptor will pick load-balanced applications rather than any previously autostarted application.

Inspecting CGIMessenger.log

Because something may go wrong as you install and deploy your WebObjects applications, the WebObjects Adaptor includes a debugging feature. To trigger the WebObjects Adaptor's debugging mode, follow these instructions.

Using logWebObjects on UNIX systems

Log in as (or su to) root and open a **Terminal.app** command window.

1. create an empty **/tmp/logWebObjects** file
   ```
   touch /tmp/logWebObjects
   ```

2. enter
   ```
   tail -f /tmp/CGIMessenger.log
   ```

This will log any activity of the WebObjects adaptor, and you will be able to trace the WebObjects Adaptor's behavior as it tries to connect to the WebObjects application. Do not forget to remove the **/tmp/logWebObjects** file to stop the logs when you have successfully deployed your WebObjects application.

Using logWebObjects on Windows NT systems

In order to enable debugging you should create the **logWebObjects** file in the directory specified by the *TEMP* environment variable.

You can use Notepad to view the resulting **CGIMessenger.log** file. Keep in mind that the WebObjects Adaptor changes the contents of the **CGIMessenger.log** file as it responds to each request for a WebObjects application, so you may need to close and reopen the file in order to monitor the WebObjects Adaptor's activity.

WebObjects Adaptor Test Plan

Before you develop your own WebObjects applications, you might want to use this test plan to be sure WebObjects is working properly. If you've modified the C source code for the WebObjects Adaptor, be sure to run this test plan.

The WebObjects Adaptor attempts to contact a requested WebObjects application by following four steps:

1. Choosing an application from the information in the URL and the **WebObjects.conf** configuration file.

2. Load Balancing with an unresolved URL using the **WebObjects.conf** configuration file.

3. Choosing an application from the private configuration file. (**/tmp/WebObjects.conf**).

4. Autostarting an application listed in the private configuration file.

If the WebObjects Adaptor fails to perform the first step, it jumps to the next.

This test plan tests the four main features of the WebObjects Adaptor. All tests should succeed unless stated otherwise. Perform the four tests in order, as each depends on the success of the previous test.

The test plan uses the **HelloWorld** and **CyberWind** example programs supplied with WebObjects. You'll use different URLs to access these applications with respect to the contents of the **WebObjects.conf** configuration file. You can check the WebObjects Adaptor's behavior by inspecting the **CGIMessenger.log** file to see if things are working as they should.

Setting up for testing

Be sure to have the WebObjects example programs **HelloWorld** and **CyberWind** in your Web server's *DocumentRoot* directory and start them up.

In a browser on a computer that does not host the Web server, type an URL to access one of these WebObjects applications.

On the Web server host, login as root and create a dummy **logWebObjects** file in the **/tmp** directory. This file will trigger the WebObjects Adaptor to log information about its behavior in the **/tmp/CGIMessenger.log** file. .

For example, from **Terminal.app**'s command window issue the following commands

```
your_host>  cd /tmp
your_host>  touch logWebObjects
your_host>  tail -f CGIMessenger.log
```

The **-f** option for the **tail** command will generate continual updates to screen as the log file changes. You should read and understand the log information before marking a test as passed.

Use the **ps** command to verify that programs are running.

On NEXTSTEP
```
ps -aux | grep DefaultApp
```

On Solaris
```
ps -eaf | grep DefaultApp
```

Now you should be ready for the first test.

Responding to a fully resolved URL

A URL that completely describes an application (fully resolved) looks like:

```
http://<host>/cgi-bin/WebObjects/CyberWind:1@<host>
```

or

```
http://<host>/cgi-bin/WebObjects/CyberWind:1
```

The WebObjects Adaptor first assumes the application is on the computer hosting the Web server. When the adaptor receives such an URL, it looks in the *cgi-bin*/**WebObjects.conf** file to find a port corresponding to this application. If the WebObjects Adaptor finds one, it tries to connect to the application. If the WebObjects Adaptor does not find a port number, it fails and returns an error message.

Testing a fully resolved URL

This first set of instructions shows you how to test the way the WebObjects Adaptor responds to URL requests for a WebObjects application.

1. From a command window start four applications

```
DefaultApp -p 3000 -n 1 HelloWorld DefaultApp -p 4000 -n 1 CyberWind
DefaultApp -p 5000 -n 2 HelloWorld DefaultApp -p 6000 -n 2 CyberWind
```

2. Create a *cgi-bin*/**WebObjects.conf** file with the following contents (try different ordering)

```
HelloWorld:1@<host> 3000
CyberWind:1@<host> 4000
HelloWorld:2@<host> 5000
CyberWind:2@<host> 6000
```

3. Try to contact the applications by issuing URLs from a browser on a remote computer.

```
http://<host>/cgi-bin/WebObjects/HelloWorld:1@<host>
http://<host>/cgi-bin/WebObjects/CyberWind:1@<host>
http://<host>/cgi-bin/WebObjects/HelloWorld:2@<host>
http://<host>/cgi-bin/WebObjects/CyberWind:2@<host>
http://<host>/cgi-bin/WebObjects/HelloWorld:1
http://<host>/cgi-bin/WebObjects/CyberWind:1
http://<host>/cgi-bin/WebObjects/HelloWorld:2
http://<host>/cgi-bin/WebObjects/CyberWind:2
```

This should work with no error messages. If so, go on to the second test.

