Migrating SQL*Forms 3.0 Applications to Internet Computing™

An Oracle Technical White Paper

March 2000

ORACLE

THE INTERNET PLATFORM

In order to be successful companies need to increase both their competitive advantage and the efficiency of their business, while at the same time increasing the quality of the service they provide and to reduce the cost of their infrastructure.

Though the Client/Server computing model has been available for over 15 years, many large scale transactional applications still reside in the domain of the character mode terminal. The migration to a graphical client/sever style deployment seen as either too difficult, or too costly to implement. As the new century begins however, the need for Y2K compliance and the desirability to remain competitive is forcing many organizations to review their application infrastructure. Character mode computing offered an extremely cost-effective solution for deployment, but due to limited interface capabilities resulted in relatively poor end-user and developer productivity. Conversely, Client-Server computing offered tremendous gains in end-user productivity, but at the cost of both accessibility and scaleabilty.

Internet Computing offers the strengths of both computing paradigms, it allows the rich user experience for which graphical applications are known, increases accessibility to the application, yet at the same time reduces the cost of application administration and maintenance. Server-based deployment moves the application complexity off the desktop onto scaleable, professionally managed application servers. In this manner applications are accessible from any client platform capable of running a standard Internet browser. The Internet Platform combines the advantages of both the character mode and client/server worlds, allowing organizations to deploy enterprise-class applications with increased accessibility and improved ease-of-use for the end-user.

This paper provides information on the Oracle Internet Platform, discusses interface and platform considerations for migrated applications, and gives an overview of the migration character mode to the Internet. Specifically, it discusses the migration of character mode SQL*Forms 3.0 applications to an Internet platform via the Oracle Forms Server. It will focus on issues specific to SQL*Forms and will not cover the conversion of SQL*ReportWriter 1.x modules.

ORACLE FORMS SERVER

The Oracle Forms Server provides a solution which enable organizations to take advantage of the benefits of the Web - low cost of administration, and increased information accessibility- while maintaining the strengths of client/server computing. That is, a rich user interface and reliable transaction management. Using a multi-tiered architecture it delivers the benefits of both paradigms in a single application, providing a Java-based thin client solution for the Web deployment of transactional database applications, while supporting the standard document formats (HTML/PDF/HTML CSS) for database publishing.

The Oracle Forms Server environment uses two components for online transaction processing (OLTP); the *Forms UI Client* and the Oracle *Forms Server* itself.

The Forms UI Client is a small generic Java Applet which is downloaded to the client platform, and functions to display the user interface and to handle communication with the Oracle Forms Server. The use of a generic Applet allows multiple Oracle Forms modules to use the same Java Client, rather than having to download a unique set of Java classes for each module. Having downloaded the Java Applet, it may be cached on the client platform, thus removing the need for subsequent downloads each time an application is run.

The Forms Server consists of two components:

- *Listener*. The Forms Server Listener initiates the Forms Server runtime session and establishes a connection between the Forms Client and the Forms Server Runtime Engine;

- *Runtime Engine*. The Forms Server Runtime Engine is a modified version of the Forms Runtime Engine, with user interface functionality redirect to the Forms client. It handles all form functionality except UI interaction, including trigger and commit processing, record management, and general database interaction.

For data publishing the Oracle Report Server, moves the reporting engine to a third tier, allowing the operational reports to be accessed from any Web browser. The Oracle Reports Server is a multi-process engine which is able to run multiple reports simultaneously, as requested by the distributed end users.

OVERVIEW OF THE MIGRATION PROCESS

Migrating a SQL*Forms 3.0 application to Oracle Forms Server involves;

- Planning the new application
- Analyzing the existing functionality
- Deciding on the nature and extent of the migration.

Is the goal to simply emulate the behavior of the original SQL*Forms 3.0 application in the Web environment, or to significantly enhance the application such that it is able to take advantage of new graphical Web functionality now available within Oracle Forms?

Assuming the latter, the migration plan should consider the following points:

- Purpose or goals of the migration
- New Graphical User Interface (GUI)
- Distribution of Processing Logic
- Platform(s) on which the application is going to run
- Programming rules and naming standards

MIGRATION GOALS

Before embarking on the migration process it is important to first determine several other points related to the goals or purpose of the migration. The answer to these questions will in many ways dictate the method used to migrate the application as well as how far the enhancement effort may be taken.

