

Forms 6i Patch 4: Forms Listener Servlet for Deployment of Forms on the Internet

An Oracle White Paper
February 2001

Forms 6i Patch 4: Forms Listener Servlet for Deployment of Forms on the Internet

OVERVIEW.....	4
Current Architecture.....	4
Socket, HTTP, and HTTPS Connection Modes.....	5
Issues with the Current Architecture for Internet Deployment of Forms	6
INTRODUCING THE FORMS 6I LISTENER SERVLET	7
What is the Forms Listener Servlet?.....	7
Why Should I Use the Forms Listener Servlet?.....	8
INSTALLING THE FORMS LISTENER SERVLET	9
BASIC CONFIGURATION	9
Getting Started.....	9
Step 1: Setting environment variables in jserv.properties	10
Example JServ.properties file for NT.....	10
Example JServ.properties file for UNIX	10
Step 2: Determine whether to run JServ in Auto-Start mode	11
Step 3: Add the New Applet parameter serverURL to the Formsweb.cfg file.....	11
Configuration using Forms CGI or the Forms Listener Servlet.....	12
Configuration using static HTML pages.....	12
CONFIGURATION NOTES REGARDING PORTS	12
END-USER (WEB BROWSER) REQUIREMENTS.....	13
USING HTTPS WITH THE FORMS LISTENER SERVLET.....	14
Server Requirements	14
Client requirements	14
Using HTTPS with Internet Explorer and native JVM.....	14
Using HTTPS with Oracle JInitiator	15
TROUBLESHOOTING	15
Ensure the Listener Servlet and native methods library is available....	15
Try to run the test form using the servlet.....	15
Debug tracing	16
PERFORMANCE/SCALABILITY TUNING.....	16
Limit the number of HTTPD processes	16
Set the maxClient directive to a High value	16
On UNIX, limit the number of sockets in CLOSE_WAIT or FIN_WAIT state.....	17
Disable JServ logging.....	17

Disable the JServ auto-reload.....	17
Run the HTTP listener and JServ engine(s) on different machines	17
LOAD BALANCING JSERV.....	17
Case 1: Two JServ engines on the same host as Apache web listener	18
Step 1: Configure the JServ engines.....	18
Step 2: Modify the jserv.conf file to distribute the load.....	19
Step 3: Create start and stop scripts.....	20
Case 2: Two JServ engines on a host other than Apache web listener	22
Step 1: Configure the JServ engines on Host 2 (the one running JServ)	23
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	23
Step 3 : On UNIX, load the Apache JServ communication module	23
Step 4 : Start the JServ engines in the JServ hosts.....	24

Forms 6i Patch 4: Forms Listener Servlet for Deployment of Forms on the Internet

OVERVIEW

This document describes the new architecture option available for the Oracle Forms Services component of the Oracle9i 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.

Current 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.

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 6i 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.

