# Oracle9*i* Application Server Forms Services

## Forms6*i* Patch 10: Oracle Forms Listener Servlet for Deployment of Forms on the Internet

*An Oracle White Paper*
*April 2002*

ORACLE

# Oracle9*i*AS Forms Services

## Forms6*i* Patch 10: Forms Listener Servlet for Deployment of Forms on the Internet

# Oracle9*i*AS Forms Services

## Forms6*i* Patch 10: Forms Listener Servlet for Deployment of Forms on the Internet

### OVERVIEW

This document describes the architecture option available for the Oracle Forms Services component in Oracle9*i* Application Server. In this document, the phrases "Oracle Forms Server" and "Oracle Forms Services" are used to indicate the set of components required to deploy Forms applications using the three-tier model.

### Pre-Patch 4 Architecture

At runtime, Oracle Forms Services consists of two separate components, the Forms Listener and the Forms Server Runtime. Each component runs as a separate process on the server machine.

The **Forms Listener** accepts new requests from clients that are executing Forms applications. When the process first starts, the Forms Listener creates a network endpoint on a port. Then, the Forms Listener goes into a wait state until it receives a network request from a client machine. Upon receiving the network request, the Forms Listener process creates a new Forms Server process and passes the details of the network connection to the Forms Server Runtime process.

The **Forms Server Runtime** runs Forms applications on the server machine. Forms Server Runtime is responsible for executing the code contained in the requested Forms application for a specific client. There may be more than one Forms Server Runtime process – one Forms Server Runtime process is created for each concurrent user. The Forms Server Runtime process assumes the client connection from the Forms Listener process and maintains the connection with the client for the duration of the Forms application session.

The Forms Server Runtime process uses a persistent connection to the client to send information in the form of structured messages about the running application, indicating what the client needs to display for the end user.

The client uses the same persistent connection to send structured messages back to the Forms Server Runtime process, indicating actions that the end user has performed.

## Socket, HTTP, and HTTPS Connection Modes

Initial releases of the Oracle Forms Server product used a simple method for connecting the client to the server. The connection from the client to the Forms Listener process was accomplished using a direct socket connection. The direct socket connection mode was suitable for companies providing thin client access to Forms applications *within* their corporate LANS/WANS. For the direct socket connection mode, the client had to be able to see the server machine and had to have permission to establish a direct network connection.

Although the direct socket connection mode is perfectly suited to deployments within a company's LAN/WAN, it is not the best choice for application deployment via unsecured network paths, such as the Internet. To safeguard valuable information and infrastructure assets, a company that is connected to the Internet typically employs a strict policy defining the types of network connections that can be made by clients on the un-trusted Internet to secure corporate networks. Permitting a direct socket connection from a client via the Internet exposes the company to potential invasions because the true identity of the client can be hard to determine.

With the widespread adoption of HTTP as the de-facto standard protocol for data transmission on the Internet, most companies permit HTTP traffic to enter and leave their corporate networks. Therefore, Oracle Forms Server 6*i* was extended to support data transmission using HTTP and HTTPS (in addition to the direct socket connection mode used in earlier versions).

Using the HTTP connection mode with Oracle Forms Server, structured messages sent to and from the client and server are encapsulated in standard HTTP messages. Companies that permit Internet access to their corporate servers through the firewall using HTTP can deploy Forms applications in the same manner, as shown in the following figure.

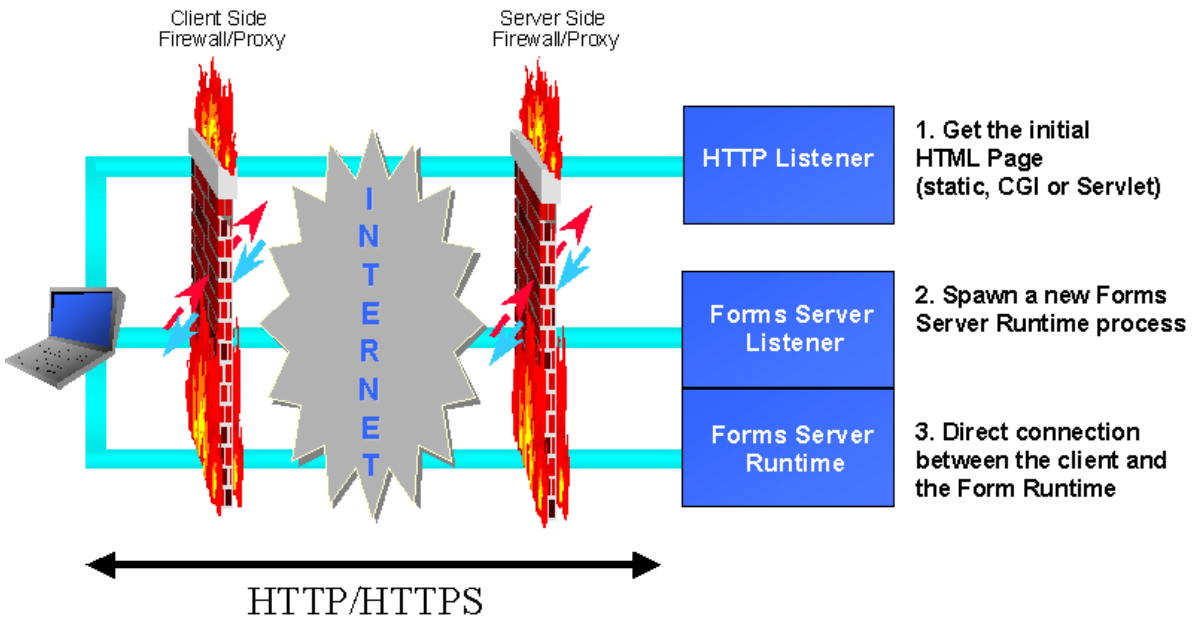



**Figure 1. Pre-Patch 4 Forms Server architecture for Internet deployment**

### Issues with the Pre-Patch 4 Architecture for Internet Deployment of Forms

Although most Forms deployment scenarios benefit from the HTTP connection mode of the Oracle Forms Server, there are some known shortcomings with the architecture:

Because the Forms Listener process manages the initial connections from the client, the machine on which the Forms Listener process is running must be exposed at the firewall level. In addition, the port that the Forms Listener process is listening to must also be exposed.

Once a client connection is handed to a Forms Server Runtime process, the client and the Forms Server Runtime process expect the connection to be persistent – that is, the network endpoints must be maintained. If the network connection at either end is dropped, the end user experiences a significant interruption and has to restart the application.

Because the data being passed uses HTTP, the Forms Listener/Server processes really become HTTP servers. Handling the slightly different HTTP formats sent by different browsers, proxies, and firewalls requires changes in the processes themselves.

## INTRODUCING THE FORMS6*i* LISTENER SERVLET

### What is the Forms Listener Servlet?

The Forms Listener Servlet is a Java servlet introduced in Forms6*i* patch 4 that improves upon the functionality of the Forms Listener.

**Note:** It is recommended that you use the Forms Listener Servlet when deploying applications using HTTP and HTTPS. The pre-Patch 4 Forms Listener is still available for direct socket connections, and still supports HTTP and HTTPS connections.

The Forms Listener Servlet requires the Oracle9*i* Application Server. The Forms Listener Servlet manages:

- The creation of a Forms Server Runtime process for each client
- Network communications between the client and its associated Forms Server Runtime process

In this scenario, the client sends HTTP requests and receives HTTP responses from the web server process. Because the web server acts as the network endpoint for the client, the other server machines and ports are no longer exposed at the firewall, as shown in the following figure.

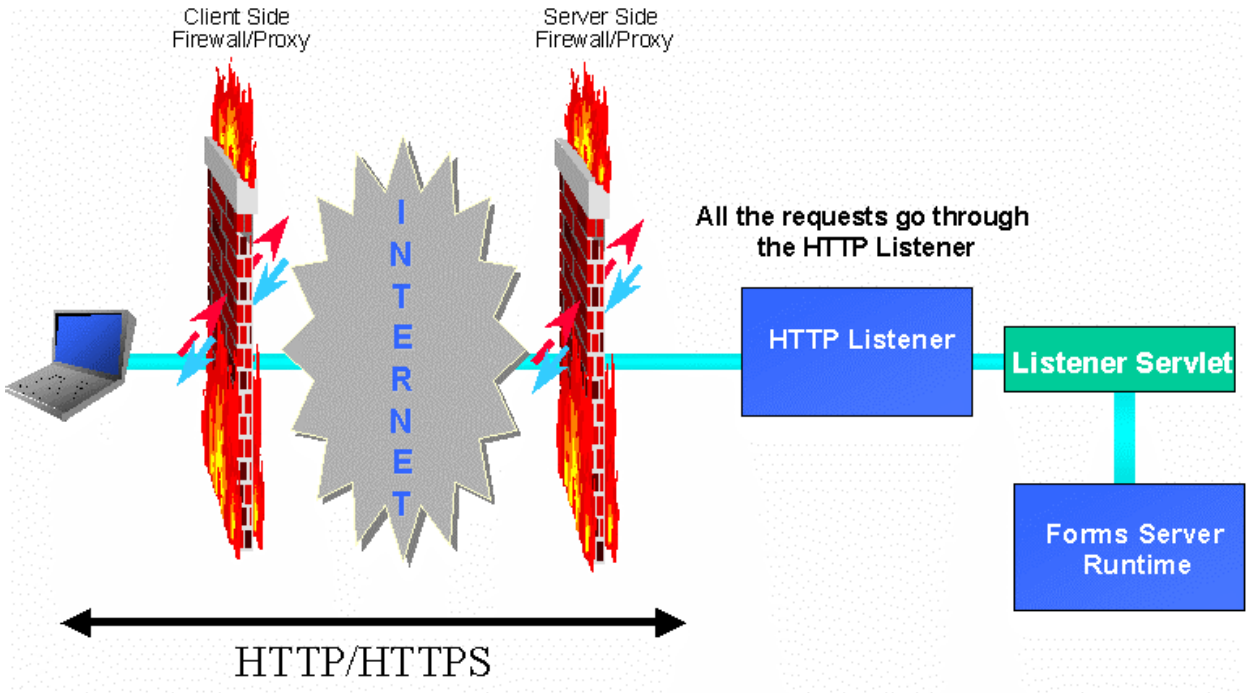



**Figure 2. New architecture using the Forms Listener Servlet**

## Why Should I Use the Forms Listener Servlet?

The Forms6*i* Listener Servlet was designed to allow a more robust and standard deployment of Forms applications on the Internet.