Load Balancing with an unresolved URL

The WebObjects Adaptor responds to an unresolved URL by randomly picking a WebObjects application from the **WebObjects.conf** file. This random selection tends to distribute activity evenly across WebObjects application hosts.

An unresolved application URL looks like

```
http://<host>/cgi-bin/WebObjects/CyberWind
```

When the adaptor receives such an URL, it reads the *cgi-bin***/WebObjects.conf** file to look for any application available under the name **CyberWind** and picks one randomly. Then the Adaptor completes the URL into a form such as

```
http://<host>/cgi-bin/WebObjects/CyberWind:1@<host>
```

As the user successively clicks on links, the same application should always be called back. There should be no more unresolved links in the page. All links in the page should look like

```
http://<host>/cgi-bin/WebObjects/CyberWind:1@<host>
```

rather than

```
http://<host>/cgi-bin/WebObjects/CyberWind
```

Testing load balancing with an unresolved URL

This second sequence tests the WebObjects Adaptor's response to unresolved URLs, with two additional WebObjects applications running.

1. Add two more WebObjects applications on the Web server host, while the same four applications stay running.

```
DefaultApp -p 6010 -n 20 CyberWind DefaultApp -p 6020 -n 30 CyberWind
```

2. Complete the /cgi-bin/WebObjects.conf file by adding two new entries

```
CyberWind:20@<host> 6010 CyberWind:30@<host> 6020
```

3. Restart the browser on the remote computer and try to make contact with the application

```
http://<host>/cgi-bin/WebObjects/CyberWind:20
http://<host>/cgi-bin/WebObjects/CyberWind:30
http://<host>/cgi-bin/WebObjects/HelloWorld
http://<host>/cgi-bin/WebObjects/CyberWind
```

4. Type several reloads from the browser at this stage and check the **CGIMessenger.log** to see that a different **CyberWind** application is picked each time.

5. Go back through the application pages to the board page. Verify that your state is preserved (that is, if you select something in the board page, then go to another page and come back, your selection is still there.

6. Now kill the two last processes.

```
CyberWind:20@<host> 6010
CyberWind:30@<host> 6020
```

7. Try again requesting for

```
http://<:host>/cgi-bin/WebObjects/CyberWind
```

   This should fail two times out of six, as two of the six **CyberWind** applications listed in the configuration file are not running any more.

Now go on to the third test.

Choosing an application from the temporary configuration file

The WebObjects Adaptor may receive an unresolved URL for an application such as

```
http:///cgi-bin/WebObjects/HelloWorld
```

The WebObjects Adaptor first parses the *cgi-bin*/**WebObjects.conf** file.

There are cases when WebObjects Adaptor fails to find or open the *cgi-bin*/**WebObjects.conf** file, or when the application is not listed in the *cgi-bin*/**WebObjects.conf** file. In these cases, the WebObjects Adaptor proceeds by opening the **/tmp/WebObjects.conf** file, which is a non-public file, and should not be touched by the user.

The **/tmp/WebObjects.conf** file can only have one instance of a given application. If the WebObjects Adaptor finds the application it is looking for there, it will get its port number and connect to it.

When you start up an application (for example in gdb) without a **-n** argument, the port information will automatically be saved in the **/tmp/WebObjects.conf** file. There is no need to list the app in the *cgi-bin*/**WebObjects.conf** file, as it will be considered unique on the computer hosting the Web server and has no application number.

Testing application choice from the temporary configuration file

To begin this third test, restart your browser on the remote computer. On the computer hosting the Web server, kill the **HelloWorld** processes and remove them from the *cgi-bin***/WebObjects.conf** file. Make sure you still have the four **CyberWind** processes (use the ps command to check).

1. Start up **HelloWorld**

```
./DefaultApp HelloWorld
```

2. Try to connect to HelloWorld from the browser.

```
http:///cgi-bin/WebObjects/CyberWind
```

   This should load-balance, based on the entries in the *cgi-bin***/WebObjects.conf** file.

3. Type several reloads from the browser and observe in the **/tmp/CGIMessenger.log** window that a different **CyberWind** application is picked each time. Then go through the application pages and go back to the board page. Verify that your state is preserved (that is, if you select something in the board page go to another page and come back, your selection is still there.

```
http:///cgi-bin/WebObjects/HelloWorld
```

This should use the **/tmp/WebObjects.conf** file. Try reloading to check that load-balancing doesn't work; you should always come back to the same application.

4. Now kill the HelloWorld process and start it up again

```
./DefaultApp -p 4000 HelloWorld
```

   This should fail because port 4000 is already claimed by `CyberWind:1`.

