Oracle9*i* Forms
Migrating Client/Server To the Web

An Oracle White Paper
April 2002

ORACLE

Oracle9*i* Forms
Migrating Client/Server To the Web

Oracle9*i* Forms
Migrating Client/Server To the Web

INTRODUCTION

This whitepaper will help developers migrate their Oracle Forms client/server applications to Forms on the web.

The new release, Oracle9*i* Forms is a web-only release of Forms. This is important to know because Oracle Forms 6i is the last version of Forms to support client/server and character mode. For this reason, Oracle Forms 6i client/server and character mode applications will remain supported until December 31st, 2004, or December 31st, 2007 with Extended Support.

What's In This Guide?

This document concerns itself with migrating from Oracle Forms 6i client/server to Oracle9i Forms on the web, but the concepts apply equally when migrating from any client/server version of Forms to any version of Forms on the web.

Chapter 1 explains the Oracle9*i*AS Forms Services architecture, and how Forms applications run on the web. See how your client/server application can run on the web, often without any modification.

Chapter 2 summarizes the upgrade process: migrating your Forms 6i application to Oracle9*i* Forms.

Chapter 3 details the impacts for Forms developers when building for the web. While the majority of Forms client/server applications can run on the web unchanged, there are some things that developers will have to take into account.

Chapter 4 describes how to configure Oracle9*i* Forms Services and deploy your Forms applications, which is considerably different to deploying client/server Forms.

Chapter 5 lists some web-only features, that weren't available in client/server, which developers can take advantage of when building Forms applications for web.

Appendix A contains a list of references and other sources of information mentioned throughout this paper. Appendix B breaks down what happens during a Forms session. In Appendix C is a sample Windows batch compilation script.

Note: The previous release of Forms was called Oracle Forms 6i, compared to this release, which is called Oracle9*i* Forms. Only migrations from Oracle Forms 6i to Oracle9*i* Forms are supported. If you wish to migrate from an earlier

version of Forms, you need to first migrate to Oracle Forms 6i, and then to Oracle9*i* Forms.

Why Move To the Web?

Deploying your Forms applications to the web has many advantages:

- Ease of maintenance
- Accessibility
- Platform independence for clients
- Cost effectiveness
- Run on the internet
- Single Sign-On
- Access to other technologies
- Reduced network traffic

Client/server applications require installation and maintenance of software on every client machine. The multi-tier architecture of Oracle9*i*AS and Forms Services means that you centrally manage the install and configuration of your application.

When an upgrade or patch to the software is required, simply do so on the central server, and all users will be immediately affected. This negates the need to maintain every single user's machine when an upgrade or patch is available, or a change in configuration is needed.

Any user with a browser, on any platform, can run a Forms application. Compare this with client/server, where each client machine needs to be installed and configured before they can run the application.

This adds up to cost effectiveness as fewer administrators are needed to maintain your application, and yet more users can be reached. A web application has the ability to operate through proxies and firewalls, thus giving an even broader audience access to your application.

Forms applications can take advantage of Single Sign-On, which means your users only need one username and password, and only need logon once. In addition, your Forms application on the web can take advantage of other technologies, such as Java, to integrate with other systems (e.g. Web Services) or enhance the functionality of Forms. These topics are covered in Chapter 4.

One of the biggest advantages of running Forms on the web, though, is the reduced network traffic between the client and the middle tier, compared to the client/server architecture. This is a great benefit, particularly for companies with low bandwidth and/or high-latency networks. See Appendix A for information on downloading a whitepaper comparing client/server and web deployed applications.

1 THE ORACLE9*i*AS FORMS SERVICES ARCHITECTURE

Apart from some well documented caveats, any client/server Forms application can run on the web. And often without any modification!

If you are new to Forms on the web, then this might seem like an amazing statement. This chapter explains how an Oracle Forms client/server application can run on the web.

1.1 Oracle9*i* Forms Components

Oracle9*i* Forms is made up of two parts:

- Oracle9*i* Forms Builder
  This is a component of Oracle9*i* Developer Suite (Oracle9*i*DS), and is used to design and build your Forms applications.
- Oracle9*i* Forms Services
  This is a component of Oracle9*i* Application Server (Oracle9*i*AS), and is used to deploy your Forms applications.

This chapter concerns itself with the latter, Forms Services, and the runtime architecture.

1.2 Multi-Tier Architectures

Oracle Forms on the web is deployed using a three-tier architecture:

- The Client Tier
- The Application Tier or Middle Tier
- The Database Tier

A client/server application, on the other hand, is two-tiered, comprising only the client tier and the database tier.

Before going into more detail on how a three-tier application works, let's look at the two-tier client/server version, and see how it needs to change to run on the web.