When compared to the Forms Listener, the Forms6*i* Listener Servlet provides the following benefits:

### Broader range of firewalls and proxies supported

Because the client browser always communicates with the web server using HTTP or HTTPS (there is no direct connection between the client and the Forms Server Runtime process), this architecture supports any firewall or proxy that can work with a standard servlet using servlet sessions.

### No protocol restriction (HTTP/1.1 or HTTP/1.0)

Although the use of HTTP/1.1-compliant proxies provides better performance, this architecture works well with HTTP/1.0-compliant proxies, too.

### No extra process to manage

Because this architecture eliminates the need for the Forms Listener process, the administrative tasks to start and stop the Forms Listener process are also no longer required.

### No specific certificate to purchase/manage for SSL deployment

In the case of deployment using SSL (secure sockets layer), the HTTPS connection occurs between the client browser and web server. Therefore, there are no specific security configuration requirements at the Forms Server level.

### Standard load balancing support

This architecture allows you to use standard load balancing techniques, such as hardware based load balancing, reverse proxy, and standard Apache JServ load balancing. (More information is available later in this document.)

### Internet Explorer 5.x with native JVM support

In addition to working with Oracle JInitiator, this architecture supports the use of Internet Explorer 5.x with native Microsoft JVM for Internet deployment using HTTP and HTTPS connection modes.

The Forms6*i* Listener Servlet **does not** support the following:

### Support from Oracle Enterprise Manager

Because the Forms Listener is no longer part of the architecture, the Forms Listener Servlet cannot be managed through the Oracle Enterprise Manager console. However, this functionality will be added in a future release.

### Forms-specific load balancing

The Forms Listener Servlet does not use Forms-specific load balancing (Load Balancer Server and Load Balancer Client). However, it supports standard load balancing methods.

## What is new in Forms6*i* Patch 10?

Time zone support. Oracle DATE values in Forms6*i* applications can be adjusted from one time zone to another. See "Time Zone Support" later in this white paper.

Cancelling a Query in a Long List LOV. Forms allows users to cancel a query of LOVs for any reason, instead of having to wait until the query is complete. See "Cancelling a Query in a Long List LOV" later in this white paper.

## What was new in Forms6*i* Patch 9?

Automatic Browser Language Detection. Forms architecture supports deployment in multiple languages. The automatic browser language detection feature allows you to automatically select the appropriate configuration to match a user's preferred language. In this way, all users can run Forms applications using the same URL, yet have the application run in their preferred language.

For more information, see "Automatic Browser Language Detection" later in this white paper.

Forms Listener Servlet has been enhanced to handle long queries. With patch 9, long operations, such as database queries which take longer than 15 minutes, are now supported in the runtime process. Previously, the servlet session would time out (typically in 15 minutes) resulting in complete loss of the Forms session.

For more information, see "Handling Long Queries" later in this white paper.

Oracle Forms Listener Servlet for Deployment of Forms on the Internet. Patch 9 includes a version of Oracle JInitiator that has full JDK 1.3 support. Oracle JInitiator allows enterprise developers to develop and deploy Oracle9*i*AS Forms Services applications, taking full advantage of JDK 1.3 features and functionality. Earlier patch releases have only included a JInitiator version with JDK 1.1 support.

For more information, read the Readme file shipped with JInitiator version 1.3.1.5.

### What was new in Forms6*i* Patch 8?

The environment configuration file and the current working directory for the runtime process can be set in the formsweb.cfg file (the FormsServlet configuration file). This will make it easier to have multiple configurations with different working directories and different sets of environment variables. For more information, see "Configuration Enhancements".

The values of the baseHTML and envFile parameters can be given as simple file names with no path (for example, `envFile=default.env`). In this case, we will look for the file in the same directory as the Forms Servlet configuration file (formsweb.cfg). For more information, see "Configuration Enhancements".

Particular PATH or LD_LIBRARY_PATH settings will not be required when starting the servlet engine. For more information, see "Configuration Enhancements".

FormsServlet baseHTML initialization parameters no longer required. For more information, see "Configuration Enhancements".

Modified, simpler configuration process. For more information, see "Basic Configuration" and "Advanced Configuration."

### What was new in Forms6*i* Patch 7?

The userid parameter value is not exposed in the HTML generated by the Forms Servlet. See "Hide User/Password" later in this white paper.

The default Forms userid format can be changed by only modifying the formsweb.cfg file.

### What was new in Forms6*i* Patch 6?

Improved performance when running Forms applications under JInitator in HTTPS mode using the Listner Servlet. See "End-User (Web Browser) Requirements" later in this white paper.

No longer uses fixed port numbers. See "Notes Regarding Ports" later in this white paper.

## INSTALLING THE FORMS LISTENER SERVLET

The Forms Listener Servlet is installed as part of Forms Patch 4 and above.

If you install the patch on top of an existing version of iAS, you will need to manually configure the Forms Listener Servlet. (Install this patch in 806 Oracle Home. Install only the Forms Server components using a custom install. Do not install Forms Builder.)

## BASIC CONFIGURATION

Location of the configuration files

| | |
|---|---|
| Jserv.properties | <Oracle_home>/apache/jserv/conf |
| Jserv.conf | <Oracle_home>/apache/jserv/conf |
| zone.properties | <Oracle_home>/apache/jserv/servlets |
| default.env | <Forms Oracle_home>/forms60/server |
| httpd.conf | <Oracle_home>/apache/apache/conf |
| Formsweb.cfg | <Forms Oracle_home>/forms60/server |

### Getting Started

The Forms Listener Servlet creates the Forms Server Runtime process (ifweb60 or f60webm) for each active Forms session and stops the process when the session ends. The environment variables required for a Forms Server Runtime process (for example, PATH, ORACLE_HOME, FORMS60_PATH) can be defined in an environment file. Any environment variables not defined in that file are inherited from the servlet engine (JServ).

**Note:** On NT, Forms reads Oracle environment settings from the registry unless they are set as environment variables.

### Pre-configuration requirement:

Forms Listener Servlet classes (<Forms Oracle_home>/forms60/java/f60srv.jar) must be available in the Java classpath of the servlet engine.

### Step 1: Configure servlet engine classpath for the Forms Listener Servlet

The CLASSPATH environment setting is configured in the **Jserv.properties** file using the wrapper.classpath directive.

**Note:** The following examples are not complete Jserv.properties files. Only lines relevant to Oracle Forms are included. In the examples, `d:\oracle\806` is the Forms Oracle Home, and `d:\oracle\isuites` is the Oracle9*i*AS Oracle Home on NT. On Solaris, `/private2/oracle/806` is the Forms Oracle Home, and `/private2/oracle/isuites` is the Oracle9*i*AS Oracle Home.

**Example JServ.properties file for NT**

```
wrapper.classpath=d:\oracle\806\forms60\java\f60srv.jar
```

**Example JServ.properties file for Solaris**

```
wrapper.classpath=/private2/oracle/806/forms60/java/f60srv.jar
```

### Step 2: Set the FormsServlet initialization parameter configFileName

Set configFileName to give the full path name of a valid servlet configuration file. Usually the servlet configuration file is the formsweb.cfg file in the forms60/server directory under ORACLE_HOME.

**Example from JServ zone.properties file**

```
servlet.f60servlet.code=oracle.forms.servlet.FormsServlet

servlet.f60servlet.initArgs=configFileName=d:\orant\forms60\server\for
msweb.cfg
```

### Step 3: Create or edit the environment file (which usually is default.env)

This file contains environment settings for Forms runtime, which is usually default.env (in the same directory as the formsweb.cfg file). On NT, registry settings are used for environment variables that are not set in the default.env file. On Solaris, this file should include the PATH and LD_LIBRARY_PATH.

Make sure the ORACLE_HOME setting is present and correct, and Forms60_PATH is set to include the directories containing your application (fmx files).

**Note:** For applications using run_product or run_report_object to run Reports or Graphics modules, set the appropriate Reports or Graphics environment variables in the default.env file (or other environment file).

**Example default.env file for NT**

```
ORACLE_HOME=d:\oracle\806

PATH=d:\oracle\806\bin

FORM60_PATH=d:\oracle\806\forms60
```

**Example default.env file for Solaris**

```
ORACLE_HOME=/private2/oracle/806

PATH=/private2/oracle/806/bin

LD_LIBRARY_PATH=/private2/oracle/806/lib:/private2/oracle/806/network/
jre11/lib/sparc/native_threads

# Path to application modules:

FORMS60_PATH=/private2/oracle/806/forms60
```

```
# For Forms applications which call Reports or Graphics modules:

REPORTS60_PATH=/private2/oracle/806/reports60
GRAPHICS60_PATH=/private2/oracle/806/graphics60

PRINTER=myprinter

# Physical and virtual locations and format for Oracle Reports output:
TMPDIR=
FORMS60_OUTPUT=/private/app/oracle/product/ias1021/6iserver/tools/web6
0/temp

FORMS60_MAPPING=http://cbarrow-sun.us.oracle.com:80/dev60temp
FORMS60_REPFORMAT=html
```

### Step 4: Create or edit required settings in the Forms Servlet configuration file (usually called formsweb.cfg)

Make sure the EnvFile parameter specifies the physical path to the environment file that contains environment variable settings.

The three baseHTML parameters should also be set to point to appropriate files (they no longer need to be set as servlet parameters). Typically, lines like the following should appear in the default section at the start of formsweb.cfg file:

```
baseHTML=base.htm
baseHTMLie=baseie.htm
baseHTMLJinitiator=basejini.htm
envfile=default.env
```

All four of the previous parameters give file names. If no paths are given (as in this example), the files are assumed to be in the same directory as the Forms Servlet configuration file (formsweb.cfg).

### Step 5: Add the Applet parameter serverURL to the Formsweb.cfg file

When using the Forms Listener, the Forms Java client connects to the Forms Listener using the values provided in the serverHost and serverPort applet parameters. However, when using the Forms Listener Servlet, you need to provide a value for a parameter, **serverURL**.

The serverURL parameter specifies the URL to access the Forms Listener Servlet. You must specify it as a relative URL – relative to the web server from which the HTML page containing the Forms applet tag was loaded.

