Oracle9*i* Application Server Release 2, Forms Services Overview

An Oracle Whitepaper
May 2002

ORACLE

Oracle9*i* Application Server Release 2, Forms Services Overview

INTRODUCTION

Oracle9*i* Forms contains the next generation of Forms Services, a proven technology helping you to deploy your applications written in Forms to the Web. Oracle9*i* Forms is released in two parts: a development environment, Forms Developer, and a deployment environment, Forms Services.

Forms Services, a part of Oracle9*i* Application Server Release 2 (Oracle9*i*AS), offers built-in performance and scalability features that are adopted immediately by any Forms application. Oracle9*i* Forms can be deployed to the Web only and does not include a client/server runtime environment to run Forms applications on a local desktop installation.

The application building environment, Oracle9*i* Forms Developer, provides a declarative IDE using PL/SQL as its programming language. With Oracle9*i* Forms Developer, application developers build attractive Java UIs without writing a single line of code in Java. This capability allows companies to leverage their existing RAD development skills when building Web applications. The Web interface of an application built with Forms Developer is a generic, 100% Java applet.

Oracle9*i* Forms Developer is part of Oracle9*i* Developer Suite (Oracle9*i*DS) Release 2 and is described in the technical whitepaper *Oracle9iAS Forms Services and Oracle9i Forms Developer*. The primary focus of this paper is Oracle9*i*AS Forms Services.

AN OVERVIEW OF ORACLE9*I* FORMS ON THE WEB

The Oracle9*i* Forms application source code is interpreted during runtime. Previous releases used a client/server–based runtime environment to execute the business logic stored in Forms application modules and to render the user interface with graphical UI elements of the underlying operation system.

Oracle9*i* Forms still uses a runtime interpreter but replaces the client/server runtime engine with a Web engine that renders the application interface using a generic set of Java classes. This means that you can use the same code when

deploying Forms on the Web. The Forms Web application is then accessible by a single URL typed into the Web browser's URL field.

What is Oracle9*i*AS Forms Services?

Oracle9*i*AS Forms Services is the collective name for the Forms components used to deploy a Forms application on the Web: Oracle9*i*AS Forms Services is a comprehensive application framework to deploy enterprise-class applications on the Internet with a rich Java interface. The Forms Services Web client, a generic Java applet that, after being loaded, renders any number and sizes of Forms applications

- The Forms servlet, accepting incoming Web requests and returning the Forms applet HTML start page
- The Forms Listener Servlet, dispatching the communication between the Forms generic Java client and the runtime process on the middle-tier server
- The Forms Web runtime, executing the business logic stored in Forms application modules and performing the database connect

Oracle9*i*AS Forms Services is an easy-to-learn architecture allowing you to deploy Java applications based on Forms to the Web.

Overview of the flow of control in a Forms Web application

When a user requests a Forms application, the following process flow is set in motion:

1. A user requests a Forms Services application by typing the application's access URL into the browser's URL field. The URL points to the Oracle HTTP server (OHS) *powered by Apache* , accessing the Forms servlet.
2. The Forms servlet, a part of Forms Services, dynamically renders a start HTML page for the Forms applet that contains all the parameter attributes required by the requested Forms application. After it is downloaded to the user's browser, this HTML page initializes the download of the Forms Java client, rendered by a generic Java applet.

   The Forms Java client executes in a certified Java Runtime Environment (JRE), which can be the Java Plug-in from Sun, Oracle Jinitiator, or the Microsoft Internet Explorer native Virtual Machine*.

   After the initial download from the server, the Java classes used on the client side are permanently cached on the user's browser , eliminating the need for the same files to be downloaded again for subsequently started

* The Java Plug-in from Sun and the Microsoft Internet Explorer native VM are not certified with the base release of Oracle9*i* Forms.

Forms applications. The Forms Java client is downloaded again only if a newer version is detected on the middle-tier server, ensuring that the client is always using the latest version of Forms classes.

3. The Forms Java client accesses the Forms Listener Servlet on the middle tier to start up a new Web runtime process. The user session is bound to the runtime process with a session cookie created by the Forms Listener Servlet. The session cookie is either kept in memory or attached to the URL of each request and response. It is never written to the client desktop.

4. A Web runtime process is started on the middle-tier server by the Forms Listener Servlet for every user session. It is this runtime process that connects to the database.