1.2.1 TWO-TIER CLIENT/SERVER ARCHITECTURE

Figure 1, on page 7, shows the two tiers in a client/server architecture. The client tier is the Oracle Forms Runtime, which does the following:

- Loads the FMX file (and other Forms files if applicable, such as PLX and MMX files)
- Connects to, and interacts with, the database
- Runs the Forms logic of the application

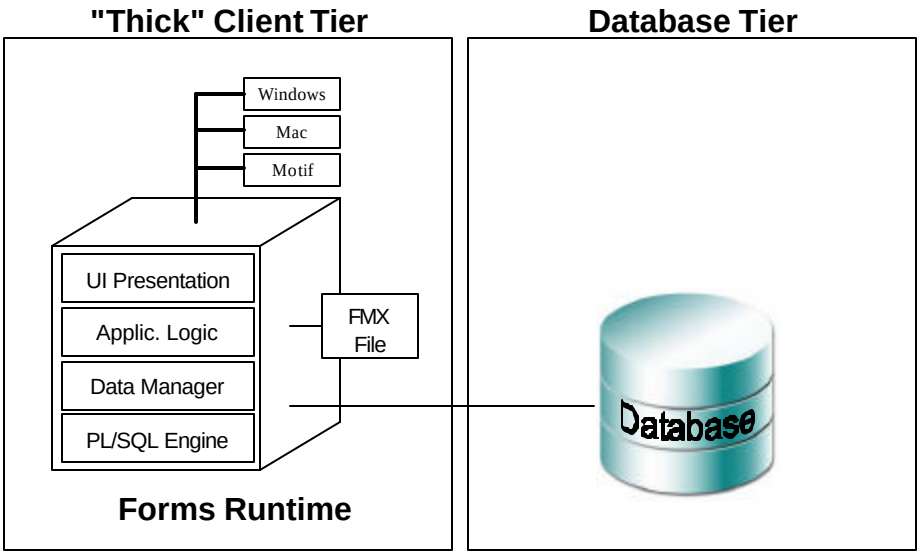

Figure 1: Client/Server Two-Tier Architecture

Oracle Forms has been designed so that, apart from the User Interface (UI) Presentation, all of the components (PL/SQL Engine, Data Manager, etc) are platform independent. It is this strength that allows Oracle Forms to run on many platforms. It is also this strength that helped Oracle Forms take the next evolutionary step and run on the web.

1.2.2 THREE-TIER WEB ARCHITECTURE

In Figure 2, you can see that the only change is where the UI Presentation layer resides.

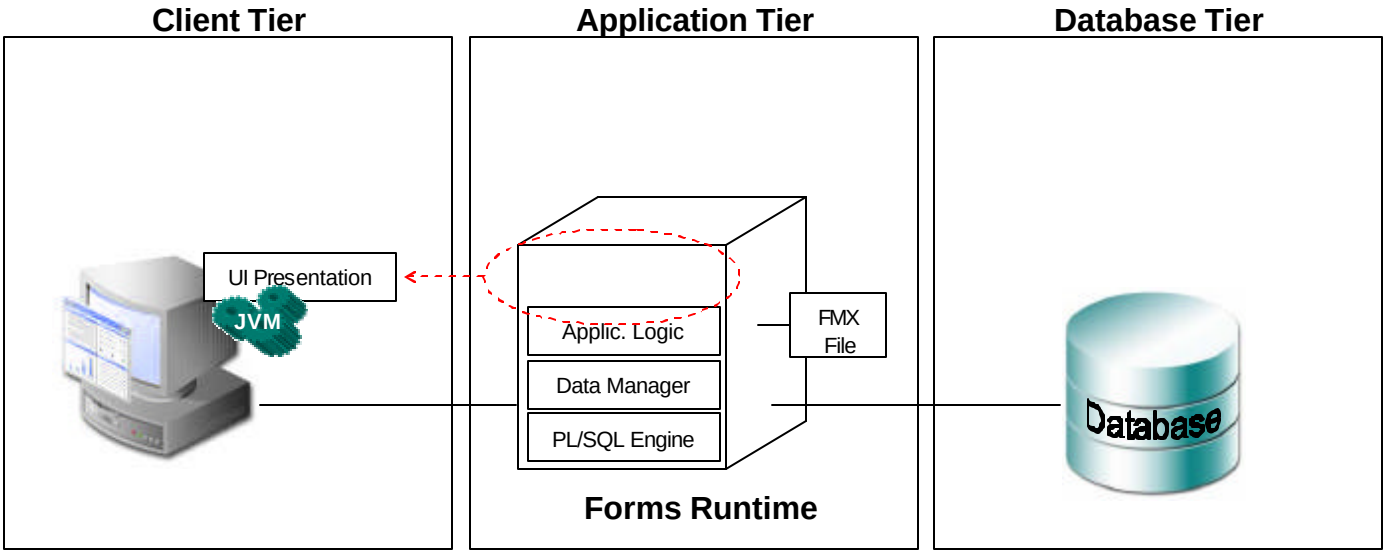

Figure 2: Forms on the Web Three-Tier Architecture

1.2.3 HOW IT WORKS

The crux of the technology is a separation of the logic and the UI. Except for how the UI is displayed, the Forms Runtime is the same in both the two-tier and three-tier architectures. In a three-tier web environment, the Forms Runtime process still loads the FMX file (and other Forms modules), connects to the database, and runs the logic of the application, in exactly the same way as the client/server Forms Runtime.

The logic, or the processing of the Forms FMX file, takes place on the middle tier. The only information sent to the client is what to display on the screen.

1.2.4 THE CLIENT TIER

The Forms Client is a Java Applet that runs in the user's browser. Anyone with a Java-enabled browser can run a Forms application, over any network: internet, intranet, or extranet. It receives messages from the server about what to display, and reports back to the server what actions the user performed, so they can be processed.

The Forms Client is a generic applet that runs all Forms, no matter how small or large, so it only needs to be downloaded once. No business logic is executed on the client; it is responsible for rendering the screen. All logic is processed on the application tier.