- What is the estimated life expectancy of the migrated application?
- Do you need to enrich or modify the application functionality?
- Does the current application run against an Oracle6 or Oracle7 instance? To be deployed with the Oracle Forms Server an application must access an Oracle7 or later database. If the database kernel is to be upgraded this effort must be factored into the migration.

Note: It is strongly recommended the data server be upgraded to Oracle8/8i in order to take advantage of the increased performance/scaleabilty it offers both OLTP and Data warehouse environments.

- Does the application currently take advantage of database kernel functionality such as Stored procedure, Database Triggers and Integrity Constraints? Given the move to a multi-tier environment is it feasible to move much of the existing application logic into the database server, or should it run on the application server tier?

- Has the character mode application been generated in SQL*Forms 3.0 with either the partial or complete assistance of a CASE tool?

- What size is the application? That is, the number of Forms, SQL*Menu modules.

- Does the current application take advantage of referenced objects/Forms from other modules?

- Does the application currently contain any SQL*Forms V2 style triggers, and is it the intention to convert these to PL/SQL during the migration ? (recommended).

- Should you plan to do the migration in several steps; in other words, is a progressive migration the best solution?

How these initial questions are answered can have cost implications with the differing migration methods. The Web migration is going to impact both the look and feel of the application, the method of data validation and with a move to a multi-tiered environment it's overall architecture.

Oracle Forms allows for the direct migration of SQL*Forms 3.0 applications to the current release (6.0), through the use of the Forms compiler (***ifcmp60.exe***). However, if the application to be upgraded contains SQL*Forms 2.0 style triggers, it will first need to be upgraded to Oracle Forms 4.5 before the migration to Release 6.0. This step is required to convert this form of trigger to PL/SQL as Release 6.0 will not support the editing of SQL*Forms2.x Triggers. In addition, PL/SQL that uses version 1.x specific functionality may not be converted to Version 8, in which case it will be necessary to manually fix this code.

By setting the command line argument UPGRADE to 'YES' the compiler will:

- Read in the INP module definition file (ASCII)

- Convert Field references to Item references

- Convert Field triggers to Item triggers. (e.g. ON-VALIDATE-FIELD => WHEN-VALIDATE-ITEM)

- Write out an Oracle Forms module definition file (binary)

To estimate the work involved in migrating the application to the Web environment, the functionality of the existing application should be reviewed and decisions made on what is to be enhanced and what to left unchanged.

The following check list covers some of the enhancements which relate to Web deployment.

GRAPHICAL USER INTERFACE

Unlike the serial nature of navigation in a character-mode environment, a graphical application revolves around the ability of the user to navigate between items in a non-linear or contiguous manner. As such, the validation model of the application focuses less on the act of navigation (e.g. KEY-NXTFLD) and more on the current item.

The following list of GUI issues is not exhaustive, but indicates areas that should be addressed in the overall migration process.

- Key-XXX
- Small fields
- Fonts
- Pop-up pages
- List of Values
- Video attributes
- Message, Prompts and Hint Text
- Boilerplate
- Dynamic properties of fields
- Window titles and ROOT_WINDOW

These areas are discussed in detail in the following sections.

KEY-XXX

One of the major changes in the migration from character-mode to a graphical environment is the control of user navigation such that the required validation and navigation is still enforced. In general navigation in character mode is achieved via the "Next Field" key, with the validation code being processed as the user presses the appropriate key. Likewise, any automatic navigation is performed at the same time. Within a GUI environment this validation model is inappropriate as the user may chose to use mouse navigation rather than the keyboard.

In order for validation to be processed independently of the method of navigation, the required processing should be tied to the item rather than the act of navigation. As such, validation code currently residing in KEY-XXX triggers should be placed in the following based on need.

- Data validation trigger : Event Fires as user attempts to leave an item after manually entering data.