A typical value is /servlet/**oracle.forms.servlet.ListenerServlet.**

This value works with Oracle9*i*AS and standard Apache/JServ installations. The part in bold is the class name for the servlet. The part before the class name is the path required to execute any servlet class, which depends on the servlet engine settings.

**Note:** If serverURL is specified (and if serverURL is not an empty string), then the Forms Listener Servlet is used. The applet parameter connectMode is ignored. Instead, the connection protocol is determined by the serverURL value (if it is a full URL with the protocol) or by the protocol used to access the page, (http:// or https://).

### Step 6: Determine whether to run JServ in Auto-Start mode

Depending on the number of concurrent users that you expect, decide whether to start JServ automatically or manually:

If the number of concurrent users will be less than 100, you can run JServ in Auto-Start mode.

If the number of concurrent users will be more than 100 or if the JServ process(es) will run on a machine separate from the Apache web listener, your site administrator must explicitly start the JServ engine or engines. (See the "Load Balancing JServ" section of this document for details on how to set up and start JServ in manual mode.)

**To run JServ in Auto-Start mode:**

In the **jserv.conf** file (which is included into httpd.conf or httpds.conf), check that the following parameter is set to "off":

```
ApJServManual off
```

**Note:** The ApJServManual parameter is set to "off" by default. When set to "off", JServ runs in automatic mode, that is, a single JServ process is used and is automatically started and stopped by the Apache Web Listener.

### Step 7: Use the Forms Servlet to start the application

The configuration steps described above allow you to run Forms applications using the FormsServlet by going to a URL such as the following:

http://myHost.myDomain/servlet/f60servlet?form=myForm

This is the recommended way of running Forms applications. However, it is also possible to use static HTML files, as described in **Configuration Using Static HTML Pages**.

## ADVANCED CONFIGURATION

### Setting Environment Variables for Specific Runtime Processes

In "Basic Configuration" (see above), a single environment file **default.env**, is used for all users. This is specified in the default (initial) section of the formsweb.cfg file. However, it is possible to specify different environment files in specific configuration sections.

#### Example

1. Create an environment configuration file, for example HumanRes.env, that contains the alternate environment variable settings using the following syntax:

HumanRes.env file

```
# comment

     myenvvar1=val1

     myenvvar2=val2

# comment
```

**Note:** Lines that are preceded by a pound sign (#) are assumed to be comments and are ignored. Lines that are not preceded by a #, but do not contain an equal sign (=) are also ignored. In the following example, the line "export myvar" is also ignored:

```
# comment
   myenvvar1=val1
   myenvvar2=val2
   export myvar
 # comment
```

**A sample environment configuration file, for example HumanRes.env, might contain the following:**

```
# French configuration

NLS_LANG=French_France

FORMS60_PATH=d:\french

PATH=d:\orant\bin
```

1. In the formsweb.cfg file, add a new section pointing to the environment file. For example:

```
[HR]

envFile = HumanRes.env

workingDirectory = /private/apps/hr
```

2. Specify the new configuration when running the application. For example:

   https://myHost.myDomain/servlet/f60servlet?config=HR

### Setting the Current Working Directory for Specific Runtime Processes

The current working directory can be set in the formsweb.cfg file, either in the default (initial) section or in a specific configuration section, by using the Working Directory parameter. For example:

```
[HR]
envFile = HumanRes.env
workingDirectory = /private/apps/hr
```

**Note:** If the WorkingDirectory parameter is not set, the working directory defaults to $ORACLE_HOME/forms60 on UNIX, or %ORACLE_HOME%\forms60 on Windows.

### User-written baseHTML Files

The default baseHTML files installed by Patch 9 (base.htm, basejini.htm, and baseie.htm) contain the applet parameter serverURL.

The value for serverURL is set in the formsweb.cfg file, as described in "Basic Configuration". The default baseHTML files are adequate for most purposes. However, you can create your own if the need arises, using the defaults as a basis. You must then edit the formsweb.cfg file to point to your new files (change the baseHTML, baseHTMLie, and baseHTMLJinitiator settings to name your files).

### Configuration Using Static HTML Pages

If you are using static HTML pages, add the serverURL applet parameter, and remove the serverPort and serverHost parameters. In the following example, the lines in **bold** are changed:

```
<APPLET CODEBASE="/forms60/java/"
         CODE="oracle.forms.engine.Main"
         ARCHIVE="f60all.jar"
         WIDTH="800"
         HEIGHT="600">
<PARAM NAME="serverURL"
VALUE="/servlet/oracle.forms.servlet.ListenerServlet">
<PARAM NAME="lookAndFeel"  VALUE="Generic">
```

```
</APPLET>
```

Since the Forms Servlet is not used to generate the HTML when using static pages, the formsweb.cfg file is not used. So the environment file must be specified as a ListenerServlet parameter, along with the working directory.

**Example zone.properties file for NT**

```
servlet.oracle.forms.servlet.ListenerServlet.initArgs=EnvFile=d:\oracl
e\806\forms60\server\default.env,workingDirectory=e:\apps\hr
```

### Setting the Runtime Executable Name for Specific Runtime Processes

You can specify the name of the runtime executable file, which may be useful for applications with user-exits.

**Note:** If the runtime executable file name is not specified, the default is used -- ifweb60.exe on NT or f60webm on UNIX.

You can specify the runtime executable for a specific runtime instance using the listener servlet initialization argument **Executable** as shown:

1. In the zone.properties file, create a ListenerServlet alias that will use the alternate runtime executable:

   ```
   # ListenerServlet alias for a different runtime executable:

   servlet.lservletHR.code=oracle.forms.servlet.ListenerServlet
   ```

2. In the zone.properties file, set the **Executable** parameter for the ListenerServlet alias. The Executable parameter specifies the physical path to the executable that is to be used:

   ```
   servlet.lservletHR.initArgs=Executable=d:\orant\bin\ifweb60x.exe
   ```

3. In the formsweb.cfg file, set the **serverURL** parameter for the ListenerServlet alias.

   ```
   [HR]

   serverURL=/servlet/lservletHR
   ```

   **A sample formsweb.cfg file might contain the following:**

   ```
   ; Default serverURL value (env. vars. in default.env will be used)

   serverURL=/servlet/oracle.forms.servlet.ListenerServlet


   ; Config section causing Executable parameter to be used

   [HR]
   ```

```
serverURL=/servlet/lservletHR
```

### Specifying the Test Information Available on the Listener Servlet Home Page

In pre-Patch 5 releases, when you go to the listener servlet home page (http://myhost/servlet/oracle.forms.servlet.ListenerServlet), there is a variety of test options and information available, some of which you may consider sensitive.

To limit the information available on this page (for security purposes), set the new **TestMode** parameter to "false" in the **zone.properties** file. (The default setting is false.)

```
servlet.lservletHR.initArgs=TestMode=false
```

To display all of the information and options available, such as the hostname of the machine, set **TestMode** to "true". (Any value but "true" will turn off sensitive information.)

### Running Forms Applications on the Web Using an Authenticating Proxy

In Forms 6i Patch 5, support was added to run Forms applications on the web using an authenticating proxy. An authenticating proxy is one that requires the user to supply a username and password in order to access the destination server where the application is running. Typically, authenticating proxies detect whether the user has logged on (i.e. been authenticated) by setting a cookie. The cookie is sent in all subsequent network requests to avoid further logon prompts.

To run Forms applications using an authenticating proxy, Forms 6i patch 5 (or later) must be installed, and you must be running the Listener Servlet (rather than the Forms Listener).

If users are running Internet Explorer 5.0 or higher with the native Microsoft Java VM or Internet Explorer 4.0 or higher with JInitiator, then no other configuration is required.

*However, if users are running Netscape with JInitiator, then you need to perform additional configuration steps.* These steps are necessary to ensure that the authentication cookie gets sent with all requests to the server. The basic requirement is that every URL that JInitiator has to access (those for the JAR files AND that for the Listener Servlet) MUST be under the document base of the HTML page. This is achieved by using the Forms Servlet to generate the page and by invoking the Listener Servlet under the /forms60java path by mapping a file extension to it. The Listener Servlet is accessed under that path by mapping /forms60java/servlet to the servlet zone.

If you have users running Netscape with JInitiator, do the following:

**Note:** The following steps assume the web server and servlet engine are Apache and JServ (as supplied with Oracle9*i*AS), and that the Forms Servlet is running using the servlet alias "f60servlet".

4. Stop Apache JServ.
5. Edit the jserv.conf file, and add the following lines (after the existing ApJServMount lines):

   ApJServMount /forms60java/servlet /root

   ApJServAction .f60 /servlet/f60servlet
6. Edit the formsweb.cfg file, and use the following serverURL setting under the config section that is being used (or alter the default setting):

   serverURL=/forms60java/servlet/oracle.forms.servlet.ListenerServlet
7. Restart Apache/JServ.
8. Access the Forms application (the page where the form runs) using a URL like:

   https://theserver.thedomain.com/forms60java/aname.f60?config=myconfig

   where aname can be any name (for example, forms or fred). Because the file name ends in ".f60" this request is routed to the Forms Servlet (f60servlet).

   **Note:** You do not have to use https, as in the example above. You can also use http.
9. Log on to the authenticating proxy when prompted.

### Configuring Session-Level Logging

To write session-level log messages to the JServ log file (such as the true client IP address and the process ID of the associated Forms runtime process), append "**/session**" to the serverURL client parameter, for example:

```
http://yourserver/servlet/f60servlet?config=servlet&serverURL=/servlet
/oracle.forms.servlet.ListenerServlet/session
```

A sample session-level log entry follows:

```
[28/03/2001 13:54:26:083 PST] Forms session <1> process id 310
started for

admin-pc.us.acompany.com (140.74.96.38)

[28/03/2001 13:54:41:625 PST] Forms session <1> ended

[28/03/2001 13:55:01:073 PST] Forms session <2> process id 367
started for

jdoe-pc.us.acompany.com (140.74.96.71)

[28/03/2001 13:55:27:061 PST] Forms session <3> process id 300
started for

admin-pc.us.acompany.com (140.74.96.38)

[28/03/2001 13:56:43:080 PST] Forms session <2> ended

[28/03/2001 13:56:47:647 PST] Forms session <3> ended
```

Other logging options are also available (see TROUBLESHOOTING).

