Oracle9*i* Application Server Forms Services

Forms6*i* Patch 6: Oracle Forms Listener Servlet for Deployment of Forms on the Internet

An Oracle White Paper
May 2001

Oracle9iAS Forms Services

Forms6*i* Patch 6: Forms Listener Servlet for Deployment of Forms on the Internet

Oracle9iAS Forms Services

Forms6*i* Patch 6: 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.

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.

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.

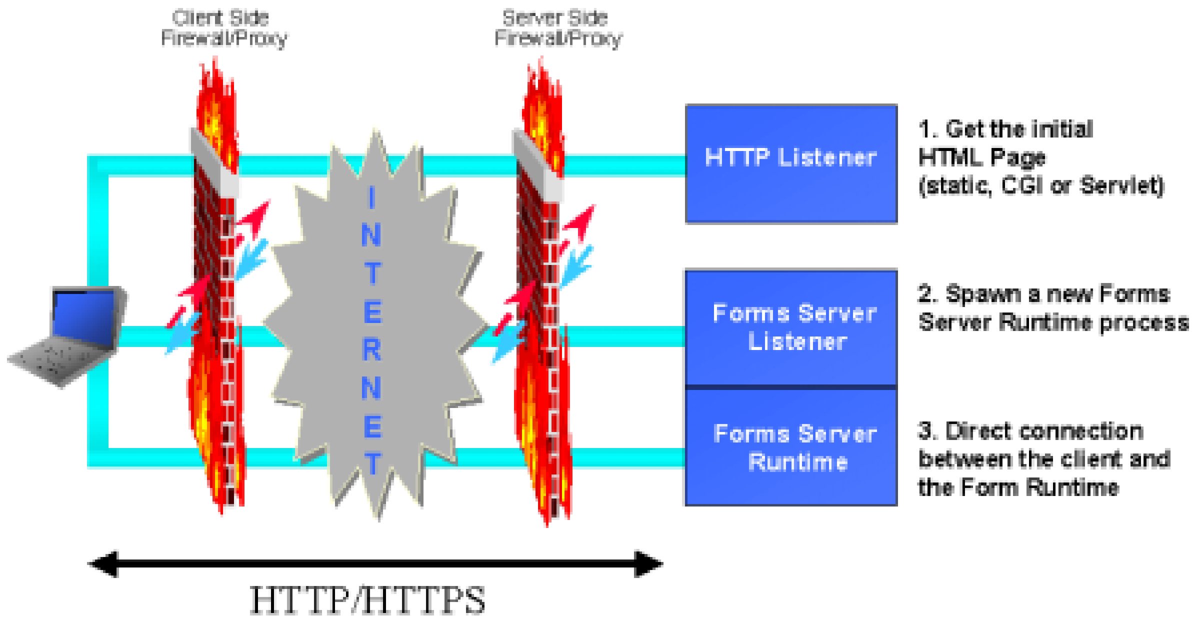


Figure 1. Pre-Patch 4 Forms Server architecture for Internet deployment

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 FORMS 6I 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 Oracle9i 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.