  *Note: The ON-VALIDATE-FIELD trigger in SQL*Forms 3.0 has been superseded by the WHEN-VALIDATE-ITEM in Oracle Forms. During the migration process the forms compiler will automatically update any FIELD triggers to the new nomenclature.*

- Data change triggers (POST-QUERY, POST-CHANGE) : Event will fire each time the data within the field changes, regardless of method (e.g. Via execution of a query)

  Note: POST-QUERY is the recommended trigger, POST_CHANGE is included for backward compatibility.

- Obtain Focus Triggers. (WHEN-NEW-XXXX-INSTANCE) : Event fires as user navigates into the new field rather than on leaving previous

By the following methods it is possible to replicate the current Key based validation behavior using mouse navigation logic within an Oracle Forms application;

- Perform the actions when entering the new item (WHEN-NEW-XXX-INSTANCE triggers), as opposed to doing them when leaving the current item.

- Save the name of the item on which the focus was placed and the name of the item on which the cursor is now, and execute all the navigation triggers between the two using DO_KEY.

  Note: This use of DO_KEY triggers can result in a significant performance degradation if the amount of processing between the fields is great.

In cases where high speed data entry is required, it is often easier for end-users to use the keyboard rather than the mouse. Consequently, it may be more efficient to disable the navigation with the mouse in some specific modules rather than attempting to implement a mouse driven interface. The level of mouse support may be defined, in part, prior to the migration by the setting of the Form

level mouse navigation property with the SQL*Forms 3.0 Build environment. (This property was made available to accommodate platforms, such as Motif, that supported running a character mode form within a GUI window.)

SMALL FIELDS

When migrating from a character mode environment to the graphical interface supported by the web, it is important to consider the effect that both the chosen font and graphical rendering of objects (such as 3D-Bevel on text fields) will have on the display of the data. It is common for small fields (3 Characters or less) to visually truncate the data after the default upgrade (if the field is small enough it may be unable to display any meaningful data) There are several solutions to this problem:

- Use the option *widen_field=yes* when upgrading the SQL*Forms 3.0 module with ***ifcmp60.exe***. It may be necessary however, to manually check the layout after the conversion process as the increasing width can result in overlapping fields on the screen.

- Choose a smaller font. Based on the resolution of the display this can result in screens that are difficult to read.

- Suppress the bevel. This may not be satisfactory for the end user as it removes much of the 'modern GUI 'feel.

- Before the migration, increase the size of each SQL*Forms 3.0 field individually. Alternatively, resize the fields after the migration using Form Builder.

During the default migration the coordinate system will be preserved as the original character cell system. If the goal of the migration is to graphically enhance the application the coordinate system should be reset to a non-character system, thus allowing for greater use of the available screen real estate. The unit of POINT is recommended as it independent of both screen resolution and regional measurement systems.

> *Note: The change in Coordinate system can result in fields 'snapping 'down to the next viable character cell, resulting in a field of little vertical height.*

FONTS

The Java 1.x environment has native support for a limited number of Fonts. In order to determine which font to use, the Forms UI client will attempt to map the font chosen in the builder against one of the supported Java fonts.

The following table lists the Java fonts, and their equivalents on the major deployment platforms.

Java Font	MS-Windows Font	X Windows Font	Macintosh Font
Courier	Courier New	Adobe-courier	Courier
Dialog	Ms Sans Serif	b&h-lucida	Geneva
DialogInput	Ms Sans Serif	b&h-lucidatypewriter	Geneva
Helvetica	Arial	adobe_helvetica	Helvetica
Symbol	WingDings	itc-zapfdingbats	Symbol
TimesRoman	Times New Roman	adobe-times	Times Roman

Figure 1: Font mapping between Java and the major deployment platforms

To convert the font used in the migrated form into its Java equivalents, Java uses an alias list, contained within the Oracle Forms Server Java registry file *Registry.dat* For Oracle Forms Server release 6.0/6i this file is located in the *%ORACLE_HOME%/forms60/java/Oracle/forms* directory.

If the font you have chosen for the migrated application does not map to one defined within the registry file, the Forms Java client will automatically assign one of the supported Java fonts. This can result in an unpredictable interface. To control display fonts it may be necessary to update the font alias list in the registry.