## AUTOMATIC BROWSER LANGUAGE DETECTION

With Forms6*i* patch 9, the preferred language settings in the client browser is automatically detected. Automatic browser detection enables the management of a Forms application deployed in multiple languages and accessible from the same application URL. When a user accesses the application URL, the Forms application automatically displays in the user's preferred language.

In most client browsers, there is a preference setting that lets users choose one or a sequence of preferred languages. If there are multiple preferred languages configured, Forms Services detects a matching language from the list and switches to that configuration.

### Configuring Language Detection

There are two ways you can specify language detection in Forms Services:

1. Language-dependent configurations are specified in the Forms configuration file, formsweb.cfg. You should name the language-specific configuration sections in the following way:

   `[<config_name>.<language_code>]`

   where <config_name> is the configuration section for which the language-detection feature is sought and <language_code> is the standard code representing the language for which settings are defined.

2. The default or system section may also have language-dependent sections. For example, this section could be configured as follows without a <config_name>:

   [.<language_code>]

   For example, if you created a configuration section named "hr", and wanted to create French and Chinese languages, your configuration section might look like the following:

```
[hr]
lookAndFeel=oracle
width=600
height=500
envFile=default.env
workingDirectory=/private/apps/hr

[hr.fr]

envFile=french.env
workingDirectory=/private/apps/hr/french

[hr.zh]
envFile=chinese.env
workingDirectory=/private/apps/hr/chinese
```

#### Inheritance in the Configurations

To avoid duplication of common values across all language-specific variants of a given base configuration, Oracle recommends that you only define parameters which are language-specific in the language-specific sections. While creating language-specific configurations, it is not necessary to duplicate parameters. Instead, the parameters common to all the languages can be specified in the base configuration section which is the configuration section without the language code. All the language-specific sections will then inherit the parameters from the base configuration, language default sections, and the default configuration in the `formsweb.cfg` file.

#### How Language Detection Works

When the Forms Servlet receives a request for a particular configuration (for example, http:// myserv/servlet/f60servlet?config=hr), it gets the client language setting from the request header "accept-language" which provides a list of languages in the preferred order. For example, if accept-language is: de,fr,en_us, then the order of preference is German, French, and US English. The Forms Servlet will look for a language-specific configuration section matching the first language. If it is not found, the servlet looks for the next language configuration and so on. If no language-specific configuration is found, it will use the base configuration.

When the FormsServlet receives a request with no particular configuration specified or no "config=" URL parameter, for example, http://myserv/servlet/f60servlet), the servlet will look for a language-specific section in the default section matching the first language.

#### Multi-level Inheritance

Four levels of inheritance are available and the servlet will look for a language-specific configuration section. For example, if a particular configuration is requested, using a URL query parameter like `config=myconfig`, the servlet searches for the value for each parameter in the following order:

1. Language-specific configuration section which best matches the user's browser language settings, for example, [myconfig.fr]
2. Base configuration section, for example, [myconfig]
3. Language-specific default configuration section, for example, [.fr]
4. Default configuration section of the file

Typically, the parameters which are most likely to vary from one language to another are "workingDirectory" and "envFile". Using a different envFile setting for each language lets you have different values of `NLS_LANG` (to allow for different character sets and data and number formats) and `FORMS60_PATH` (to pick up

language-specific fmx files). Using different "workingDirectory" settings provide another way to pick up language-specific fmx files.

#### Language-specific Configuration Steps

To configure the language-specific configuration sections, do the following:

1. Create language-specific configuration sections in the `formsweb.cfg` file for a set of languages. For example:

```
; default language sections for US English, French and
German languages
      [.en-US]
      ....
      ....
      [.fr]
      ....
      ....
      [.de]
      ....
      ....
      ; Base configuration section
      [app]
      ....
      ....
      ; Configuration section for US English
      [app.en-us]
      ....
      ....
      ; Configuration section for French
      [app.fr]
      ....
      ....
      ; Configuration section for German
      [app.de]
      ....
```

. . . .

2. Set the parameters in these language-specific configuration sections. With the exception of their special naming convention and meaning, these language-specific configuration sections are no different from other configuration sections in the `formsweb.cfg` file. There are no other restrictions to the parameters that can be included in these sections. However, the following guidelines are considered as best practices.

   The parameters that are generic and common to all the forms and all the languages should be set in the system or default configuration section.

   The parameters that are generic to all the forms but are language-specific should be set in language default section.

   The parameters that are generic to the language but specific to a particular form should be set in the default configuration section for that particular form.

   The parameters that are specific to both a form and a particular language should be set in the language configuration section of that form.

   Always create a language-specific configuration section for all the languages supported by a form even if there is no specific configuration parameters. In other words, create a blank or empty configuration section. For example if the "HR" application supports the English (US) ,Chinese, and German languages, Oracle recommends that you create all the following configuration sections.

   3. [HR]
   4. [HR.en-US]
   5. [HR.de]
   6. [HR.zh]

**7.** Set up browser language preferences. The language preference can be modified in the browser, and prompting it to generate headers making requests for preferred languages.

The steps for setting language preferences in Internet Explorer and Netscape are provided in the appropriate section below:

***Internet Explorer***

1. Choose the **Tools | Internet Options…** menu.
2. Click the **Languages**… button. The Language Preference dialog appears listing the current language preferences.

3. Click **Add…** to add more languages to the preference list.
4. Use the **Move Up** and **Move Down** buttons to change the preference order. The first language at the top of the list is the most preferred language.
5. Click **OK** to accept the settings.

***Netscape Navigator***

1. Choose the **Edit | Preferences…** menu.
2. Click **Languages** in the Navigation tree.
3. Click **Add** to add more languages to the preference list.
4. Use the arrow buttons to change the preference order. The first language at the top of the list is the most preferred language.

## HANDLING LONG QUERIES

With Patch 9, the Listener Servlet and Forms client have been enhanced to handle long operations, such as database queries which take longer than 15 minutes, are now supported in the runtime process. Previously, the servlet session would time out (typically in 15 minutes) resulting in complete loss of the Forms session.

Testing had revealed that when using the Listener Servlet any request to the server which is very slow (e.g. a long running query) caused the servlet session to time out, which terminated the Forms session. This timeout occurred when the Forms runtime engine took longer to respond to the request than the configured servlet engine session timeout. The default session timeout in iAS (Apache JServ) is 15 minutes. This may seem long, but Applications users often need to do long queries, and expect them to come back with a response.

The reason for the behavior is that during a long request the client is blocked waiting for the response and therefore does not send its normal heartbeat messages (which are usually sent every 2 minutes). The heartbeat keeps the servlet session alive when the client is waiting for input from the user, but it is inoperative while the client is waiting for the server to respond.

Long requests are now handled without requiring the servlet session timeout to be increased.

## CONFIGURATION ENHANCEMENTS

### Support envFile and workingDirectory parameters in formsweb.cfg

With Forms6i Patch 5, environment variables and the current working directory became configurable via two Listener Servlet initialization parameters: envFile and workingDirectory. To achieve multiple configurations you were required to define multiple aliases for the ListnerServlet. Also, you had to configure both FormsServlet and ListenerServlet initialization parameters.

Starting with Patch 8, the name of the environment configuration file and the current working directory for the runtime process can be set in the formsweb.cfg file (the FormsServlet configuration file). This will make it easier to have multiple configurations with different working directories and different sets of environment variables.

If there are values for the envFile and/or workingDirectory parameters in the formsweb.cfg file (in the default section or configuration section), those parameters will be used when the runtime process is started.

For backwards compatibility with previous releases, if neither the envFile nor workingDirectory parameter is given in the formsweb.cfg file, or the values are empty, the ListenerServlet initialization parameters ("envFile" and "workingDirectory") will be used instead, if they are specified.

An empty value for the workingDirectory will default to <ORACLE_HOME>/forms60,

where <ORACLE_HOME> is the value of the ORACLE_HOME environment variable (obtained from the envFile, or from the calling environment if it is not given in the envFile or there is no envFile specified).

### Look for baseHTML files and envFile in same directory as formsweb.cfg

BaseHTML files and envFile parameters can be given simple file names with no path (for example, `envFile=default.env`). In cases where the path is not identified in the name, Forms will look for the baseHTML and envFile in the same directory as the Forms Servlet configuration file (formsweb.cfg ).

### Alteration in the default formsweb.cfg file

The following values in the default Forms Servlet configuration file (formsweb.cfg), which is installed when Forms is first installed, will be altered:

```
baseHTML=base.htm
baseHTMLJInitiator=basejini.htm
baseHTMLIE=baseie.htm
envFile=default.env
workingDirectory=
```

### Example formsweb.cfg file showing how the new functionality can be used (new functionalities are in bold face)

```
baseHTML=base.htm
baseHTMLJInitiator=basejini.htm
baseHTMLIE=baseie.htm
envFile=default.env
;workingDirectory will default to ORACLE_HOME/forms60 if unset:
workingDirectory=
userid=
```

```
[HRapp]
envFile=d:\apps\hr\hr.env
workingDirectory=d:\apps\hr
userid=%user%/%password%@%database% (new SSO facility)
```

### Avoid need to configure PATH (or LD_LIBRARY_PATH) for the servlet engine

Particular PATH or LD_LIBRARY_PATH settings will not be required when starting the servlet engine. If the native methods library (ifjsl60.exe on Windows or libifjsl60.so on UNIX) fails to load on the first attempt, a second attempt will be made. In the second attempt the Forms Servlet engine will look for the library in <ORACLE_HOME>/bin (Windows) or <ORACLE_HOME>/lib (UNIX), where <ORACLE_HOME> refers to the ORACLE_HOME environment setting as specified by the environment configuration file (which is default.env by default).

### FormsServlet baseHTM initialization parameters no longer required

The baseHTML, baseHTMLie, and baseHTMLJInitiator Forms Servlet parameters are no longer required. Instead, they should be specified in the default (initial) section of the Forms Servlet configuration file (formsweb.cfg). For compatibility with previous releases, if the baseHTML servlet parameters *are* specified, then those values will be used in preference to any values given in the default section of formsweb.cfg.