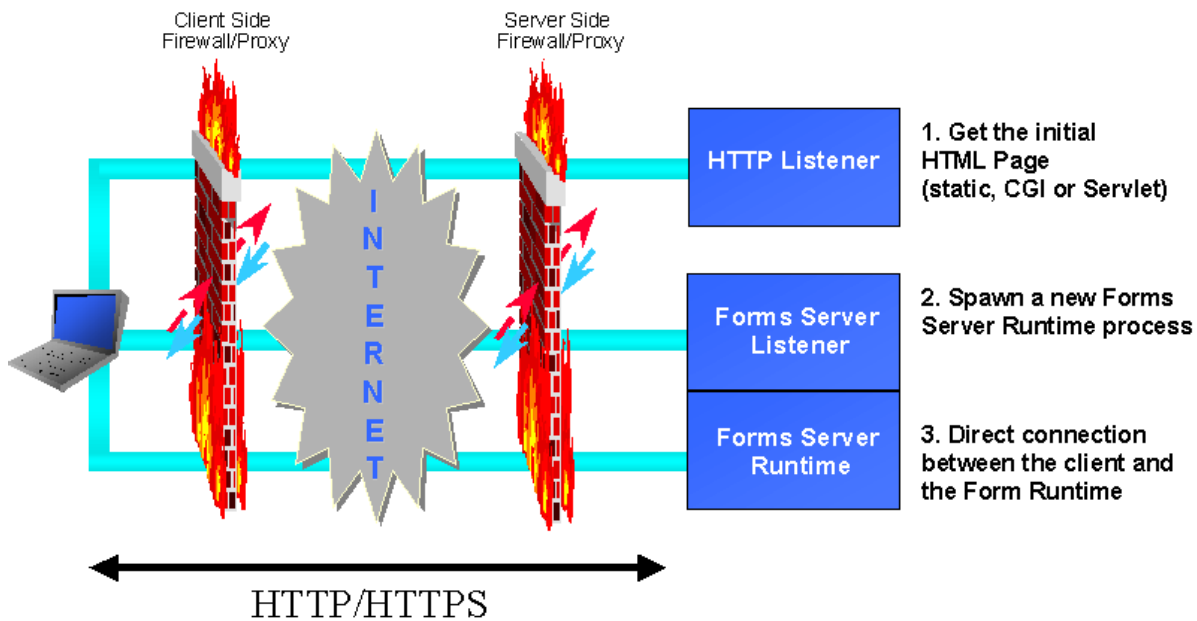


Figure 1. Current Forms Server architecture for Internet deployment

Issues with the Current 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 FORMS 6I LISTENER SERVLET

What is the Forms Listener Servlet?

The Forms Listener Servlet is a Java servlet 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 Forms Listener is still available for direct socket connections, and still supports HTTP and HTTPS connections.*

The Forms Listener Servlet runs on a web server that is equipped with a servlet engine, such as the Oracle9i Application Server. The web server directs HTTP requests for the Forms Listener Servlet directly to the servlet instances.

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.

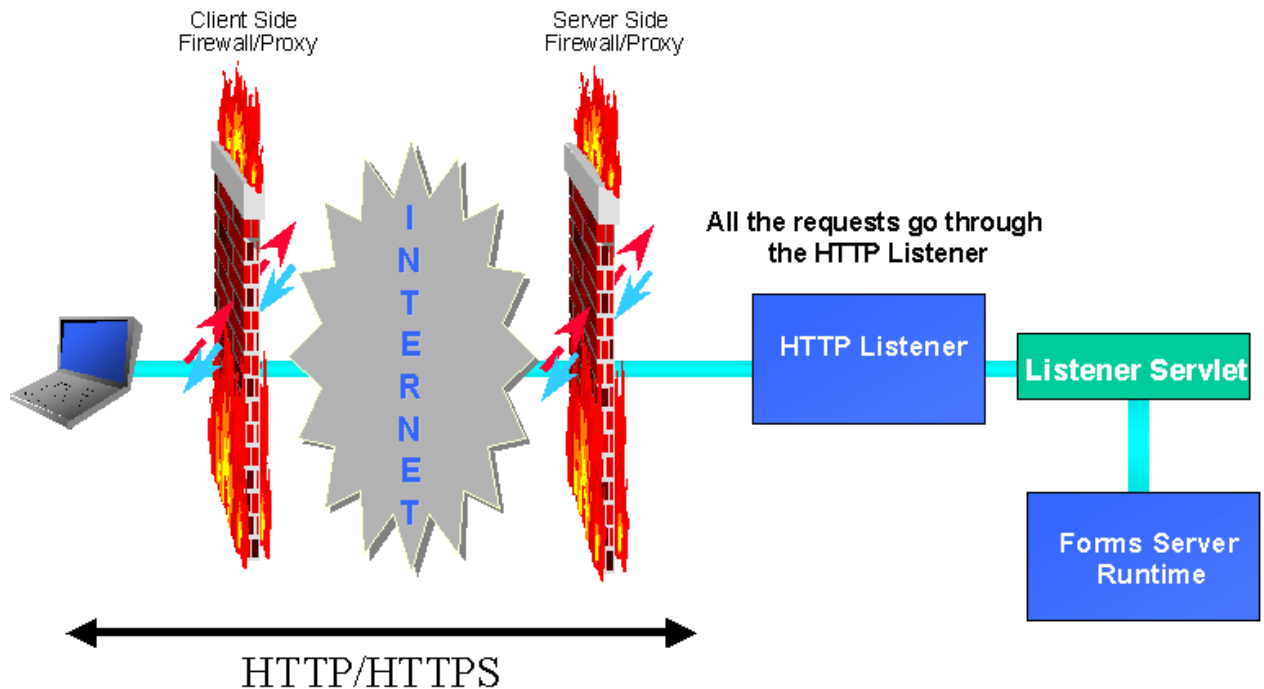


Figure 2. New architecture using the Forms Listener Servlet

Why Should I Use the Forms Listener Servlet?