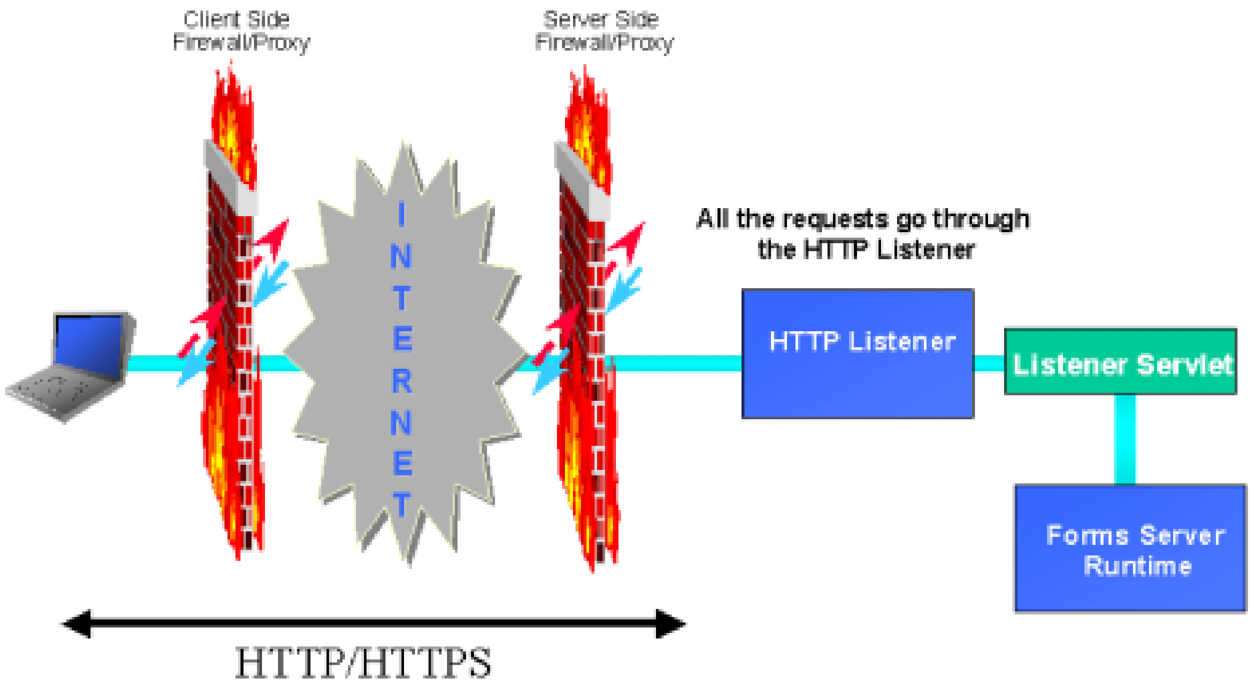

Client Side Firewall/Proxy
Server Side Firewall/Proxy
INTERNET
All the requests go through the HTTP Listener
HTTP Listener
Listener Servlet
Forms Server Runtime
HTTP/HTTPS

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 Forms6i 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

Note: If you are using Patch 5 or higher, you have added configuration flexibility. See **Configuration Options for Patch 5 and Higher** later in this white paper for more information**.**

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. As of Patch 5, it is also possible to set them in a special environment file, as described in detail later in this white paper.

Note: On NT, Forms reads Oracle environment settings from the registry unless they are set 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 Solaris, LD_LIBRARY_PATH must include <Forms oracle_home>/lib.
  - On HP, SHLIB_PATH must include <Forms oracle_home>/lib.

  - On AIX, LIBPATH must include <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.

Step 1: Configure environment settings for the Forms Listener Servlet

PATH, CLASSPATH, and LD_LIBRARY_PATH 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 on NT. On Solaris, /private2/oracle/806 is the Forms Oracle Home, and /private2/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\isuites\bin;d:\oracle\806\bin
```

Example JServ.properties file for Solaris

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

wrapper.path=/private2/oracle/isuites/bin: /private2/oracle/806/bin:

wrapper.env=LD_LIBRARY_PATH=/private2/ias/lib:/private2/oracle/806/lib
```

Step 2: Edit the default.env file

This file contains environment settings for Forms runtime. 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.

> **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 using the new EnvFile parameter described in "Setting Environment Variables for Specific Runtime Processes."

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/j
re11/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/web60
/temp

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

Step 3: Edit the zone.properties file to specify the location of the environment file

In the zone.properties file, set the **EnvFile** parameter in the servlet initialization parameter. The EnvFile parameter specifies the physical path to the file that contains environment variable settings.

Example zone.properties file for NT

```
servlet.oracle.forms.servlet.ListenerServlet.initArgs=EnvFile=d:\oracle
\806\forms60\server\default.env
```

Step 4: 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 5: Add the Applet parameter serverURL to the Formsweb.cfg file

When using the Forms Listener in pre-Patch 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 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.

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 this patch (base.htm, basejini.htm, and baseie.htm) contain the 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 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>
```

CONFIGURATION OPTIONS FOR PATCH 5 AND HIGHER

If you are using Forms6*i* Patch 5 or higher, you have additional configuration flexibility, as described in the sections that follow.

Setting Environment Variables for Specific Runtime Processes

For pre-Patch 5 releases, the only way to set environment variables (such as ORACLE_HOME and NLS_LANG) for runtime processes is in the jserv.properties file (or in the shell script used to start up JServ when it is run manually). This type of setup forces all users to have the same set of environment variables.

If you want to allow users to have different environment variable settings, or if your default environment variable settings clash with the needs of other products, you can create environment configuration files that contain alternate environment variable settings. Then specify the alternate file using the **EnvFile** parameter.

Do the following:

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
```

2. In the zone.properties file, create a ListenerServlet alias that will use the file with the alternate environment variable settings:

```
# ListenerServlet alias for a different set of env vars:
servlet.lservletHR.code=oracle.forms.servlet.ListenerServlet
```