## END-USER (WEB BROWSER) REQUIREMENTS

Supported Web browsers and configurations are similar to Oracle Forms6*i*, except for the following additional requirements:

**JInitiator version 1.1.8.2 or above must be used.** If Netscape Navigator or Internet Explorer is being used with Oracle JInitiator as the JVM, then you must upgrade to JInitiator 1.1.8.2 or higher. (JInitiator 1.1.8.11 is supplied with this patch.) If you attempt to run an Oracle Forms application with the Oracle Forms Listener Servlet using a previous version of JInitiator, an error message explaining the problem is displayed. If you are using Netscape with JInitiator and you want to run Forms applications on the Web using an authenticating proxy, then you must edit the jserv.conf and formsweb.cfg files, as described in "Running Forms Applications on the Web Using an Authenticating Proxy" on page 18 of this white paper.

**Improved performance when running Forms applications under JInitator in HTTPS mode using the Listener Servlet.** Previously, JInitiator's HTTPS implementation did not use HTTP keep-alive. Now that keep-alive has been implemented (available in JInitiator 1.1.8.11 and higher), the client does not have to reconnect every time it makes a URL request and, consequently, eliminates the need for an SSL handshake every

time the thin client communicates with the server. We therefore recommend that you upgrade to JInitiator 1.1.8.11 especially on high-latency networks.

**For this patch, session cookies are not required when using Internet Explorer with the native JVM.** Instead, URL rewriting, which is supported by most servlet engines (including the one with iAS), maintains servlet sessions.

## USING HTTPS WITH THE FORMS LISTENER SERVLET

Using HTTPS with the Forms Listener Servlet is no different than using HTTPS with any other web-based application.

### Server Requirements

HTTPS requires the use of digital certificates. Because the Forms Listener Servlet is accessed via your web server, you do not need to purchase special certificates for communications between the Forms client and the server. You only need to purchase a certificate for your web server from a recognized Certificate Authority.

### Client requirements

#### Using HTTPS with Internet Explorer and native JVM

If your end users are running Internet Explorer 5.0 or higher with native JVM (rather than Oracle JInitiator), then all that is required to access a page that contains the Forms applet tag is an https-style URL.

For example, if an application was being accessed in HTTP mode as follows (using Forms CGI):

```
http://myserver/dev60cgi/ifcgi60.exe?config=myapp or

http://myserver/servlet/f60servlet?config=myapp
```

then, users would request the following URL instead:

```
https://myserver/dev60cgi/ifcgi60.exe?config=myapp or

https://myserver/servlet/f60servlet?config=myapp
```

**Note:** The web server must be configured with an appropriate certificate to support HTTPS. If you are using the test certificate supplied with Oracle9iAS for test purposes, you will be prompted by Internet Explorer on whether or not to accept the certificate. This is because the demo root certificate authority is not a real one and will not be recognized by Internet Explorer.

As mentioned in the configuration section, the serverURL value set in the formsweb.cfg file should use a relative URL, for example:

serverURL=/servlet/oracle.forms.servlet.ListenerServlet

The Forms applet automatically accesses the Forms Listener Servlet using HTTPS by using a URL constructed from the document base combined with the serverURL setting, for example:

```
https://myserver/servlet/oracle.forms.servlet.ListenerServlet
```

#### Using HTTPS with Oracle JInitiator

If your end users are running Oracle JInitiator as the web browser JVM, then you need to check that the Root Certificate Authority of your web site's SSL certificate is one of those defined in the JInitiator **certdb.txt** file.

The certdb.txt file is usually found under c:\program files\oracle\jinitiator <version>\lib\security on the machine where JInitiator was installed.

**Note:** If you are using the test certificate supplied with Oracle9iAS for test purposes, you must edit the JInitiator certdb.txt file and append the contents of the demo root certificate, which is located in <9iAS oracle_home/Apache/Apache/conf/ssl.crt/demoCAcert.txt. Otherwise, you will get the following error when attempting to run a Form: javax.net.ssl.SSLException: SSL handshake failed:X509CertChainInvalidErr.

## TROUBLESHOOTING

### Ensure the Listener Servlet and native methods library is available

The following test checks that the JServ classpath and PATH settings (and LD_LIBRARY_PATH on Solaris) are correct.

1. Point your web browser to a URL like:
   ```
   http://your_server/servlet/oracle.forms.servlet.ListenerServlet
   ```
2. You should get a page titled "Forms6*i* Listener Servlet".
3. Click on the link "Test native method call (JNI)" to validate your configuration. You should not see any errors. If you do, the PATH or LD_LIBRARY_PATH settings are probably wrong, or the ifjsl60.dll (libifjsl60.so) file is not present in <Forms oracle_home>/bin (or <Forms oracle_home>/lib).

### Try to run the test form using the servlet

Point your web browser to a URL like:

```
http://your_server/servlet/f60servlet?config=servlet.
```

The test form should come up.

The formsweb.cfg file must have the following section in order for this test to work:

```
[servlet]
```

```
serverURL=/servlet/oracle.forms.servlet.ListenerServlet
```

### Debug and performance tracing

Trace messages are written to the servlet engine's log file (jserv.log) if "/debug" , "/sessionperf", or "/perf" is appended to the serverURL value.

### To write performance messages to the jserv.log file, do the following:

```
http://yourserver/servlet/f60servlet?config=servlet&serverURL=/servlet
/oracle.forms.servlet.ListenerServlet/perf
```

This causes a performance message to be written whenever a request from the client is processed, stating the time taken to process the request and the number of bytes of input and output. A summary giving the average performance for the session is written whenever a Forms session ends normally, for example, as the result of an exit_form call in the Forms application. If you only want the summary, use "/sessionperf".

### To write full debug messages to the jserv.log file, do the following:

```
http://your_server/servlet/f60servlet?config=servlet&serverURL=/servle
t/oracle.forms.servlet.ListenerServlet/debug
```

## PERFORMANCE/SCALABILITY TUNING

When using the Forms Listener Servlet, most tuning steps are those that would be appropriate for any high throughput servlet application.

### Limit the number of HTTPD processes

To avoid spawning too many HTTPD processes (which is memory consuming) set the following directive in the Apache configuration file (httpd.conf):

```
KeepAlive Off
```

If you must use KeepAlive On (for another application, for example), make sure that KeepAliveTimeout is set to a low number (for example, 15 seconds, which is the default).

### Set the maxClient directive to a High value

It is best to let Apache determine when to create more HTTPD daemons. Therefore, set the maxClient directive to a high value in the Apache configuration file (httpd.conf). However, you need to consider the memory available on the system when setting this parameter.

MaxClient=256 means that Apache can create up to 256 HTTPD processes to handle concurrent requests.

Based on our tests, about 30 HTTPD processes are created to handle 300 concurrent users. This value may vary depending on the usage of your application.

### Disable JServ logging

Logging impacts performance. It is best to reduce the generated log or even disable it in a production environment if performance is an issue. To do this, set the following parameter in the JServ configuration file (jserv.conf):

```
Set log=false
```

### Disable the JServ auto-reload

By default, every servlet repository is checked and all timestamps are compared for modification each time a servlet is executed. To avoid this, set the following parameters in the **zone.properties** file:

```
autoreload.classes=false

autoreload.file=false
```

### Run the HTTP listener and JServ engine(s) on different machines

Based on our tests, the only scalability difference between the Forms Listener Servlet architecture and the Forms Listener architecture is related to the HTTP listeners.

If the JServ engine(s) are running on a different machine than the HTTP listener, the two architectures have the same scalability.

See the Load Balancing section that follows for the configuration needed to run JServ on a machine other than the HTTP listener.

## LOAD BALANCING JSERV

The new Forms Listener Servlet architecture allows you to load balance the system using any of the standard load balancing techniques available.

Apache provides a built-in load balancing mechanism that allows you to run multiple JServ engines on different hosts. For a complete description of this feature, please refer to the Apache documentation available as part of Oracle9iAS at http://<server>:<port>/jservdocs/howto.load-balancing.html, where server and port are the server name where Oracle9iAS was installed and port is the port number of your Apache web listener (80 or 7777 depending on the version and the operating system).

In this section we look at two scenarios:

- How to balance incoming requests between two JServ engines on the same host as the Apache web listener
- How to balance incoming requests between two JServ engines on a different host than the Apache web listener

### Case 1: Two JServ engines on the same host as Apache web listener

Use this configuration if you are expecting more than 100 concurrent users, which is an approximate limit for one JServ engine without experiencing noticeable performance degradation, and if you have enough system resources to handle the HTTP listener load and the Forms Server load.

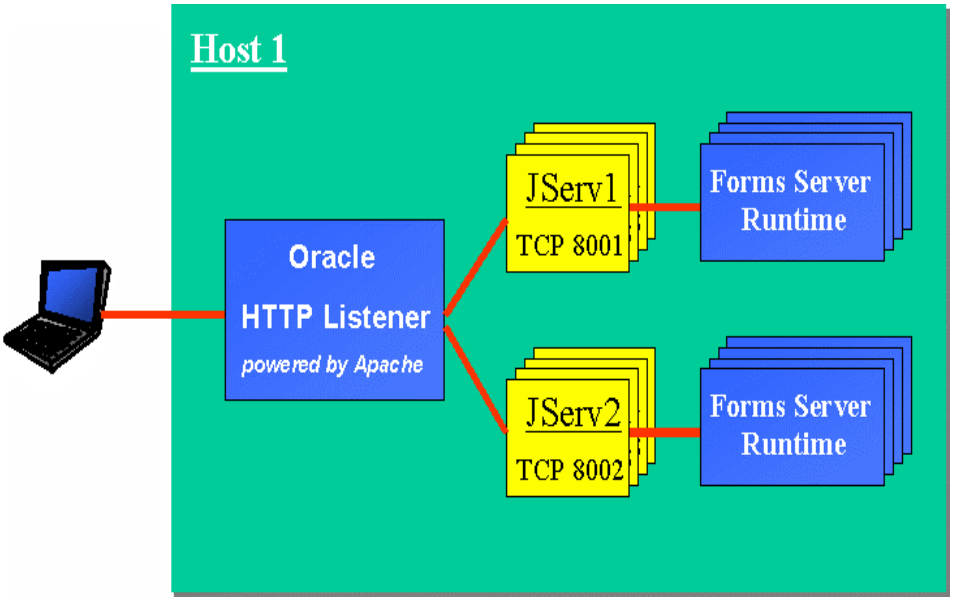



**Figure 3. Multiple JServ engines on one host**

#### Step 1: Configure the JServ engines

In order to balance the load among multiple JServ processes, the JServ engines must be configured to listen on different ports and to log to separate files.