The Forms 6i 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 Forms 6i Listener Servlet provides the following benefits:

- **Broader range of firewalls and proxies supported**

Because the client browser always communicates with the web server (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 5.0 with native JVM support**

In addition to working with Oracle JInitiator, this architecture supports the use of Internet Explorer 5.0 with native Microsoft JVM for Internet deployment using HTTP and HTTPS connection modes.

The Forms 6i 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.

- **Forms Server log**

In the previous architecture, the Forms Listener logged all connections to a log file on the server when logging was turned on. This log file does not exist for the Forms Listener Servlet.

- **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 standards load balancing methods.

INSTALLING THE FORMS LISTENER SERVLET

The Forms Listener Servlet is installed as part of Forms Patchset 4.

- If you install Forms Patchset 4 on top of an existing version of iAS, you will need to manually configure the Forms Listener Servlet. (Install Forms 6i Patchset 4 in 6iserver or 806 Oracle Home depending on the version of iAS you are using – 1.0.1 or 1.0.2. Install only the Forms Server components using a custom install. Do not install Forms Builder.)
- If you install Forms Patchset 4 as part of iAS 1.0.2.2 and higher, the basic configuration of the Forms Listener Servlet is automatic.

BASIC CONFIGURATION

Location of the configuration files

Jserv.properties	<Oracle_home>/apache/jserv/conf
Jserv.conf	<Oracle_home>/apache/jsserv/conf
httpd.conf	<Oracle_home>/apache/apache/conf
Formswb.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 for Forms Server Runtime processes is inherited from the servlet engine (JServ). Therefore, the environment variables required for a Forms Server Runtime process (for example, PATH, ORACLE_HOME, FORMS60_PATH) are set in the JServ environment.

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

Pre-configuration requirements:

- <Forms oracle_home>/bin must be in the PATH so that the Forms Server Runtime executable can be located.
- Shared libraries must be locatable, notably the Forms Listener Servlet Java Native Methods (or JNI) library.
 - On NT, PATH must include the <Forms oracle_home>/bin directory.
 - On UNIX, LD_LIBRARY_PATH must point to <Forms oracle_home>/lib.
- Forms Listener Servlet classes (<Forms Oracle_home/forms60/java/f60srv.jar) must be available in the Java classpath of the servlet engine.

The following configuration steps 1 through 3 are performed automatically if you install Forms Patchset 4 as part of iAS 1.0.2.2 and higher.

Step 1: Setting environment variables in jserv.properties

JServ environment settings are configured in the **Jserv.properties** file using wrapper.env directives (or wrapper.path to set the PATH).

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 Oracle9iAS Oracle Home.

Example JServ.properties file for NT

```
wrapper.classpath=d:\oracle\806\forms60\java\f60srv.jar
wrapper.path=d:\oracle\806\bin;d:\oracle\isuites\bin
wrapper.env=ORACLE_HOME=d:\oracle\806
wrapper.env=FORMS60_PATH=d:\oracle\806\forms60;c:\myapp
```

Example JServ.properties file for UNIX

```
wrapper.path=/private2/oracle/806/bin:/private2/oracle/isuites/bin
wrapper.env=ORACLE_HOME=/private2/oracle/806
wrapper.env=FORMS60_PATH=/private2/oracle/806/forms60:/home/myapp
wrapper.env=LD_LIBRARY_PATH=/private2/oracle/806/lib
```

Step 2: 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 `https.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 3: Add the New Applet parameter serverURL to the Formsweb.cfg file

When using the Forms Listener in pre-patchset 4 releases, 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 new 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 Oracle9iAS 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 `serverHost`, `serverPort` and `serverURL` are specified (and if `serverURL` is *not* an empty string), then `serverURL` takes precedence, meaning that the Forms Listener Servlet is used.

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://).

Configuration using Forms CGI or the Forms Listener Servlet

The base HTML files installed by the Forms 6i patch 4 release (base.htm, basejini.htm, and baseic.htm) contain the new applet parameter serverURL.