Refer to Appendix B for a more detailed look into what happens when you run an Oracle Forms application on the web.

1.2.5 THE APPLICATION TIER

The Forms Client communicates to the Forms Runtime process, not directly, but via the HTTP Listener within Oracle9*i*AS. By default, HTTP communication is used, but you may configure Oracle9*i*AS to use SSL (HTTPS) for secure networking.

The application tier is made up of several components. Appendix B contains more information about how the middle tier works.

> **Note:** Oracle9*i* Forms is only supported on Oracle9*i*AS Release 2. While it may be possible to configure other application servers to run Forms, this is neither certified, nor supported.

1.2.6 THE DATABASE TIER

The database typically runs on a dedicated machine, or on a cluster of machines. A Forms application will usually connect to one database instance. However there may be many different Forms applications running on the one Application Server,

each connecting to a different database.

2 THE UPGRADE PROCESS

There are three ways to upgrade your Forms modules from Forms 6i to Oracle9*i* Forms:

- Using the Forms Builder (Interactive)
- Using the Forms Compiler (Batch)
- Using the Forms JDAPI (Programmatic/Batch)

In addition, there is the optional step of using the Oracle9*i* Forms Migration Assistant to find the features that are obsolete, or which have changed in Oracle9*i* Forms.

For full details on how to migrate your Forms applications, refer to the Forms Migration Book on the Oracle Documentation CD, or downloadable from OTN[1].

2.1 Upgrading Using the Forms Builder (Interactive)

This is the simplest, though least time efficient, method. For each module, FMBs, PLLs, MMBs, do the following:

- Open the module in the Forms Builder
- Compile the module
- Save the module

The advantage of this method is that you receive instant feedback on any upgrade errors when you compile the module. The disadvantage is is that if you have many modules, it can be tedious to do them one at a time. Sections 2.2 and 2.3 describe a way to upgrade using a batch process.

2.2 Upgrading Using the Forms Compiler (Batch)

Use `<OracleHome>/bin/ifcmp90.exe` to upgrade your Form modules in batch mode. It is easy to create a script (such as a DOS Batch script on Windows, or a shell script on Unix, etc) that runs `ifcmp90.exe` for each of your Forms modules. Appendix C has an example script for batch compilation on Windows.

By default, `ifcmp90.exe` only creates the runtime file, leaving the source untouched; for example, an FMX file from an FMB. By specifying `upgrade=yes` as a parameter to `ifcmp90.exe`, the source file will be upgraded to Oracle9*i* Forms as well. This is equivalent to opening the FMB in the Forms Builder, compiling, and then saving. If there were any upgrade errors in a module, `ifcmp90.exe` produces an ERR file with the same base name as the module, which lists all of the errors.

[1] http://otn.oracle.com

The disadvantage of this method is that you have to check for ERR files and resolve the problems (unless you previously ran the Migration Assistant, see section 2.4).

2.3 Upgrading Using the Forms JDAPI (Programmatic/Batch)

It's also possible to write a Java program to migrate your Forms applications using the Forms Java Development API (JDAPI). This provides a programmatic interface of all of the features in the Forms Builder. Thus you could create a Java program to open your modules, compile them, and save them. You could also optionally manipulate the modules programmatically, performing actions that would normally be done manually in the Builder.

This option is for experts: you need to know Java, as well as become familiar with JDAPI. Documentation for JDAPI is available from the Forms area on OTN[1]

2.4 Using the Migration Assistant (Optional)

The Oracle9*i* Forms Migration Assistant is an optional tool to help with migrating from Forms 6i. It can be run on any Forms modules in either interactive or batch mode, and will either make modifications for you, or will notify you of code that needs to be changed or is obsolete.

The Migration Assistant is optional because if any of your modules are invalid, you will be notified when you attempt to compile them anyway. This tool can be helpful by either making the changes for you beforehand, or warning you of potential problems.

You can download the Migration Assistant from the Forms area on OTN. For a list of obsolete features, refer to the Oracle9*i* *Forms Feature Obsolescence* whitepaper, downloadable from OTN.

[1] http://otn.oracle.com/products/forms/content.html

3 BUILDING FORMS FOR THE WEB

This chapter describes some of the issues and changes in development for Forms web applications, compared to building client/server applications. Each issue or feature is classified as one of the following:

- Works differently than you might expect
- Works the same, but may have performance implications
- Doesn't work in a web environment
- Platform-specific

3.1 Features Which Work Differently Than You Might Expect

This section contains features of Forms which still exist and work in a web environment, but which might seem to work differently than in a client/server environment.

3.1.1 HOST AND ORA_FFI BUILT-INS, USER EXITS, JAVA IMPORTER (ORA_JAVA)

These let Forms applications access functionality outside of Forms, on the machine where Forms is running. In a web environment, Forms is running on the middle tier. Therefore, the command will execute, not on the client, which you might be used to from client/server, but on the middle tier. Often, this could be what you want. But if your intention is to do this on the client, then another method is needed.

The alternative depends on what you are trying to achieve. It is possible to use Java to run commands on the client using PJCs or JavaBeans. See Appendix A for a list if papers and examples on using PJCs and JavaBeans in Forms.

3.1.2 INTERACTING WITH THE LOCAL (CLIENT) FILE SYSTEM

Similarly to section 3.1.1, built-ins that interact with the file system will execute on the middle tier, not on the client:

- READ_IMAGE_FILE
- TEXT_IO
- TOOL_RES

You can use JavaBeans on the client to access the client file system. There is a caveat, though: the Java security model does not allow Java Applets access to the local file system, unless that Java is signed. You may sign the Java with a certificate that you created yourself, or using one purchased from a signing authority.