   The data used by a Forms Web application is thus not directly loaded to the user's browser but exchanged between the middle-tier server and the database server, greatly improving performance and reducing network traffic. The Forms Web runtime communicates with the Java client throughout the application session using the HTTP or the HTTPS protocol.

5. The Forms Java client uses a smart communication mechanism when talking to the server, reducing the network usage to a minimum.

   Rather than sending the full text of each message, the Forms Web engine compares the content of each message with that of the message sent previously , deriving the difference between the two. It is this difference, then, the delta, that is sent over the network to the client.

6. The Forms user interface is rendered as a 100 % Java applet using a fixed set of generic Java classes. This same set of Java classes is used to render any number and any size of Forms applications, with no additional downloads required.

   The Forms Java UI provides the same quality, the same look and feel, and the same user experience as a client/server application.

Figure 1 captures this process flow.

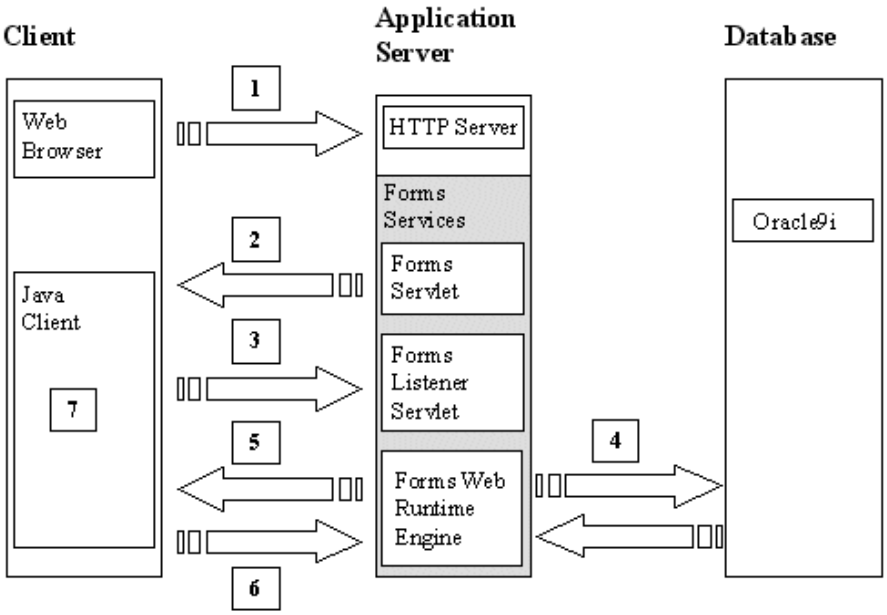

Figure 1: Requesting a Forms Services application

The benefits of Forms Services

Forms Services is a highly optimized architecture designed to effectively deploy Forms applications on the Web.

With its sophisticated architecture, Forms Services provides many unique benefits to Forms applications deployed on the Web: With Forms Services, a developer need not program in Java to deploy real Java applications.

- Once downloaded to the client browser, the generic Forms Java client renders any number of Forms applications, of any size, without requiring additional downloads.
- Forms Services uses smart metadata for the communication between Java client and Web runtime, reducing the network traffic to a minimum.
- Forms Services provides integrated scalability and performance optimization. Ongoing enhancements to the Forms Services architecture are automatically incorporated into existing Forms applications on the Web.
- Forms Services applications can be deployed to the Intranet, the Extranet, and the Internet.
- Forms Services integrates into the Oracle9*i* Application Server architecture, which means that there is no additional application code required to use, for example, the Oracle Single Sign-On solution.
- Oracle9*i* Forms allows a developer to extend the Java client.

- Forms Services is proven technology.

What if you are running Forms client/server applications today?

Oracle is aware that many Forms customers still require a Forms client/server runtime engine to run business applications built with a previous release of Forms. For this reason, client/server support for Forms6*i* will remain until 2008.

When enabling their Forms applications for the Web, the first issue for customers is often that they don't know enough about how to move a client/server application to the Web. If you haven't worked with Web-enabled software before, then the Web might appear incomprehensive to you, making it impossible to plan the migration project , either in terms of money or time. Although this paper is not a guide to Forms Web migration, it makes sense to give you a brief idea of what is needed when moving your Forms client/server application to the Web.