If you have an existing jserv.properties file that contains all of the correct parameters to run your application:

1. Copy the jserv.properties file to two separate files, for example jserv1.properties and jserv2.properties.
2. Edit each of the files as follows:

**jserv1.properties**

```
port=8001

log.file=/usr/local/jserv/logs/jserv1.log
```

**jserv2.properties**

```
port=8002

log.file=/usr/local/jserv/logs/jserv2.log
```

**Note:** The settings used in the examples are based on a given port usage scheme and a specific log directory location. You must set these parameters according to your local conventions and operating system.

**Step 2: Modify the jserv.conf file to distribute the load**

1. When using multiple JServ engines, you cannot allow Apache to start the JServ processes automatically. Instead, the JServ processes have to be started manually. To do so, set the following parameter in the jserv.conf file:

```
ApJServManual on
```

2. Next, modify the lines that describe where to send servlet requests. In the jserv.conf file, you will see one or more lines starting with "ApJServMount". For example:

```
ApJServMount /servlet /root
```

This line specifies that if a request is made that starts with "http://your.server.com/servlets/", then the class file for the requested servlet can be found in the repositories for the "root" zone.

When only one JServ process is used, it is implicit that the process will service the zone. When you do load balancing, however, you must describe how the work is to be split among the available processes, and how they can be found. In this example, you would replace the line above, with the following:

```
ApJServMount /servlet balance://set/root

ApJServBalance set JServ1

ApJServBalance set JServ2

ApJServHost JServ1 ajpv12://127.0.0.1:8001

ApJServHost JServ2 ajpv12://127.0.0.1:8002

ApJServRoute JS1 JServ1

ApJServRoute JS2 JServ2
```

**ApJServMount** indicates where to send the requests for a servlet starting with ".../servlet/". It says to balance them among the JServ processes.

**ApJservBalance** defines which JServ engine to use. (jserv1.properties and jserv2.properties files contain the parameters for the engines.)

**ApJServHost** describes on which host and port these processes are listening. In our example, the two processes are running on the same host. (ajpv12 is the JServ communication protocol.)

**ApJServRoute** defines how the user sessions find their way back to the "right" JServ process. The ApJServRoute value is JS1 , JS2, and so on. This information is automatically used by the JServ session mechanism that sends the process route information back to the user (in a cookie).

#### Step 3: Create start and stop scripts

Finally, create the start and stop scripts for the JServ engines.

### The following are example scripts in a Solaris environment:

### Start.sh

```
#!/bin/sh

ORACLE_HOME=/u01/app/oracle/product/9ias/806
export ORACLE_HOME

IAS_HOME=/u01/app/oracle/product/9ias
export IAS_HOME

APACHE=${IAS_HOME}/Apache
export APACHE

APACHE_HOME=${APACHE}/Apache
export APACHE_HOME

JSERV_HOME=${APACHE}/Jserv
export JSERV_HOME

JSERV_CONF=${JSERV_HOME}/etc
export JSERV_CONF

JAVA_HOME=${APACHE}/jdk
export JAVA_HOME

CLASSPATH=${APACHE}/Jsdk/lib/jsdk.jar:${IAS_HOME}/jdbc/classes12.zip:$
{JSERV_HOME}/libexec/ApacheJServ.jar:${ORACLE_HOME}/forms60/java/f60sr
v.jar:${ORACLE_HOME}/forms60/java:
export CLASSPATH

JAVA=${JAVA_HOME}/bin/java
export JAVA

LD_LIBRARY_PATH=${ORACLE_HOME}/lib

PATH=${ORACLE_HOME}/bin:$PATH

FORMS60_PATH=${ORACLE_HOME}/forms60

export LD_LIBRARY_PATH PATH FORMS60_PATH

# now kick off jservs (listening on different ports)
```

```
nohup $JAVA -classpath $CLASSPATH org.apache.jserv.JServ
${JSERV_CONF}/jserv1.properties >>

${JSERV_HOME}/logs/jserv1.startup.log 2>&1 &

nohup $JAVA -classpath $CLASSPATH org.apache.jserv.JServ
${JSERV_CONF}/jserv2.properties >>

${JSERV_HOME}/logs/jserv2.startup.log 2>&1 &
```

## Stop.sh

```
#!/bin/sh

ORACLE_HOME=/u01/app/oracle/product/9ias/806
export ORACLE_HOME

IAS_HOME=/u01/app/oracle/product/9ias
export IAS_HOME

APACHE=${IAS_HOME}/Apache
export APACHE

APACHE_HOME=${APACHE}/Apache
export APACHE_HOME

JSERV_HOME=${APACHE}/Jserv
export JSERV_HOME

JSERV_CONF=${JSERV_HOME}/etc
export JSERV_CONF

JAVA_HOME=${APACHE}/jdk
export JAVA_HOME

CLASSPATH=${APACHE}/Jsdk/lib/jsdk.jar:${IAS_HOME}/jdbc/classes12.zip:$
{JSERV_HOME}/libexec/ApacheJServ.jar:${ORACLE_HOME}/forms60/java/f60sr
v.jar:${ORACLE_HOME}/forms60/java:
export CLASSPATH

JAVA=${JAVA_HOME}/bin/java
export JAVA

LD_LIBRARY_PATH=${ORACLE_HOME}/lib

PATH=${ORACLE_HOME}/bin:$PATH

export LD_LIBRARY_PATH PATH

# now stop jservs

nohup $JAVA -classpath $CLASSPATH org.apache.jserv.JServ
${JSERV_CONF}/jserv1.properties -s
```

```
nohup $JAVA -classpath $CLASSPATH org.apache.jserv.JServ
${JSERV_CONF}/jserv2.properties -s
```

### The following are example scripts in an NT environment:

### Start.bat (for 1 JServ engine)

```
Set FORMS60_PATH=c:\myapp

set properties1=D:\Oracle\iSuites\Apache\Jserv\conf\jserv1.properties

set log=D:\Oracle\iSuites\Apache\Apache\logs\jserv_manual

set
CLASSPATH=%CLASSPATH%;D:\Oracle\iSuites\Apache\Jserv\ApacheJServ.jar;D
:\Oracle\iSuites\Apache\Jsdk\lib\jsdk.jar;D:\Oracle\iSuites\jdbc\lib\c
lasses111.zip;D:\Oracle\iSuites\Apache\Apache\htdocs\_pages;D:\Oracle\
iSuites\Apache\Apache\htdocs\OnlineOrders_html;D:\Oracle\iSuites\Apach
e\Apache\htdocs\OnlineOrders_html\OnlineOrders.jar;D:\Oracle\iSuites\A
pache\BC4J\lib\connectionmanager.zip;D:\Oracle\806\forms60\java\f60srv
.jar;D:\Oracle\806\forms60\java

set JAVA=D:\Oracle\iSuites\Apache\jdk\bin\java

%JAVA% -classpath %CLASSPATH% org.apache.jserv.JServ %properties1% >>
%log%1.log
```

### Stop.bat (for 1 JServ engine)

```
set properties1=D:\Oracle\iSuites\Apache\Jserv\conf\jserv1.properties

set
CLASSPATH=%CLASSPATH%;D:\Oracle\iSuites\Apache\Jserv\ApacheJServ.jar;D
:\Oracle\iSuites\Apache\Jsdk\lib\jsdk.jar;D:\Oracle\iSuites\jdbc\lib\c
lasses111.zip;D:\Oracle\iSuites\Apache\Apache\htdocs\_pages;D:\Oracle\
iSuites\Apache\Apache\htdocs\OnlineOrders_html;D:\Oracle\iSuites\Apach
e\Apache\htdocs\OnlineOrders_html\OnlineOrders.jar;D:\Oracle\iSuites\A
pache\BC4J\lib\connectionmanager.zip;D:\Oracle\806\forms60\java\f60srv
.jar;D:\Oracle\806\forms60\java

set JAVA=D:\Oracle\iSuites\Apache\jdk\bin\java

%JAVA% -classpath %CLASSPATH% org.apache.jserv.JServ %properties1% -s
```

### Case 2: Two JServ engines on a host other than Apache web listener

Use this configuration if you are expecting more than 100 concurrent users, which is an approximate limit for one JServ engine without experiencing noticeable performance degradation, and if you do not have enough system resources to handle the HTTP listener load and the Forms Server load.

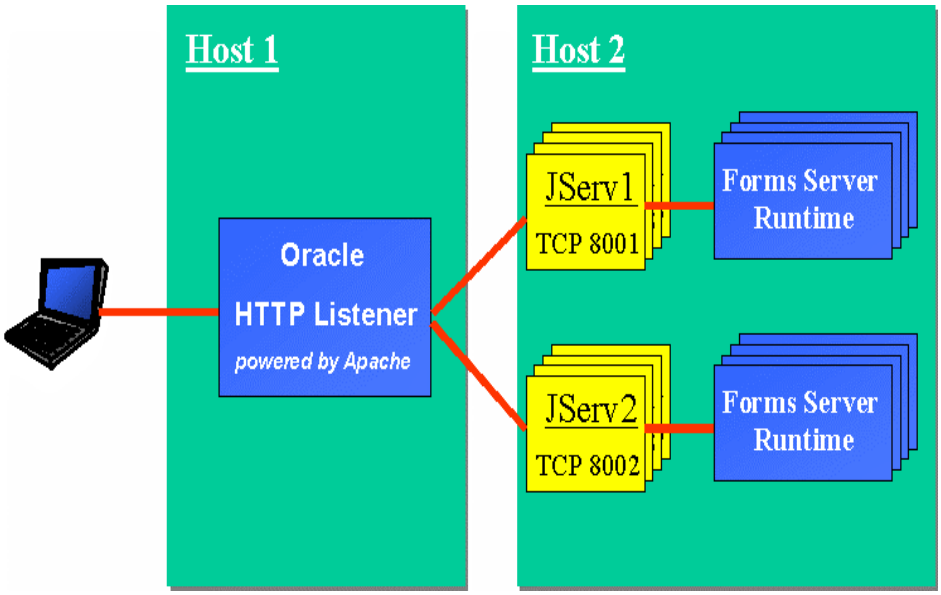



**Figure 4. Multiple JServ engines on multiple hosts**

#### Step 1: Configure the JServ engines on Host 2 (the one running JServ)

As in case 1 (multiple JServ engines on the same host as the web listener), you need to create and configure two jserv.properties files with specific ports and log files. See Case 1, Step 1 for details.