If your application is to be used internally, then you can sign the code using a self-created certificate, since you trust yourself or your company. However, if the

public are going to use your system, then they would probably like a properly verifiable certificate that proves it's really from you.

3.2 Features With Performance Implications

This section lists features which work in the same way as in the client/server environment, but which may have other implications.

See Appendix A on how to download the *Forms Best Practices Guide*, which contains tips and tricks on improving the performance of your Forms applications.

3.2.1 SYNCHRONIZE BUILT-IN

The SYNCHRONIZE built-in results in a roundtrip from the client to the server. Experience shows that many developers overuse this command, thus generating unnecessary network traffic. Evaluate each use of SYNCHRONIZE to determine if it is really needed.

3.2.2 TIMERS

The Forms Client is responsible for maintaining timers, and alerting the server when they expire. This results in a roundtrip from the client to the server. Thus, if you have a timer that executes every second, a message is sent to the server every second. And each concurrent user is doing the same. This could result in unnecessary network traffic.

3.2.3 TABBED CANVASES

If you have a tabbed canvas in your form, then all of the items on all of the tabs are downloaded to the client. This may result in a larger initial download time, especially if the tabbed canvas has many items, and the network is slow.

When the tabbed canvas is first shown, only one tab is visible. Some developers may wish to defer the downloading of items on the hidden tabs until the user actually navigates to them. This will reduce the startup time, but could result in a delay when the user navigates to a new tab for the time, as they must wait until the items are downloaded.

If you wish to implement the deferred downloading of items for tabbed canvases, use stacked canvases. Create a stacked canvas for each tab, and make sure that they are hidden (set Visible to No). Put the items for each tab on the equivalent stacked canvas.

In the WHEN-TAB-PAGE-CHANGED trigger, either do a GO_ITEM to an item on the appropriate stacked canvas, or explicitly show the canvas using SHOW_VIEW. Depending on how your application works, you may need to explicitly show the first stacked canvas so that the items for the first tab are visible.

Using this technique, the items are only downloaded to the client when the stacked canvas is shown for the first time.

3.3 Features Which Do Not Work in a Web Environment

The features in this section are not valid in a web environment. For a full list of obsolete features, refer to the Oracle9*i Forms Feature Obsolescence* whitepaper.

3.3.1 MOUSE EVENTS

Some mouse events are not used in Forms on the web due to the amount of network traffic that would be generated:

WHEN-MOUSE-MOVE
WHEN-MOUSE-ENTER
WHEN-MOUSE-LEAVE

Other mouse events work normally.

3.3.2 ICONS IN OPERATING SYSTEM SPECIFIC FORMATS

Images in a browser are in either JPG or GIF format. This also applies to Forms applications on the web that use icons, such as iconic buttons. In a Forms client/server runtime environment, icons are in ICO format on Windows, or XPM format on Unix. Therefore, icons from a client/server Forms application need to be converted to either JPG or GIF format for Forms on the web.

The Oracle9*i* Forms Builder still uses the ICO/XPM format to display icons in the layout editor. A future patchset will allow the Forms Builder to display JPG and GIF format icons in the Builder. But for now, it means that you need two sets of files for your icons: ICO/XPM format if you wish to see them in the Builder, and JPG or GIF format for runtime.

You always specify icons without an extension in the Forms Builder, so a developer need only specify the name of the icon, without an extension, and Forms will load the appropriate file.

At design-time, the Builder looks for ICO/XPM files in the path specified in the environment variable, `UI_ICON`. (This is usually in the Windows Registry, on Windows platforms.)

At runtime, for Forms on the web, Forms looks in the path specified by a variable in the registry.dat file. This is explained in more detail in Chapter 4.

You may keep your ICO/XPM files in a separate directory to your JPG and GIF files, or they could be in the same directory, there is no rule or recommendation. This means your build-time `UI_ICON` variable might have a different path that what's specified for runtime, in the registry.dat file.

3.3.3 VBX CONTROLS

VBX items are applicable only to 16-bit windows. Moreover, it is an obsolete technology, and is therefore not supported in Oracle9*i* Forms.

3.3.4 ACTIVEX (OCX) CONTROLS

ActiveX Controls (also known as OCX Controls) are applicable to Windows platforms only. Oracle has no plans for ActiveX support over the web. The alternative is to use JavaBeans, which provide similar functionality.

There are many JavaBeans available which offer the same functionality as many common ActiveX Controls (e.g. progress bars). But you may need to write your own JavaBeans to replace the functionality of your ActiveX Controls.

3.3.5 OLE CONTAINERS

Oracle has no plans for OLE container support over the Web. This means using OLE to interact with other applications on the client will not work. e.g. Embedding an Excel spreadsheet in Forms on the client. However, programmatic OLE interaction will still be supported with external OLE servers on the middle tier (on Windows platforms only). See section **Error! Reference source not found.** for more information.

3.3.6 SOUND ITEMS

Sound Items only function in client-server deployment and will not be supported in Oracle9*i* Forms.

3.3.7 IMAGE CONTROL PALETTES

The control panel that can be displayed using the Show Palette property on Image items does not display when web deployed.