- The same Forms fmx modules used to run an application in client/server environments are used to run Forms on the Web.
- Forms applications have the same behavior on the Web as they have in client/server environments.
- The application logic resides on the middle tier, guaranteeing a better performance.
- PL/SQL is still the programming language of Forms9*i* on the Web. The database connection is performed using SQL*Net on the middle tier.
- The graphical user interface, though rendered with a generic Java applet, maintains the same look and feel as in client/server environments.
- On the Web, icons are used in a `.gif` or `.jpg` format rather than in the .ico format. If you use standard Forms icons, you won't need to make any changes. If you use custom icons, you'll need to use a conversion tool to migrate `.ico` files to either the `.gif` or `.jpg` format.
- When Forms applications are run on the Web, OS access is performed on the middle-tier server, rather than on the client desktop as in client/server environments. A JavaBeans solution is provided as a sample to access client-side files from within Forms on the Web.
- If your application uses OS authentication when connecting to the database, you can replace it with the Single Sign-On solution provided in Oracle9*i*AS Release2, without the need for additional coding.
- Once Oracle9*i*AS Release 2 is installed, your Forms Services setup is configured for you to use out-of-the box.
- Calls to integrated Oracle Reports need to be passed to Reports Services, reusing the same Reports definition files as used in client/server environments.

TECHNICAL OVERVIEW OF ORACLE9/AS FORMS SERVICES

Oracle9*i* Forms Services is a three-tier architecture that can easily be extended to a four-tier architecture where required.

Before you can deploy an Oracle9*i* Forms application on the Web, Forms Services must be installed on the middle-tier server, the Application Server. Oracle9*i*AS Release 2 automatically installs Forms Services, providing you with an out-of-the box deployment configuration and setup.

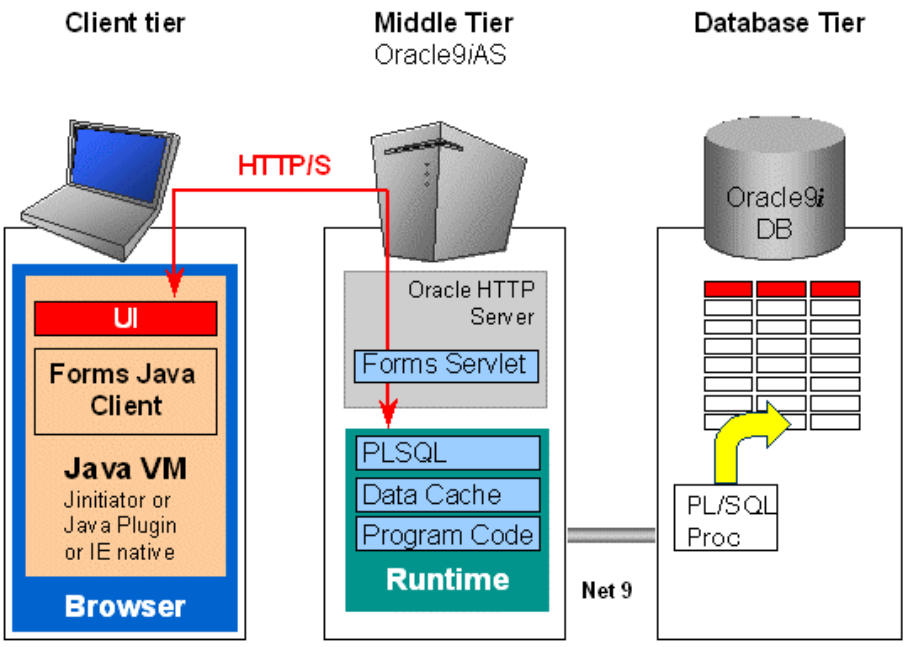

Figure 2: The three-tier architecture of Forms Services

To start a Forms Web application, a user enters the URL for the application into the browser's address field. The request is first handled by the Oracle HTTP server *powered by Apache*, a part of Oracle9*i*AS Release 2, which then routes the incoming Forms application request to the Forms servlet. There is no special port required when running Forms Web applications using either the HTTP or the HTTPS protocol.