The formsweb.cfg file sets serverURL to an empty string by default (so that, by default, the Forms Listener is used). In order to use the Forms Listener Servlet, you must set serverURL to an appropriate value (as described in the previous section). This can be done by editing the formsweb.cfg file and changing the default serverURL setting, or by adding a serverURL parameter setting in a specific section as follows:

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

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 new:

```
<APPLET CODEBASE="/if994/java/"  
  
        CODE="oracle.forms.engine.Main"  
  
        ARCHIVE="f60all.jar"  
  
        WIDTH="800"  
  
        HEIGHT="600">  
  
<PARAM NAME="serverURL"  
VALUE="/servlet/oracle.forms.servlet.ListenerServlet">  
  
<PARAM NAME="serverArgs" VALUE="myform myuser/mypass">  
  
<PARAM NAME="splashScreen" VALUE="No">  
  
<PARAM NAME="lookAndFeel" VALUE="Generic">  
  
</APPLET>
```

CONFIGURATION NOTES REGARDING PORTS

In future patchset releases, the handling of ports may change. Currently, the Forms Listener Servlet communicates with the Forms Server Runtime processes using hard-coded port numbers.

By default, the ports used start with 9031. The port numbers used are 9031, 9032, up to 9031+100-1 = 9130, and then start again at 9031. If there is a second JServlet engine, it will use 9131, 9132, up to 9230, and then start again at 9131.

If other software is listening on one of the ports that we use, a conflict might result, preventing some Forms sessions from starting successfully.

If you anticipate a conflict, you can change the port numbers that the Forms Listener Servlet uses by setting the following servlet initialization parameters in the **zone.properties** file:

- startPort (default 9031)
- maxPorts (default 100)

By setting these values, port numbers will start at startPort, and will recycle after maxPort times.

For example, if startPort=9001, and maxPorts=10, the Forms Listener Servlet will use ports 9001, 9002, and so on through 9010, and then go back and reuse 9001. If there are two JServ engines, the second engine will use 9011, 9012, and so on through 9020, and then reuse 9011.

Set maxPorts to the maximum number of simultaneous requests that you anticipate the web server will receive to start a Forms session. The default value of 100 is fairly high.

To set these parameters, append the following line in your zone.properties file:

```
servlet.oracle.forms.servlet.ListenerServlet.initArgs=startPort=8500,maxPorts=50
```

In the above example, 8500 is the port that the Forms Listener Servlet will start with, and 50 is the maximum number of ports opened before cycling back to the first port.

END-USER (WEB BROWSER) REQUIREMENTS

Supported web browsers and configurations are similar to Oracle Forms 6i, except for the following additional requirements:

- **JInitiator version 1.1.8.6 or above must be used.** If Netscape Navigator or Internet Explorer is being used with Oracle JInitiator as the JVM, users must upgrade to JInitiator 1.1.8.6 or higher. (JInitiator 1.1.8.6 is supplied with Forms 6i patch set 4). If you attempt to execute a Forms application with the Forms Listener Servlet using a previous version of JInitiator, an error message explaining the problem is displayed.
- **When using Internet Explorer with the native JVM, session cookies must be enabled.** All end users running Forms applications in Internet Explorer 5.0 or higher with Microsoft's native JVM (rather than Oracle JInitiator), MUST configure Internet Explorer to accept session cookies. Session cookies are not stored on the user's disk and only last for the duration of the browser session.

Check this setting as follows:

1. Select the option **Internet Settings** from the **Tools** menu.
2. Click on the **Security** tab. If the security level is set to **Medium**, then session cookies are already enabled.
3. To double-check the setting, or if the security level is marked as **High** or **Custom**, click on the **Custom Level** button.
4. Scroll down to and double-click the **Cookies** node to expand it.
5. Look under the node marked **Allow per-session Cookies (not stored)**. If it is marked **Disabled**, change it to **Enable** or **Prompt**.

USING HTTPS WITH THE FORMS LISTENER SERVLET

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.

Note: 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/Oracle/Oracle/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 UNIX) are correct.

1. Point your web browser to a URL like:

```
http://your_server/servlet/oracle.forms.servlet.ListenerServlet
```
2. You should get page titled "Forms 6i 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 ifjssl60.dll (libifjssl60.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.

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

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

Debug tracing

Trace messages are written to the Forms Listener Servlet engine's log file (usually jserv.log) if "/debug" 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.

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

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

On UNIX, limit the number of sockets in CLOSE_WAIT or FIN_WAIT state

On Solaris (and possibly some other UNIX platforms), set the following kernel TCP parameters to low values (like 1000):

```
tcp_close_wait_interval  
tcp_fin_wait_2_flush_interval
```

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.