3.3.8 `GET_FILE_NAME` BUILT-IN

This functionality can be replaced on the client tier using a JavaBean.

3.4 Features Which Are Platform-Specific

With Forms on the web, it becomes more likely that you could develop on one platform, and deploy to multiple different platforms. Moreover, at runtime, the middle tier could be on one platform, and your clients on multiple different platforms. Care must be taken to handle platform specific issues.

3.4.1 `HOST` BUILT-IN

The HOST built-in executes platform specific commands. For example, the following command would work if the middle tier were on a Windows platform:

```
BEGIN
  host('dir > dirlist.txt');
END;
```

It would fail on Unix, though, since the `ls` command is used for a directory listing on that platform.

3.4.2 CASE SENSITIVITY

Similarly, developers need to be careful with case. Unix is case-sensitive, while Windows is the opposite. For instance, if you develop on Windows and attach a library, and then deploy to the middle tier on Unix, the filename on both platforms should have the same case, or else the library won't be found.

3.4.3 FONTS

Because the Forms application is rendered inside a Java Applet, only Java fonts are supported. This means that when developing an application, you need to ensure that only supported fonts are used. The following Java fonts are used in Forms:

MonoSpaced
Dialog
DialogInput
Serif
SansSerif

Fonts used in Forms are mapped to the Java fonts in the REGISTRY.DAT file. There is a default Java font specified as well, in case you use a font that Forms doesn't recognize.

3.4.4 EXTERNAL OLE SERVERS

Programmatic OLE interaction will be supported with external OLE servers on the middle tier (on Windows platforms only). That is, you may use the OLE built-ins in Forms to make calls to an externally activated OLE server, which runs on the middle tier.

4 CONFIGURING AND DEPLOYING FORMS SERVICES

Oracle9*i*AS is autoamatically configured to use Forms Services out-of-the-box, during installation. The only configuration you need to worry about is your application configuration.

Configuring Forms for client/server runtime is different than for Forms on the web. For client/server, you only have to set the appropriate environment variables (or use the Windows Registry on Windows platforms). With Forms on the web, you are centrally managing your Forms installation on an Oracle9*i* Application Server.

There are three files to configure Forms Services for runtime deployment. They reside centrally on the server, so only need to be configured in one place, versus on every machine in a client/server environment:

- <OracleHome>/forms90/server/formsweb.cfg
- <OracleHome>/forms90/server/default.env
- <OracleHome>/forms90/java/oracle/forms/registry/Registry.dat

The names are configurable, but these are the default. The combination of these three files allows you to have many different runtime environments from a single installation.

This chapter is only a guide. It is for those who are familiar with Forms client/server but don't know where to begin when configuring and deploying Forms on web. Each file contains comments, so you can also learn about them by looking at the files themselves. For full details, refer to the Forms documentation.

4.1 formsweb.cfg

With the Oracle Forms client/server Runtime, you could specify many options on the command line when launching a Forms application. With Oracle Forms on the web, you specify runtime parameters in the `formsweb.cfg` file.

This file controls the settings for each Forms application. It has a default section, plus optional sections for each application, to override the default settings. The following is a snippet of some of the entries in the `formsweb.cfg` file:

```
userid=
form=test.fmx
background=
width=640
height=480

[summit]
userid=summit/summit@forms1-pc
```

```
form=customers
background=summit.jpg
width=994
height=582
```

The first settings are the default settings. All Forms applications will use them, unless specifically overridden.

The `[summit]` represent application-specific settings, and overrides the defaults, in this case for the Summit application. For instance, by default, a form will have no background. When the Summit application is run, the background will be the image `summit.jpg`.

(Note that `background` here refers to the background in the Forms Client Java Applet, not the background canvas within the Form itself. Similarly, the `width` and `height` parameters refer to the width and height of the Java Applet, not the width and height of the Forms windows.)

When you run a Forms application, you can specify the section to use from this file with the config parameter. For example:

http://myserver.com/forms90/f90servlet?config=summit

This is explained more in section 4.4, Deploying Forms to the Web.

While it is convenient to name the application section (`[summit]`) in `formsweb.cfg` the same as the application itself (Summit), it is not necessary. Several different applications may use the same application section, for example.

4.2 default.env

This is where environment variables are specified. The client/server equivalent is to specify the environment variables in the Windows Registry on Windows platforms, or on the command line on Unix.

Some variables that you might typically want to set are:

- ORACLE_HOME
- FORMS90_PATH
- CLASSPATH (if you are using Java)

By default, the `default.env` file gets sourced each time a user starts a Forms application. However each application can choose to use its own ENV file with the `envFile` setting:

```
userid=
form=test.fmx
background=
width=640
height=480
```

```
envFile=default.env

[summit]
userid=summit/summit@forms1-pc
form=customers
background=summit.jpg
width=994
height=582
envFile=summit.env
```

4.3 Registry.dat

This file lets you specify settings for controlling icons within Forms at runtime. (It is also for controlling fonts at runtime. See section 3.4.3 on page 16 for more information.) This is the client/server equivalent of the `UI_ICON` environment variable.

You specify where Forms can find icons with the `default.icons.iconpath` setting:

```
default.icons.iconpath=C:\MyApp\icons;C:\MyOtherApp\icons
```

Each application can choose to use its own registry file with the `serverApp` setting:

```
userid=
form=test.fmx
background=
width=640
height=480
envFile=default.env

[summit]
userid=summit/summit@forms1-pc
form=customers
background=summit.jpg
width=994
height=582
envFile=summit.env
serverApp=/summitapp/summit_registry
```

`ServerApp` specifies the registry file, without extension, using a URL. In this example, it is a relative URL, where `/summitapp/` is defined as a virtual directory in the web server configuration files.