Forms Servlet

The Forms servlet detects the user's browser type, Microsoft Internet Explorer or Netscape Navigator, and the settings for preferred languages, before rendering the applet start HTML file, downloaded to the client in response to the application request. The application start parameters, contained in the start HTML <Applet> tag, are read from a server-side HTML template and a configuration file containing application-specific configurations. The Forms servlet simplifies the administration of different Forms applications handled by one Forms Services installation.

Forms Java Client

The Forms Java client consists of a set of generic Java classes that are downloaded to the client the first time a user requests a Forms application. Once loaded, the

generic Java classes of the Forms client are cached permanently on the user's browser, to be renewed only when a new Forms client version is detected on the Application Server.

The Forms Java client is rendered as a 100% Java applet and executed in the browser's Java Virtual Machine (Java VM), which can be the MSIE native VM*, Sun's Java Plug-in, or Oracle JInitiator.

Extending the Forms Java Client

Pluggable Java Components (PJCs) is an extension of the Forms Web user interface, allowing you to customize the appearance of Forms graphical UI elements like buttons, checkboxes, and text fields. A PJC does not affect the underlying transactional framework of Forms.

JavaBeans can be added that provide more functionality to the Forms client, for example, by defining a facility to upload client-side files to the middle tier. A collection of sample JavaBeans is shipped with the Forms9*i* demos.

If you are not yet a Java programmer, you might want to take advantage of the Forms9*i* Bean Manager, which allows you to communicate with a JavaBean using PL/SQL. With the Bean Manager, you can integrate JavaBeans into your Forms applications without writing any Java code.

Forms Listener Servlet

The Forms Java client communicates back to the middle tier, accessing the Forms Listener Servlet to start up a dedicated Web runtime process for each user. The Forms Web runtime is started in an environment defined by a configuration file individual for all custom applications. The same application can then, if required, be started in different environments, supporting the deployment of multilanguage Forms applications on the Web.

Forms Web Runtime

The business logic itself is saved in Forms `.fmx` files and PL/SQL library files. The Forms Web runtime executes both file types and connects to the database using Net9. While it executes the business logic, the Web runtime sends metadata messages to the client instructing it to render the user interface. Sending key-value paired metadata rather than full text messages helps to decrease the network traffic produced by the architecture of Forms Services.

Performance and Scalability

Because all Forms application modules are executed on the middle-tier server, Forms Services automatically shares static application resources, thus reducing the memory requirement of each individual user process. Data individual to a user process (for example, data cached from the database) is stored in its own location in memory, while all immutable code (such as boilerplate labels, library codes, and program codes) is stored in a common area to be reused by all processes of the same application. In this way, most of the memory that a Forms application uses is taken from shared memory.

---

* The Java Plug-in from Sun and the Microsoft Internet Explorer native VM are not certified with the base release of Oracle9*i* Forms.

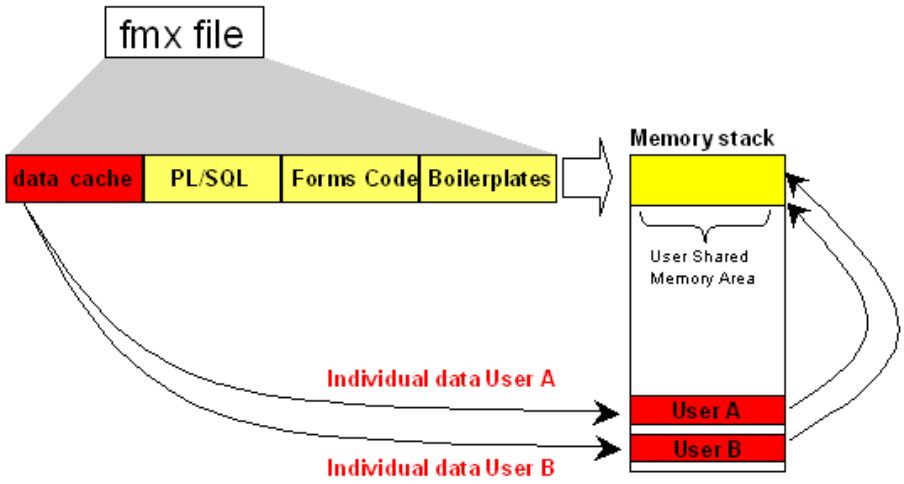

Figure 3: Forms Services uses a shared memory area for static application code

Communication between the Forms Java Client and the Forms Web Runtime

The communication channel between the generic Forms Java client on the user browser and the Forms Services runtime on the middle tier is managed by the Forms Listener Servlet . The communication protocol used is either HTTP or HTTPS, enabling you to run a Forms Web application on both the Internet and any intranet. If you are using HTTPS, no additional SSL certificate is required to run a Forms Web application. Forms Services will use the same certificate that is used for all your applications.