3. In the zone.properties file, set the **EnvFile** parameter for the ListenerServlet alias. The EnvFile parameter specifies the physical path to the file that contains the alternate environment variables.

```
servlet.lservletHR.initArgs=envfile=d:\orant\forms60\server\HumanRe
    s.env
```

A sample zone.properties file for NT might contain the following:

zone.properties file

```
servlet.oracle.forms.servlet.ListenerServlet.initArgs=envfile=d:\orant\forms60\server\default.env

# ListenerServlet alias for a different set of env vars:

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

servlet.lservletHR.initArgs=envfile=d:\orant\forms60\server\HumanRes.env
```

4. 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:

formsweb.cfg file

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

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

; Config section causing env. vars. in HumanRes.env to be used:

[HR]

serverURL=/servlet/lservletHR
```

Setting the Current Working Directory for Specific Runtime Processes

For pre-Patch 5 releases, you must use the default working directory for any runtime instance. This patch allows you to set the current working directory for a specific runtime process.

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

You can specify the current working directory for a specific runtime instance using the listener servlet initialization argument **WorkingDirectory** as shown:

1. In the zone.properties file, create a ListenerServlet alias that will use the alternate current working directory:

```
    # ListenerServlet alias for a different current working directory:

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

2. In the zone.properties file, set the **WorkingDirectory** parameter for the ListenerServlet alias. The WorkingDirectory parameter specifies the physical path to the current working directory:

```
servlet.lservletHR.initArgs=WorkingDirectory=d:\HumanRes\myfiles
```

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 WorkingDirectory parameter to be used

[HR]

serverURL=/servlet/lservletHR
```

Setting the Runtime Executable Name for Specific Runtime Processes

This patch allows you to specify the name of the runtime executable file, which can 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 Oracle iAS), and that the Forms Servlet is running using the servlet alias "f60servlet".

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

   ```
   ApJServMount /forms60java/servlet /root

   ApJServAction .f60 /servlet/f60servlet
   ```
3. 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
   ```
4. Restart Apache/JServ.
5. 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.
6. 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
```

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" of this white paper.
- **Improved performance when running Forms applications under JInitator in HTTPS mode using the Listner 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 tracing

Trace messages are written to the servlet engine's log file (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.

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.

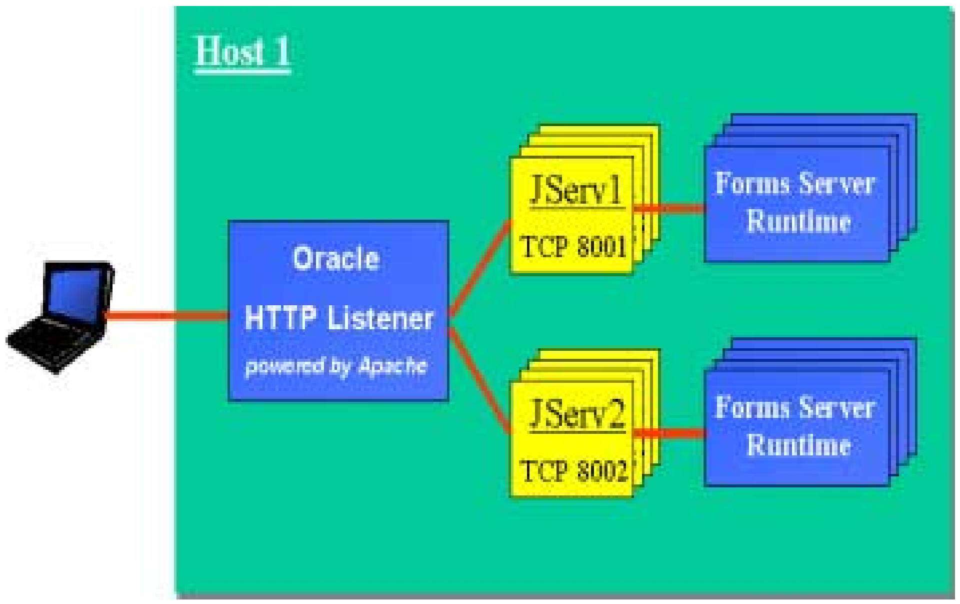


Figure 3. Multiple JServ engines on one host

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:

<u>Start.sh</u>

```
#!/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/f60srv.
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/f60srv.
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\cla
sses111.zip;D:\Oracle\iSuites\Apache\Apache\htdocs\_pages;D:\Oracle\iSu
ites\Apache\Apache\htdocs\OnlineOrders_html;D:\Oracle\iSuites\Apache\Ap
ache\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\cla
sses111.zip;D:\Oracle\iSuites\Apache\Apache\htdocs\_pages;D:\Oracle\iSu
ites\Apache\Apache\htdocs\OnlineOrders_html;D:\Oracle\iSuites\Apache\Ap
ache\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.