4.4 Deploying Forms to the Web

Once you have your Forms modules upgraded, and you have configured Forms Services, you can run your applications. To do so, you enter the URL for your application in a browser. The URL will look something like the following:

http://myserver.com/forms90/f90servlet?config=summit

The Oracle9*i* Application Server is configured to route these requests to the Forms Servlet. The "`config=summit`" is a parameter to the Forms Servlet to tell it to look up the `[summit]` section in `formsweb.cfg`, and use those settings. That is how Forms Services knows which form to run, which userid to use, etc.

The `config` parameter is optional. If omitted, Forms Services will use the parameters from the default setting of the `formsweb.cfg` file. From the example in section 4.3, for instance, if the `config` parameter were not present, the form `test.fmx` would be run.

You may also override many of the settings from `formsweb.cfg`, on the URL. For example, the following URLs are all valid:

- http://myserver.com/forms90/f90servlet?form=myform.fmx
  Use the default settings, but run the module `myform.fmx` instead.
- http://myserver.com/forms90/f90servlet?config=summit&form=summitTest.fmx
  Use the settings specified in the `[summit]` section, but use a different form: `summitTest.fmx`.
- http://myserver.com/forms90/f90servlet?config=summit&form=summitTest.fmx&usreid=scott/tiger@test
  Same as the previous example, except connect to a different database.

The `formsweb.cfg` file contains comments specifying which parameters may be overridden by the user, on the URL. The others may only be set by the administrator, and cannot be overridden on the URL.

If no `userid` is specified, either in `the formsweb.cfg` file, or on the URL, then the user will be prompted to enter a username and password.

5 TAKING ADVANTAGE OF THE WEB

Now that you are deploying your Forms applications to the web, you should take advantage of some features that are not available to client/server applications:

- Extensible Java Client
- Single Sign-On
- Launch Web Pages

5.1 Extensible Java Client

It is also possible to extend the functionality of the Forms Client using Java in two ways:

- Pluggable Java Components (PJCs)
- JavaBeans

5.1.1 PLUGGABLE JAVA COMPONENTS (PJCs)

Because the Forms Client is a Java Applet, you can take advantage of Java's powerful inheritance and extensibility model to extend the functionality of the Forms Client. Oracle has exposed certain hooks in the Forms Client that allow developers to change the way that it behaves.

The functionality of most of the GUI elements within Forms can be modified using Java. Here are some examples of why you might want to do this:

- You could turn your buttons into rollover buttons. Whenever the mouse moves over a button, the image on the button changes, in a similar way to many web pages.
  This is done by writing some Java that watches for when the mouse moves over the button, and changing the image on button. All of this takes place on the client, and doesn't require any roundtrips to the middle tier, apart from downloading the images in the first place.
- You could have validation on a text field that monitors key strokes, and reacts to them. Forms natively has immediate validation, but only when the user navigates out of the field. However with Java, you could even have keystroke validation as the user types.
  Again, because the Java is run on the client, this occurs without any roundtrips to the server.
- It's possible to put a filter on a text item more sophisticated than that provided natively by Forms. In conjunction with the previous example, it could react immediately, as the user types. For example, the filter could spell check, and make corrections, even as the user types!

All of these examples, and more, can be downloaded from OTN so you can see how they work. See Appendix A for details.

5.1.2 JAVABEANS

JavaBean support has been enhanced in Oracle9*i* Forms to make using them easier. JavaBeans are Java classes that adhere to certain coding specifications. They are components, usually offering specific functionality, which can be combined to build applications, or be embedded into applications. They are analogous to ActiveX (OCX) Controls on Windows.

You can include JavaBeans in the Forms Client to add functionality to Forms. They are often visual in nature, such as a progress bar, but need not be. Some examples:

- Use a calendar JavaBean to display a calendar on the client. Unlike a calendar implemented using Forms, a JavaBean calendar would not require any roundtrips to the server, since the JavaBean executes on the client.
- Display a ticker clock on the client.
- Get client information such as machine name, IP address, and local operating system username.

All of these examples, and more, can be downloaded from OTN so you can see how they work. See Appendix A for details.

5.2 Single Sign-On

Forms Services integrates with the Single Sign-On (SSO) capabilities of Oracle9*i*AS. By using SSO, users will only have one username and password to remember. Moreover, once you logon to any application supporting SSO, you can run any other application that also supports SSO without having to logon again.

The good news is that existing Forms applications can take advantage of SSO without any modification! Configuring Forms to use SSO is done outside of the Forms modules so is transparent to your applications.

See Appendix A for references to Single Sign-On.

5.3 Launch Web Pages

Using the `WEB.SHOW_DOCUMENT` built-in, it is possible to launch a browser from within Forms to show a specific web page. The URL can be built up dynamically, allowing you to invoke web pages depending on the data in the Form or what the user is doing.

For example:

- If your application has a multi-row block that lists products, you could have a button next to each row to bring up a web page containing information about that product.