Then, in each of the jserv.properties file, define the name or the IP address of the machine where the JServ engine is running using the bindaddress parameter. Replace:

```
bindaddress=localhost
```

with

```
bindaddress=<name or ip address of the machine where JServ is running,
host2 in our example>
```

#### Step 2: Modify the JServ configuration file (jserv.conf) in Host1 (the one running the web listener) to define where the JServ engines are running

1. Be sure that jserv.conf and oracle_apache.conf are included in httpd.conf of the http server host  Make sure that the following lines are present:

   ```
   include "/private/oracle/Apache/Jserv/etc/jserv.conf"
   include "/private/oracle/Apache/Apache/conf/oracle_apache.conf"
   ```

2. As in case 1 (multiple JServ engines on the same host as the web listener), configure the jserv.conf file (on the http server machine). See Case 1, Step 2 for details. For example:

```
ApJServMount /servlet balance://set/root

ApJServBalance set JServ1

ApJServBalance set JServ2

ApJServHost JServ1 ajpv12://host2:8001

ApJServHost JServ2 ajpv12://host2:8002

ApJServRoute JS1 JServ1

ApJServRoute JS2 JServ2
```

   Be sure that the host name specified for the ApJServHost is the same as the one specified in the bindaddress parameter defined in the jserv.properties file.

#### Step 3 : On Solaris, load the Apache JServ communication module

On Solaris, make sure that the JServ communication module is loaded. Check for the following lines in httpd.conf:

```
LoadModule jserv_module $APACHE_HOME/Jserv/libexec/mod_jserv.so

AddModule mod_jserv.c
```

If these lines are not there, add them.

#### Step 4 : Start the JServ engines in the JServ hosts

Follow the steps in Case 1, Step 3 to start the multiple JServ engines in the JServ hosts.

## NOTES REGARDING PORTS

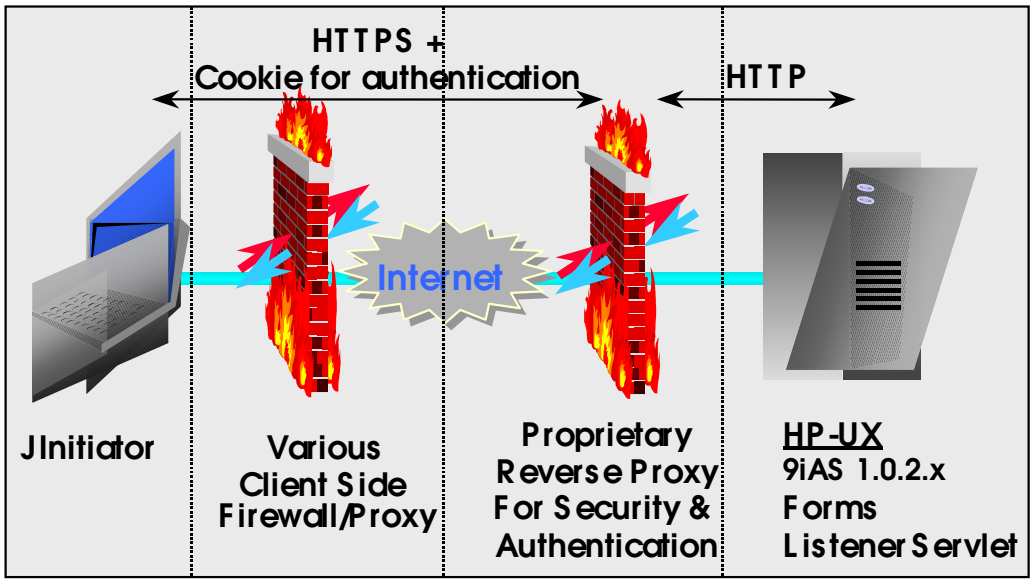
Previously, as a temporary workaround, the Forms Listener Servlet communicated with the Forms Server Runtime processes by using fixed port numbers. However, starting with Forms6*i* patch 6, fixed port numbers are no longer used. Consequently, the maxPorts and startPort servlet initialization parameters are obsolete and, if specified, will be ignored.

## EXAMPLE SYSTEM ARCHITECTURES USING AUTHENTICATION

The following three figures show examples of how authentication is accomplished in system architectures that include the Forms Listener Servlet.

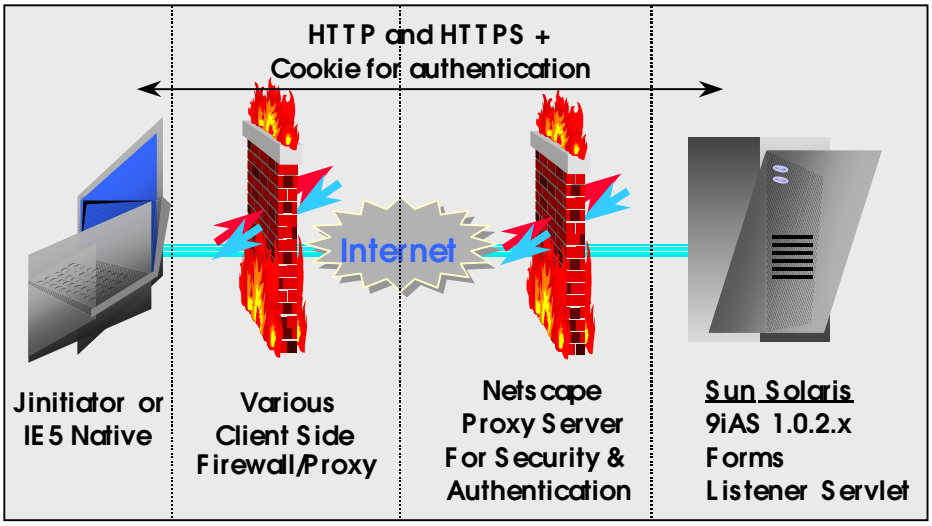
### Listener Servlet Using a Reverse Proxy (cookie-based authentication)

| | |
|---|---|
| **Server Platform** | HP-UX |
| **Protocol** | HTTPS between the client and reverse proxy for internet communication<br>HTTP between the client and web listener behind the firewall |
| **Reverse Proxy** | HTTP/1.0 proprietary reverse proxy (developed by the customer) for client authentication using a session cookie. (All HTTP requests need to include the cookie to pass through the reverse proxy.) |
| **Client** | Netscape/Internet Explorer using JInitiator |



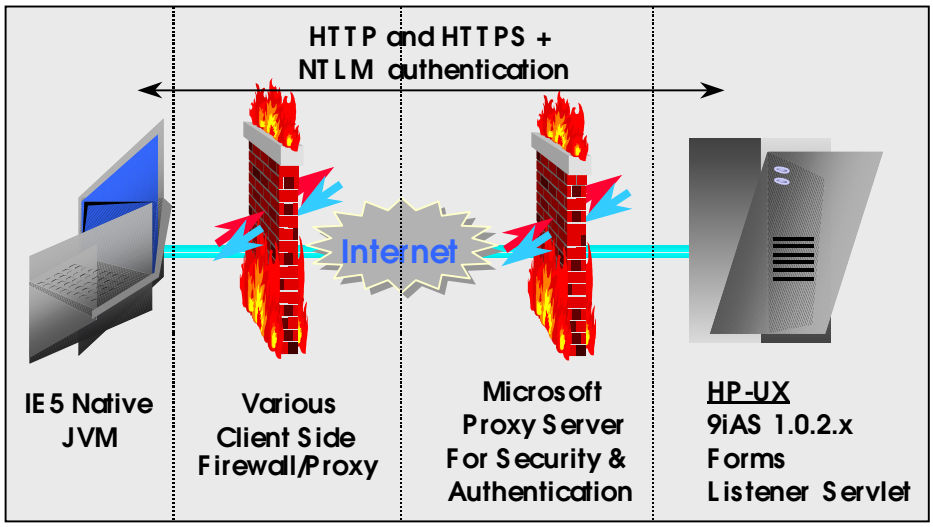
## Listener Servlet Using a Netscape Proxy Server

| | |
|---|---|
| **Server Platform** | Sun Solaris 2.7 |
| **Protocol** | HTTP and HTTPS |
| **Proxy Server** | Netscape proxy server with authentication enabled |
| **Client** | Netscape or Internet Explorer using JInitiator and Internet Explorer 5 with native JVM (without JInitiator) |



### Listener Servlet Using a Microsoft Proxy Server

| | |
|---|---|
| **Server Platform** | HP-UX |
| **Protocol** | HTTP and HTTPS |
| **Proxy Server** | Microsoft proxy server with authentication enabled (NTLM) |
| **Client** | Internet Explorer 5 with native JVM (without JInitiator) NTLM authentication is Microsoft proprietary and works only in Internet Explorer. No JInitiator support. |




### HIDE USER/PASSWORD

Prior to Forms6*i* Patch 7, if the user id is specified in the configuration file (formsweb.cfg), a Forms session using the Forms Servlet (or CGI) generates a page where the entire userid value, including the password, is visible in the HTML source. This visibility created an obvious security issue. The only alternative was to not specify a userid value or password, in which case Forms would prompt the user for the userid or password.

With Patch 7, the userid parameter value is not included in the HTML generated by the Forms Servlet, provided the Forms Servlet is used in conjunction with the Listener Servlet. It will not work when using the Forms CGI or static HTML pages or when using the Forms Listener.

You must do the following for this enhancement to work:

Specify the user/password@database using a parameter called "userid" (not case-sensitive). This is already done if you are using the default baseHTML files, which are provided when Forms is installed. They contain syntax like `userid=%userid%`.

Use the FormsServlet and ListenerServlet.

Specify the serverURL value using a parameter called "serverURL" (not case-sensitive). Again, this is already done if you are using the configuration files provided with Forms.

## ENHANCED SINGLE SIGN-ON (SSO) SUPPORT

Forms Services applications running in a single sign-on environment (using Oracle Login Server) require a special logon format that consists of name-value pairs separated by ampersands (&). Prior to patch 7, users had to adapt to this special logon format by modifying the base html files, which were read by the Forms Servlet.

With patch 7, the default Forms userid format can be changed by only modifying the formsweb.cfg file.