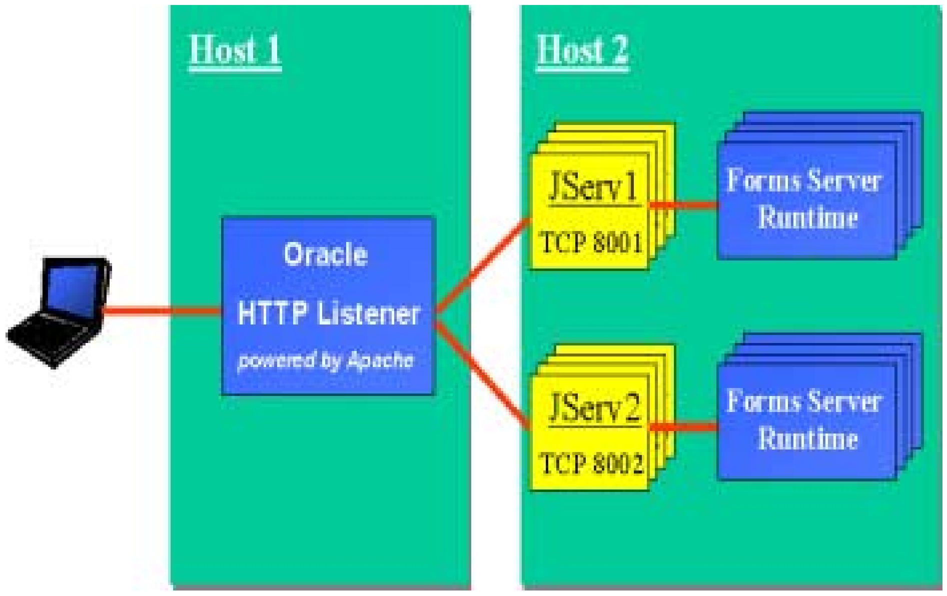


Figure 4. Multiple JServ engines on multiple hosts

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 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

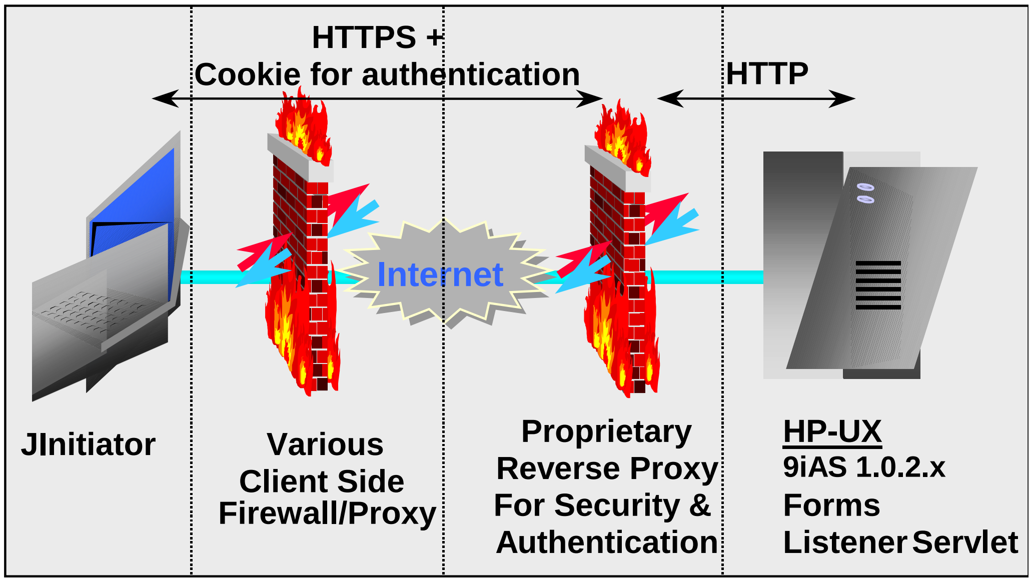
Previously, as a temporary workaround, the Forms Listener Servlet communicated with the Forms Server Runtime processes by using fixed port numbers. However, starting with Forms6*i* patch 6, fixed port numbers are no longer used. Consequently, the maxPorts and startPort servlet initialization parameters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