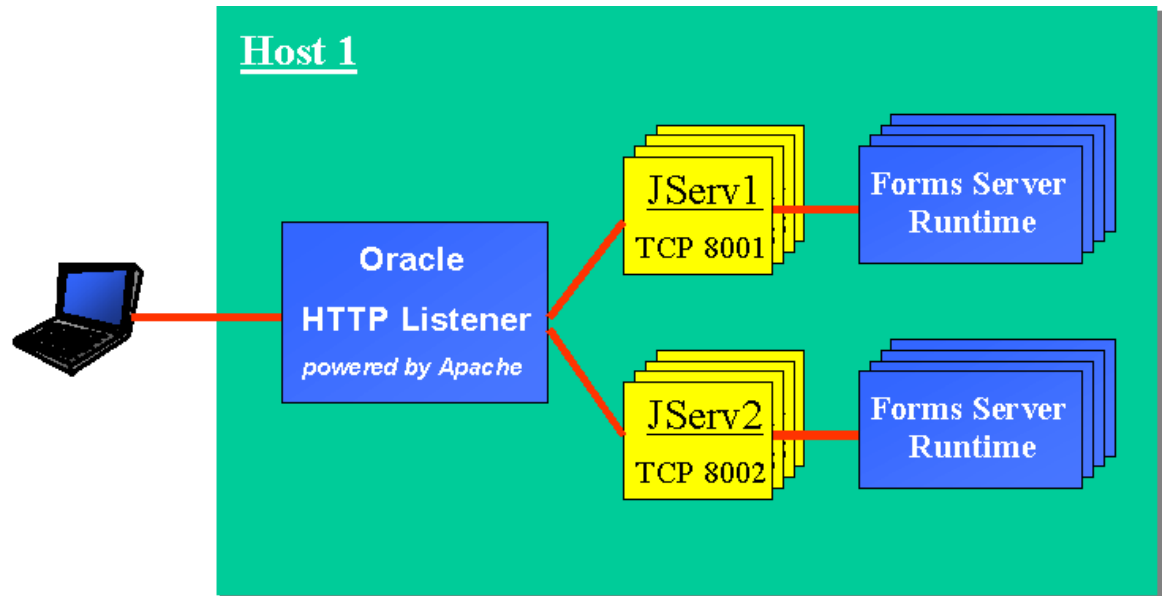


Figure 3. Multiple JServ engines on one host

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 above 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. The Forms Listener Servlet assumes 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).

Note: JS1, JS2 and so on are the *required* values for the ApJServRoute directive. The Forms Listener Servlet needs this information to be able to route each request to the appropriate JServ session.

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 UNIX environment:

Start.sh

```
#!/bin/sh -x

# Setting environment variables
FORMS60_PATH=/privatell/myapp
Export FORMS60_PATH

# Setting the properties files
properties1=/privatel/oracle/iAS102/Apache/Jserv/etc/jserv1.properties
properties2=/privatel/oracle/iAS102/Apache/Jserv/etc/jserv2.properties

# setting the log files
log1=/privatel/oracle/iAS102/Apache/Apache/logs/jserv_manual1.log
log2=/privatel/oracle/iAS102/Apache/Apache/logs/jserv_manual2.log

# setting the classpaths
CLASSPATH=$CLASSPATH:/privatel/oracle/iAS102/Apache/Jserv/libexec/Apach
eJServ.jar:/privatel/oracle/iAS102/Apache/Jsdk/lib/jsdk.jar:/privatel/o
racle/iAS102/jdbc/lib/classes111.zip:/privatel/oracle/iAS102/Apache/Apa
che/htdocs/_pages:/privatel/oracle/iAS102/Apache/Apache/htdocs/OnlineOr
ders_html:/privatel/oracle/iAS102/Apache/Apache/htdocs/OnlineOrders_htm
```

```
l/OnlineOrders.jar:/private1/oracle/ias102/Apache/BC4J/lib/connectionmanager.zip:/private1/oracle/ias102/jdbc/lib/classes111.zip:/private1/oracle/iASHome1/806/forms60/java/f60srv.jar:/private1/oracle/iASHome1/806/forms60/java
```

```
#Setting the java Compiler
```

```
JAVA=/private1/oracle/ias102/Apache/jdk/bin/java
```

```
nohup $JAVA -classpath $CLASSPATH org.apache.jserv.JServ $properties1 >> $log1 2>&1 &
```

```
nohup $JAVA -classpath $CLASSPATH org.apache.jserv.JServ $properties2 >> $log2 2>&1 &
```