Communication with the database follows the SQL*Net protocol, allowing you both to query data and to access database stored procedures. If a Forms application built for a client/server environment implements database stored procedures in a standard fashion, for example, to perform calculations, those procedures will function in the same way on the Web.

The difference between Forms Services and client/server Forms

So, what is the architectural difference between Forms applications on the Web and Forms applications executed in client/server environments?

The primary difference is that the application's user interface is rendered in Java rather than with the operation system's native graphical components, components such as Motif on UNIX or Microsoft foundation classes on Windows. Another difference is that the application logic is executed on the middle tier rather than on the client, which improves database access performance. This improvement in performance is automatically gained if the Application Server is located next to the

database server machine, reducing the distance that data has to be transported via SQL*Net.

ORACLE9*i*AS FORMS SERVICES INTEGRATION WITH THE APPLICATION SERVER

In Oracle9*i*AS Release 2, all contained components are designed to integrate with each other, forming a unique Application Server offering.

Single Sign-On

Oracle9*i*AS Forms Services leverages the Oracle9*i*AS Single Sign-On server and the Oracle9*i* Internet Directory (OID) for single sign-on. All Oracle Forms applications can participate in this Single Sign-On feature without any additional code. Single Sign-On is activated when Oracle9*i*AS Forms Services is installed, enabling you to run any number of Forms applications using only one username/password pair.

Enterprise Manager

Figure 4: The Enterprise Manager console for Oracle9*i* Forms

The Oracle9*i* Enterprise Manager can be used to manage Forms Services installations. The Enterprise Manager provides information about memory and CPU usage in a Web-based HTML console, and it can be used to start the Oracle9*i* Forms runtime diagnostics.

Java

Oracle9*i*AS provides a Java runtime environment for Enterprise JavaBeans, Web services, and other Java applications. Forms Services, in turn, offers you easy access to these components through the Forms Java Importer. The Java Importer

generates PL/SQL methods for existing Java methods, allowing you to use PL/SQL in Forms applications to call Java. By generating PL/SQL methods for existing Java methods, the Java Importer allows you to call Java in Forms applications using PL/SQL.

OC4J

Oracle Containers for J2EE (OC4J) is the powerful new servlet engine in Oracle9*i*AS. Forms Services supports the new servlet engine by using the new Apache `mod_oc4j` module. This module integrates the OC4J servlet engine with Apache, as did Apache Jserv previously.

Load balancing

Load balancing in Forms is managed by Oracle9*i*AS and Enterprise Manager. Because Forms Services uses OC4J as a servlet engine, it makes use of the 9*i*AS load-balancing feature to balance application load across servlet engines and Application Server machines.

CONCLUSION

Oracle9*i* Forms Services is the new Web deployment engine for Forms applications.

The Web architecture of Oracle 9*i* Forms is designed for high performance and scalability, allowing you to run Forms applications built for client/server deployment on the Web. Oracle9*i* Forms is currently the fastest, and easiest, way to build new Java-based Web applications, without writing any Java at all.

Over the years, Forms software has proven itself a leader in the field of building enterprise business applications. Recently, Forms6*i* Services has been very successful in deploying Forms applications on the Web, as customer references published on http://otn.oracle.com attest to. Now, as the successor to Forms6*i* Services, Oracle9*i*AS Forms Services brings Forms into a pure Web paradigm, for better and cleaner integration with the Web.

Oracle9*i* Forms allows you to quickly, easily, efficiently build scalable Java Web applications that support large numbers of users. If time is a constraint in your project, and scalability and performance are exit criteria, then you will be well served by Oracle9*i*AS Forms Services.

Oracle9*i* Application Server Release2, Forms Services Overview
May 2002
Author: Frank Nimphius
Contributing Authors: Regis Louis, Robin Zimmermann, Grant Ronald

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.