### To use enhanced single sign-on support:

1. Change the logon mask to a format like `username=value&password=value&database=value`, by specifying the following entry in the formsweb.cfg file: `userid=%user%/%password%@%database%.` This can be done in a separate configuration section. For example:

   ```
   [myapp_sso]
   userid=%user%/%password%@%database%
   ```

   **Note:** The names between % (user, password, and database) can be changed as long as they match what is later used in the URL.

2. Go to **Logon Server Adminstration** page, which is accessible from Oracle Portal. Configure your Forms application as a new *external application*. Make sure you specify:

   The URL used to invoke the application (for example, http://myserver/servlet/f60servlet).

   The field names of the logon parameters used to give the username and password.

   Extra logon parameter (displayed to the user or with a suitable default value), such as the configuration parameter and the database connect information. The name of the database connect parameter is whatever you specify in the formsweb.cfg file.

For example, to define a name “Source” for the database connect parameter, specify the database connect parameter in the formsweb.cfg file to read:
`userid=%name%/%password%@%Source%`

The "Type of Authentication Used" as POST. This means logon parameters will be passed to the Forms Servlet as URL parameters using the POST method, so they will not be visible to users in the browser's location.

## TIME ZONE SUPPORT

An important, but often overlooked, consideration when deploying applications is the simple concept of time. When a user records a time value in an application, what exactly do they mean? The time where they are? The time where Forms Services is running? Or the time on the Database Server?

Imagine the scenario where a call center employee based in England is on the phone to a customer reporting the time of a credit card theft whilst on holiday in Hawaii, with the data stored in a database in New York. Then if that needs to be matched up with a fraudulent purchase made over the Internet to a Web site in Singapore, you can see how a universal concept of time is important.

Oracle6*i* Forms supports the ability to define the time zones at all levels of your application so that the correct time is stored in the database.

### About DATETIME Items

Values for Forms items with data type DATETIME are automatically converted between the server time zone region for the value stored in the Forms record manager, and the local time zone region for the value in the client's widget.

This conversion does not apply to items of data type DATE.

Time zone conversions take into account historical data when transitions between daylight and standard time have occurred (or are anticipated to occur) in the relevant time zone regions. This historical data is extracted from the time zone file specified by the FORMS60_TZFILE environment variable.

An automatic conversion of an ambiguous DATETIME item value (because it falls into the 2-hour overlap period that occurs once a year when switching from daylight back to standard time) assumes that the ambiguous value represents standard time.

#### About Automatic Conversion of DATETIME Items

There are special considerations regarding the automatic conversion of DATETIME item values in a block that's in Enter-Query mode. Item values entered by the end user in enter-query mode will be time-zone adjusted (to produce the value used in the query's WHERE clause) when the value is:

A full DATETIME value.

A date-only portion of a DATETIME value. When no time-zone adjustment is in effect, such a value will generate a BETWEEN clause specifying the entire day (from 00:00:00 through 23:59:59). When time-zone adjustment is in effect, the BETWEEN clause still specifies a 24-hour range, but it's adjusted from the local time to server time (and thus might, e.g., specify a range of 03:00:00 through 02:59:59 on the following day in the server's time zone).

A DATETIME value preceded by a relational operator.

Item values entered by the end user in Enter-Query mode are not time-zone adjusted (to produce the value used in the query's WHERE clause) when the value is:

A DATETIME value containing wildcard character(s) ('%' or '_').

A value beginning with '#'. Such a value, in enter-query mode, is interpreted as a fragment of a WHERE clause, and the characters that were typed following the '#' are inserted into the WHERE clause with no change.

A value beginning with ':' or '&'. These values are used in conjunction with the deprecated query-where dialog, and are treated somewhat analogously to '#' values.

Item values set programmatically by the COPY built-in are time-zone adjusted (to produce the value used in the query's WHERE clause) only when the value is:

A DATETIME value preceded by a relational operator. Note that the date portion of such a value must be in the item's input date format, not in built-in date format (as COPY values normally are).

A full DATETIME value or a date-only portion of a DATETIME value that's not valid for the built-in date format but is valid for the item's input date format. (COPY will first attempt to convert such a value using the built-in date format - with no time-zone adjustment, and if that fails, will then attempt to convert it using the item's input date format - with time-zone adjustment).

Full DATETIME item values that are set programmatically by a PL/SQL assignment are not time-zone adjusted.

**Note:** :SYSTEM.LAST_QUERY always shows DATETIME values in the server's time zone.

### ADJUST_TZ Built-in

This built-in adjusts an Oracle `DATE` value from one time zone region to another.

**Syntax:**

PROCEDURE ADJUST_TZ
(date_var IN OUT DATE,
from_tz VARCHAR2,
to_tz VARCHAR2);

PROCEDURE ADJUST_TZ
(date_var IN OUT DATE,
from_tz VARCHAR2,
to_tz VARCHAR2,
is_daylight IN OUT BOOLEAN,
is_in_overlap OUT BOOLEAN,
timezone_label OUT VARCHAR2);

**Parameters:**

`date_var` is adjusted from the `from_tz` time zone region to the `to_tz` time zone region. `date_var` can be a Forms `DATE` or `DATETIME` item, or a Forms `DATE` parameter, or a local PL/SQL `DATE` variable.

Three additional parameters provide extra feedback as to where the date falls in the `to_tz` time zone region.

> `is_daylight` indicates whether the date is in daylight saving time. The `is_daylight` parameter is also used as an input parameter, to disambiguate dates that fall within the `from_tz` time zone region's overlap period. It's ignored if `date_var` falls outside the `from_tz` time zone region's overlap period. If omitted, `is_daylight` defaults to `FALSE`.
>
> `is_in_overlap` indicates whether the date falls within the "overlap period" (the ambiguous 2-hour period that occurs once a year when switching from daylight saving time back to standard time).
>
> `timezone_label` is a label for daylight versus standard that's appropriate for the `to_tz` time zone region. For example, when the to_tz time zone region is US/Pacific, timezone_label will be set to PST when is_daylight is set to FALSE, and to PDT when is_daylight is set to TRUE.

### FORMS60_TZFILE

**Note:** *Beginning with patchset 10 (6.0.8.19), environment variable FORMS60_TZFILE controls certain aspects of time zone support. Oracle Applications customers must not set this environment variable; the value for this will*

*be automatically set based on how time zones are set up within Oracle Applications 11i.*

The FORMS60_TZFILE environment variable specifies the name of the time zone file. The time zone file specifies historical data and rules for daylight saving time, for a set of time zone regions. While you can use either the `timezone.dat` or `timezlrg.dat,` it is recommended you use timezlrg.dat, since it is useful for global companies. Whichever file you choose, it should be used in the database as well as in Forms.

If the value of the environment variable does not specify a full path (for example, "timezone.dat" or "timezlrg.dat"), then Forms looks for the specified file in the current working directory and, if not found, then in #ORACLE_HOME/forms60/zoneinfo (on Unix) or %ORACLE_HOME%\forms60\zoneinfo (on Windows).

If the FORMS60_TZFILE environment variable is not set, the only valid time zone region is `GMT`. Thus, if FORMS60_DATETIME_SERVER_TZ and FORMS60_DATETIME_LOCAL_TZ are also not set (or are set to '`GMT`'), then time zone conversions are turned off.

If any of these environment variables is set to an invalid value (non-existent time zone file or invalid time zone region), then immediately after startup, an error message is sent to the client:

```
FRM-91126 Invalid value %s for environment variable
%s
```

and the runform process terminates.

#### FORMS60_DATETIME_LOCAL_TZ

This environment variable specifies the initial value of the local time zone region for DATETIME items.

#### FORMS60_DATETIME_SERVER_TZ

This environment variable pecifies the permanent value of the server time zone region for DATETIME items.

### Datetime Local TZ Property

This property specifies the local time zone region for DATETIME items.

**Set At Runtime:** Yes

**Default:** Value of the FORMS60_DATETIME_LOCAL_TZ environment variable (if set); otherwise the time zone region extracted from Java client (if FORMS60_TZFILE is set); otherwise GMT.

**Valid Values:** Any string listed as a valid time zone region in the time zone file.

**Usage Note:** Specifying an invalid timezone_region_string in SET_APPLICATION_PROPERTY (DATETIME_LOCAL_TZ, timezone_region_string); results in the generic error:

```
FRM-41340 Invalid property or property value for
Set_Application_Property.
```

### Datetime Server TZ Property

This property specifies the server time zone region for DATETIME items.

**Set At Runtime**: No

**Default:** Value of the FORMS60_DATETIME_SERVER_TZ environment variable (if set); otherwise GMT.

**Valid Values:** Any string listed as a valid time zone region in the time zone file.

**Usage Note:** Issuing SET_APPLICATION_PROPERTY (DATETIME_SERVER_TZ, timezone_region_string); will result in the generic error:

```
FRM-41340 Invalid property or property value for
Set_Application_Property
```

whether or not timezone_region_string is valid.

## CANCELLING A QUERY IN A LONG LIST LOV

This feature allows end users to see how many rows are being fetched as the query is being run, and can cancel it anytime during the run. To use this feature, the Form should be in Non-Blocking mode (i.e. Interaction Mode should be set to Non-Blocking) and Filter Before Display property should be set to Yes.

In previous versions of Forms, if a query in a Long List LOV returned a large number of rows (such as 100,000 or more) end users had to wait until Forms fetched all the records before anything could be done with the LOV. If, for example, the user had made a mistake when specifying a condition which caused a large number of records to be fetched, they had no way of knowing that the underlying query was going to fetch a large number of records. Also, even if they did know that a large number of records was being returned, they still had to wait until all the rows were fetched, before they could try again.

With this feature turned on the users can see how many records are fetched (at any point in time) and can cancel any time and would be able to work with the fetched set (as if the query had run to completion).

**Oracle9iAS Forms Services, Forms6i Patch 10: Forms Listener Servlet for Deployment of Forms on the Internet**
**April 2002**
**Authors: Chris Barrow, Dean Ho, Regis Louis, Frank Nimphius, Kannan Parthasarathy, Jonas Rheborg**

**Oracle Corporation**
**World Headquarters**
**500 Oracle Parkway**
**Redwood Shores, CA 94065**
**U.S.A.**

**Worldwide Inquiries:**
**Phone: +1.650.506.7000**
**Fax: +1.650.506.7200**
**www.oracle.com**

**Oracle Corporation provides the software**
**that powers the internet.**