- You could invoke context sensitive help which brings up a certain web page, depending on where the cursor is on the screen.
- Using Java on the client, you could have hypertext within your Forms application that launches a browser when a user clicks on it. It will even show the hyperlink hand icon when the mouse hovers over the link, just like in a browser.
  There is demo available that shows this which is downloadable from OTN. See Appendix A for details.

You are not constrained to http: URLs. You can invoke a browser with any URL that your browser supports, such as ftp:, JavaScript:, etc.

6 CONCLUSION

As you can see from this whitepaper, the migrating your Oracle Forms applications from client/server to the web is an easy process. In many cases, it is not even a migration, but more of a new deployment option – your business logic and functionality are completely preserved allowing you to take advantage of the many benefits of running on the web.

APPENDIX A – FURTHER READING

This appendix lists papers or information referred to by this document.

The following references and resources may be found on the Oracle Technology Network (OTN): http://otn.oracle.com.

- The Migration Book may be found in the Oracle9*i*AS documentation set[1].
- The Oracle9*i*AS documentation set contains information on Single Sign-On. OTN has an area for Oracle Internet Directory[2] information. [??? Chase up URL for SSO Server with John Heimann.]

The following items may be found in the Forms area of OTN[3]:

- The whitepaper *Customer Testimonial: Retek – A Benchmark Comparison of Client/Server and Web Deployment*.
- The *Forms Best Practices* Guide.
- The Oracle9*i* Forms Migration Assistant, as explained in section 2.4 on page 11.
- The *Oracle9*i *Forms Feature Obsolescence* whitepaper, which details the features that have been removed in Oracle9*i* Forms.
- The *Oracle9*i *Forms Architecture* whitepaper.
- The whitepaper *Forms and Desktop Integration on the Web* contains detailed information on integration points between the Forms Client and the desktop, such as PJCs, JavaBeans. Some of these were summarized in Chapter 3.
- There are many examples available in the Sample Code section of Forms on OTN[4]. Topics include Pluggable Java Components (PJCs) examples, JavaBean examples, and many more.
- JDAPI Documentation.

[1] http://otn.oracle.com/docs/products/ias/content.html
[2] http://otn.oracle.com/products/oid/content.html
[3] http://otn.oracle.com/products/forms/content.html
[4] http://otn.oracle.com/sample_code/products/forms/content.html

APPENDIX B – ANATOMY OF A FORMS SESSION

If you are coming from client/server background, it might be useful to understand how a Forms application can run over the internet or over an intranet.

Figure 3 shows the architecture of deploying a Forms application. This is a three-tier architecture, with a client tier, a database tier, and the application tier being composed of the HTTP Server and Forms Services on two separate machines. (The middle tier does not necessarily mean a single machine.)

Figure 3 shows only one possibility. Other combinations include having Forms Services on the same machine as the HTTP Server, or having multiple machines with Forms Services, and load balancing over them. In all cases, the components are the same, and must exist (HTTP Server, Servlet Engine, Forms Servlet, Forms Listener Servlet, and Forms Runtime Processes). The only difference is how many there are, and on which machines they run.

Also note from Figure 3 that the user may be on the same intranet network as the middle tier, or they may be accessing Forms over the internet, through firewalls and proxies. It is irrelevant to Forms how the client accesses the middle tier.