5. Try an unclaimed port number.

```
./DefaultApp -p 3000 HelloWorld
```

   From the browser try to contact CyberWind.

```
http:///cgi-bin/WebObjects/CyberWind
```

   This should load-balance using the *cgi-bin***/WebObjects.conf** file.

6. Reload from the browser, check the **CGIMessenger.log** window for a different **CyberWind** application picked each time, then return the board page to verify that state is preserved.

```
http:///cgi-bin/WebObjects/HelloWorld
```

This should use the **/tmp/WebObjects.conf** file. Try reloading to check that load-balancing doesn’t work; you should always come back to the same application.

7. Now kill all **CyberWind** processes and remove them from the *cgi-bin***/WebObjects.conf** file.

8. On the computer hosting the Web server start up **CyberWind**.

   ```
   ./DefaultApp CyberWind
   ```

   From the browser try to connect to **HelloWorld**.

   ```
   http:///cgi-bin/WebObjects/HelloWorld
   ```

   This should use the **/tmp/WebObjects.conf** file. Try reloading to check that load-balancing doesn't work; you should not always come back to the same application.

9. Now try

   ```
   http:///cgi-bin/WebObjects/CyberWind
   ```

   This should use the **/tmp/WebObjects.conf** file. Try reloading to check that load-balancing doesn't work; you should not always come back to the same application.

Before you go on to the fourth and final test suite, kill the **CyberWind** and **HelloWorld** processes started in this section.

Autostarting from the temporary configuration file

In addition to the *cgi-bin***/WebObjects.conf** file, the WebObjects Adaptor maintains a private version in the **/tmp** directory. This **/tmp/WebObjects.conf** file is created by the first WebObjects application that starts up after boot time; as subsequent WebObjects applications start up, they record their port and host information. Be careful not to alter this file in any way. The following example illustrates its use.

The WebObjects Adaptor may receive an application request with an unresolved URL such as

```
http:///cgi-bin/WebObjects/CyberWind
```

The Adaptor first parses the *cgi-bin***/WebObjects.conf** file. If the Adaptor fails to find a match, it tries to parse the **/tmp/WebObjects.conf** file. There are cases when the WebObjects Adaptor fails. There may be no valid **/tmp/WebObjects.conf** file, or the application may not be listed in the **/tmp/WebObjects.conf** file, or the application listed in the **/tmp/WebObjects.conf** file may be a dead process. In these cases, the WebObjects Adaptor then tries to autostart the application **CyberWind** on the computer hosting the Web server.

Testing autostarting from the temporary configuration file

To test autostarting, restart your browser.

1. On the computer hosting the Web server, kill the **HelloWorld** processes and remove them from the *cgi-bin*/**WebObjects.conf** file. Make sure you still have the four **CyberWind** processes.

2. From the browser try to connect to **CyberWind**.

   ```
   http:///cgi-bin/WebObjects/CyberWind
   ```

   This should load-balance using the *cgi-bin*/**WebObjects.conf** file.

3. Reload from the browser, check the **CGIMessenger.log** to see that a different **CyberWind** application picked each time, then return the board page to verify that state is preserved.

4. Try to connect to **HelloWorld**.

   ```
   http:///cgi-bin/WebObjects/HelloWorld
   ```

   This should autostart **HelloWorld** and use the **/tmp/WebObjects.conf** file. Try reloading to check that load-balancing doesn't work; you should not always come back to the same application.

5. Now kill all **CyberWind** processes and remove them from the *cgi-bin*/**WebObjects.conf** file.

6. From the browser try to connect to **HelloWorld**.

   ```
   http:///cgi-bin/WebObjects/HelloWorld
   ```

   This should autostart **HelloWorld** using the **/tmp/WebObjects.conf** file. Try reloading to check that load-balancing doesn't work; you should not always come back to the same application.

7. Now try to connect to **CyberWind**.

   ```
   http:///cgi-bin/WebObjects/CyberWind
   ```

This should autostart **CyberWind** using the **/tmp/WebObjects.conf** file. Try reloading to check that load-balancing doesn't work; you shouldn't always come back to the same application.

8. Now try to contact a non-existent application.

   ```
   http:///cgi-bin/WebObjects/FooxXXXX
   ```

   This should fail.

This completes the basic test plan for the WebObjects Adaptor. Of course, you could explore a little further.

Extra Points

Install each WebObjects application on a different host, then repeat these tests.

Try this test plan for each different platform (Windows NT, Solaris, and NEXTSTEP) running WebObjects. (Remember that a Web server on Windows NT cannot access WebObjects applications on UNIX hosts.)

Repeat tests on each platform (Windows NT, Solaris, NEXTSTEP) not running WebObjects.

Repeat also on Windows NT and Solaris by using WebObjects Netscape Interface (NSAPI) Adaptor instead of the standard CGI WebObjects Adaptor. The syntax of the URLs stays the same.