Stop.sh

```
#!/bin/sh -x
```

```
properties1=/private1/oracle/ias102/Apache/Jserv/etc/jserv1.properties
```

```
properties2=/private1/oracle/ias102/Apache/Jserv/etc/jserv2.properties
```

```
log=/private1/oracle/ias102/Apache/Apache/logs/jserv_stop.log
```

```
CLASSPATH=$CLASSPATH:/private1/oracle/ias102/Apache/Jsdk/lib/jsdk.jar
```

```
CLASSPATH=$CLASSPATH:/private1/oracle/ias102/Apache/Jserv/libexec/ApacheJServ.jar:/private1/oracle/iASHome1/Apache/jdk/lib/classes.zip
```

```
JAVA=/private1/oracle/ias102/Apache/jdk/bin/java
```

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

```
nohup $JAVA -classpath $CLASSPATH org.apache.jserv.JServ $properties2 -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\classes111.zip;D:\Oracle\iSuites\Apache\Apache\htdocs\_pages;D:\Oracle\iSuites\Apache\Apache\htdocs\OnlineOrders_html;D:\Oracle\iSuites\Apache\Apache\htdocs\OnlineOrders_html\OnlineOrders.jar;D:\Oracle\iSuites\Apache\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\classes111.zip;D:\Oracle\iSuites\Apache\Apache\htdocs\_pages;D:\Oracle\iSuites\Apache\Apache\htdocs\OnlineOrders_html;D:\Oracle\iSuites\Apache\Apache\htdocs\OnlineOrders_html\OnlineOrders.jar;D:\Oracle\iSuites\Apache\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.

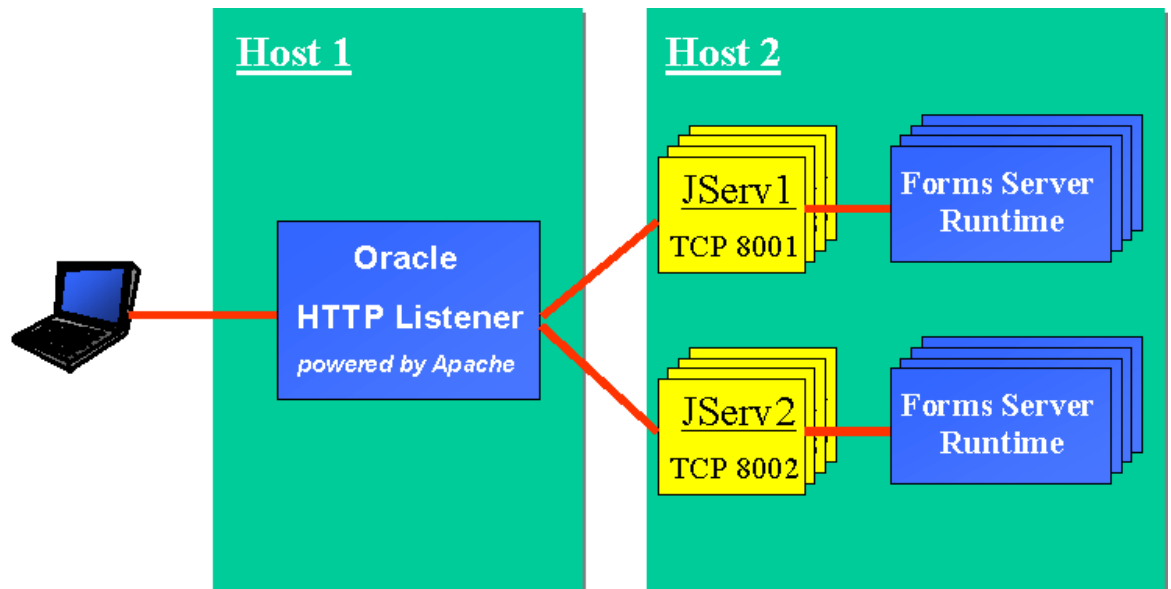


Figure 4. Multiple JServ engines on multiple hosts

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 UNIX, load the Apache JServ communication module

On UNIX, 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.



Forms 6i Patch 4: Forms Listener Servlet for Deployment of Forms on the Internet
February 2001
Author: Regis Louis

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.

Oracle is a registered trademark of Oracle Corporation. Various
product and service names referenced herein may be trademarks
of Oracle Corporation. All other product and service names
mentioned may be trademarks of their respective owners.

Copyright © 2001 Oracle Corporation
All rights reserved.