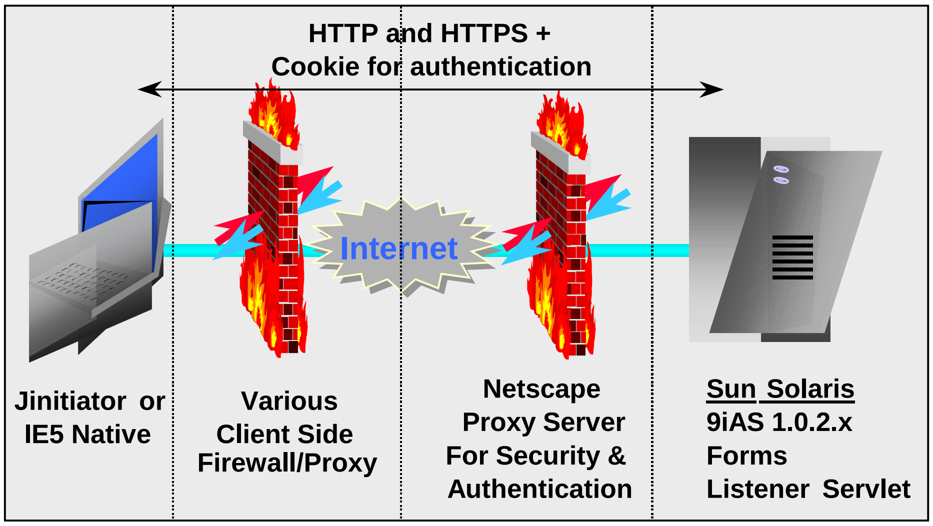
Listener Servlet Using a Reverse Proxy (cookie-based authentication)

Server Platform	HP-UX
Protocol	HTTPS between the client and reverse proxy for internet communication 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


HTTPS +
Cookie for authentication
HTTP
Internet
JInitiator
Various Client Side Firewall/Proxy
Proprietary Reverse Proxy For Security & Authentication
HP-UX
9iAS 1.0.2.x
Forms Listener Servlet

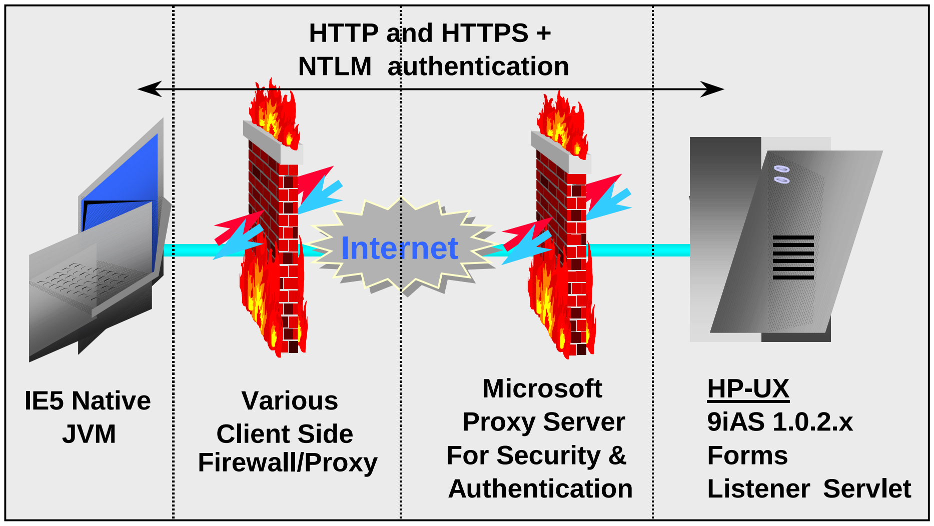
Listener Servlet Using a Netscape Proxy Server

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)


HTTP and HTTPS +
Cookie for authentication
Internet
Jinitiator or IE5 Native
Various Client Side Firewall/Proxy
Netscape Proxy Server For Security & Authentication
Sun Solaris 9iAS 1.0.2.x Forms Listener Servlet

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.


HTTP and HTTPS +
NTLM authentication
Internet
IE5 Native JVM
Various Client Side Firewall/Proxy
Microsoft Proxy Server For Security & Authentication
HP-UX
9iAS 1.0.2.x Forms Listener Servlet

Oracle9iAS Forms Services, Forms6i Patch 6: Forms Listener Servlet for Deployment of Forms on the Internet
May 2001
Author: Regis Louis, Jonas Rheborg, and Dean Ho

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