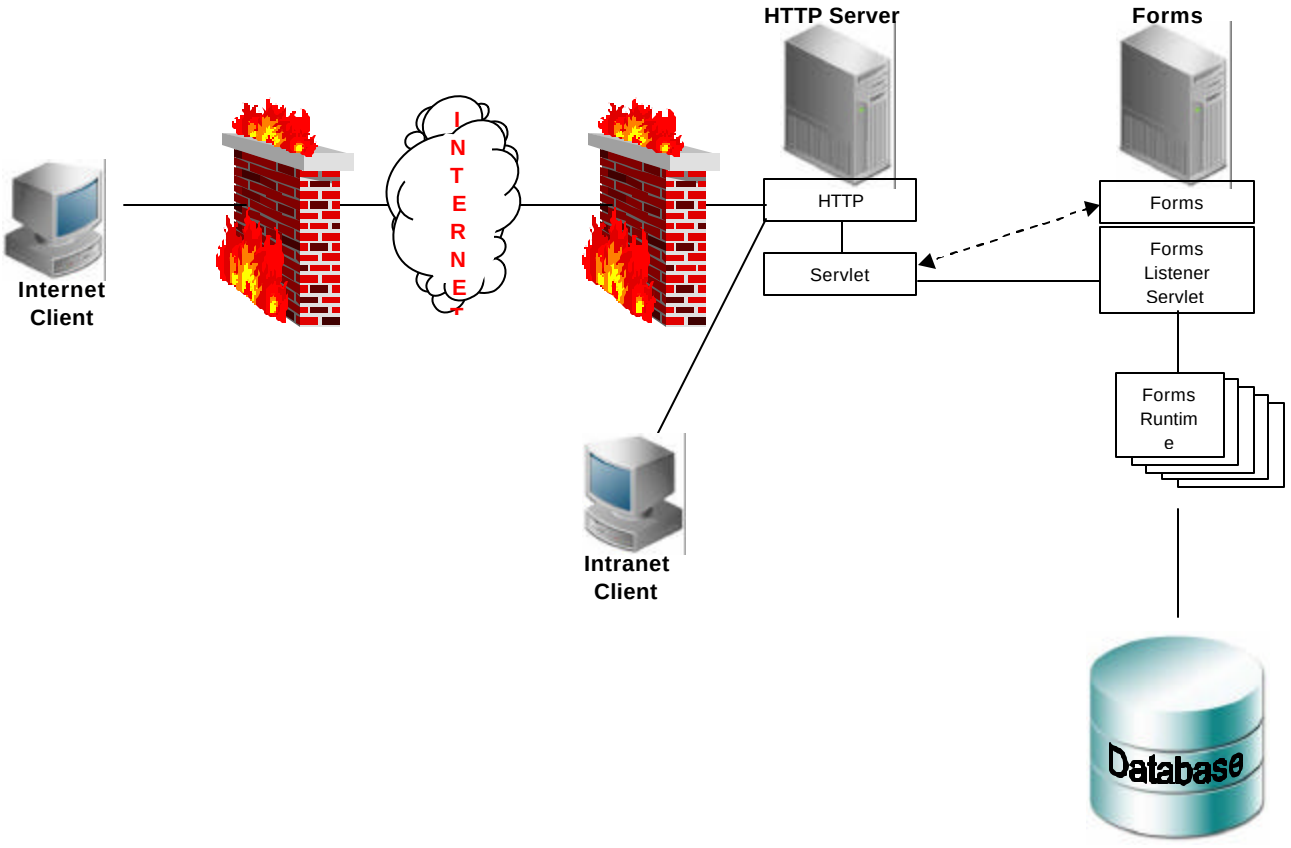

Figure 3: Oracle9*i* Forms Deployment Architecture

Here, at a high-level, is what happens when you a run on the web (refer to Figure 3 as you read these steps):

1. The user starts up a browser and navigates to the advertised URL for the Forms Application, just like they would for any other URL: by clicking a link, choosing it from a favorites list, typing it manually, etc.
2. The application server (the middle tier) responds with an HTML page that contains a Java Applet. This is the Forms Client. There is only the one Applet which runs all Forms, no matter how small or large. In addition, the Applet is cached on the client and need only be downloaded once.
3. The Forms Client starts and connects to the server, via the application server. All Forms communication goes through the application server, just like any other web application would, which means Forms will work through proxies and firewalls, over the internet.
4. The server sees that the user is starting the application, and does two things: creates a "servlet session", and starts a Forms Runtime Process.
5. The Forms Runtime Process is the engine that actually runs the application. Each user has their own Forms Runtime Process, which is terminated when the session ends.

   The Forms Listener Servlet is responsible for making sure that each request from the client gets routed to the correct Runtime Process, and that the responses from the Runtime Process gets sent to the correct client.
6. All of the Forms logic is executed by the Forms Runtime Process on the server. So the Forms Runtime Process opens the appropriate FMX file, connects to the database, and does everything needed to start the form. It puts the cursor in the first field and notifies the client. The information sent to the client is the layout of the screen and the cursor position.
7. Only the UI information is sent to the client. This is the biggest key to understanding how Forms on the web works: *No logic executes on the client.* The client displays the screen, with the cursor in the first field, and the user can begin to interact with the form.
8. Suppose the user types in a value and navigates to the next field. The client sends a message to the server with the value and what the user did. The server processes this and sends a response to the client as to what it should display now.

For example, there might be a WHEN-VALIDATE-ITEM trigger on the first field which failed, and re-directs the cursor back to it. The Forms Runtime Process sends a message relaying this information to the client. Remember, the client doesn't execute any Forms logic. It simply updates the relevant area on the screen, and puts the cursor in the field that it was told to. Now the user can continue.

Each major action on the client, such as navigating between fields, choosing an item from a list, etc, results in a roundtrip to the server to be processed by the Forms Runtime Process.

The Forms Servlet and the Forms Listener Servlet

Because of the similarity of their names, it is easy to get the two servlets provided by Forms confused.

The Forms Servlet is used only for the initial connection when a user starts a Forms application. It returns the HTML file to the client which contains the applet that is the Forms Client.

Once the Java Applet has started and a session is created, all of the communication goes through the Forms Listener Servlet.

What is a servlet?

Servlets are modules of Java code that run in an application server to answer client requests. Since they are written in the highly portable Java language and follow a standard framework, they provide a means to create sophisticated server extensions in a server and operating system independent way. They add functionality to an application, similarly to how JavaScript adds functionality to an HTML page.

Some example uses for HTTP Servlets include:

- Processing and/or storing data submitted by an HTML form.
- Providing dynamic content, e.g. returning the results of a database query to the client.

Managing state information on top of the stateless HTTP, e.g. for an online shopping cart system which manages shopping carts for many concurrent customers and maps every request to the right customer.

APPENDIX C – BATCH COMPILATION

The following is an example script for batch compilation on Windows. Copy the text into a BAT file in the top-level directory where your Forms 6i modules reside.

```
@echo off
if "%1" == "" goto ERROR

REM Forms
for /R %%F in (*.fmb) do start /w ifcmp90.exe userid=%1 batch=y module=%%F

REM Menus
for /R %%F in (*.mmb) do start /w ifcmp90.exe userid=%1 batch=y module=%%F module_type=menu

REM Libraries
for /R %%F in (*.mmb) do start /w ifcmp90.exe userid=%1 batch=y module=%%F
module_type=library

goto exit

:ERROR
echo Userid/password must be supplied
pause

:EXIT
```

White Paper Title
March 2002
Author: Robin Zimmermann
Contributing Authors: Grant Ronald, Duncan Mills

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.