> *Note: The registry file is used to store much of the information required by the Java client to function correctly. Incorrect modifications to this file may result in undesired behavior, as such, it is strongly recommended that the default font used be one which has a defined Java equivalent.*

POP-UP PAGES

Consider whether the existing character mode application uses pop-up pages. The initial upgrade of the SQL*Forms 3.0 module using the forms compiler (*upgrade=yes*) will result in each page (whether pop-up or not) being converted to a content canvas contained within it's own associated window. In SQL*Forms 3.0 pop-up pages where often used to hide/show sets of fields within a given window/page. In order to have same behavior in the migrated application it is necessary to convert the newly created CONTENT canvases to STACKED canvases within the same window. Use of the SHOW_VIEW and HIDE_VIEW built-ins will allow the Form items to appear or disappear as in the original application.

LISTS OF VALUES

If a 'list of values' has been defined on a field, the migration process will convert this to a standard pop-up LOV window. It should be noted that when run with the Oracle Forms Server the default vertical size of the created LOV window is such that only one record will be displayed at a time. Likewise, the conversion will set the definition of numeric columns within the LOV as Number(5). It is therefore likely that any List of Values will require manual sizing after the default migration.

MESSAGES, PROMPTS AND HINT TEXT

One of the biggest advantages of web deployment is that the application may be accessed from any browser, or client platform, capable of running the appropriate Java virtual machine. This ubiquity however can result in further migration effort if the application has been traditional aimed at a specific terminal type (e.g. VT220, WYSE 50) . Direct references to the physical keyboard layout (e.g. "Press F5 to ... ") are often meaningless in an environment where the end user may be accessing the application through many different devices. As such all references should be to the required function rather than to the physical key.

> *Note: If there is a requirement to map functions to specific keyboard layouts (to maintain backward compatibility) Oracle Forms Server supplies a key map file. Note this is a simple text file and unlike the technique used to map keys in SQL*Forms 3.0 does not require Oracle Terminal (**oraterm**). Refer to the section on Keyboard mapping for the location of this file.*

In the character-mode application standard Forms Error or Informational messages (or explicit calls to the Forms MESSAGE built-in) were directed to the console line at the bottom of the screen (Line 25). The user asked to acknowledge the message by hitting the return key before any processing

could continue. In contrast to this, a multi-window GUI environment typically displays important messages via a modal dialog box rather than send them to the windows status line.

> *Note: Within the graphical Oracle Forms environment multiple calls to the message line (either implicit or explicit) will result in the first (n-1) messages being displayed on both the console and via an Alert box. The last message is displayed only on the message line.*

It is recommended that as part of the migration process ON-ERROR and ON-MESSAGE triggers be added to the module to allow for the redirection of these messages to a more user-friendly dialog. Like-wise explicit calls to the MESSAGE built-in should be with references to a standard Alert procedure.

Forms 3.0 supports the ability to automatically display a 'hint' on the status line as the user moves into a given field. As with many GUI environments, Oracle Forms supports the concept of 'Tool tips' or Bubble help (displayed as the user moves the mouse over the item). As part of the migration process it should be determined if it is viable to display the current Field level hint text as a tool tip or 'bubble help'.

BOILERPLATE

Text

The default migration from SQL*Forms 3.0 will convert individual field labels to boiler plate text objects on the form canvas. They are however, static objects that have no direct relationship to any data item.

Oracle Forms allows for the definition of the prompt as a property of the Item. Resulting in:

- Greater control of screen Geometry (the prompt will move with the item)
- Programmatic control over the text within an Item prompt
- Optimize network traffic between the client tier and the application server. This is achieved through optimized property reuse and client side string caching.

> *Note. Please refer to the "Tuning Developer Server for deployment of Internet Applications" white paper for a detailed discussion on the network optimizations used by the Oracle Forms Server. This is available from the Oracle Technology Network (http://technet.oracle.com)*

In order to facilitate the conversion of a field prompts from static string to item property, the Oracle Forms Builder allows for the association of a boilerplate text string with a given Oracle Forms Item

(text item, radio button etc.). Once associated with an item the prompt property is set and the original boilerplate text item removed.

It is highly recommended that after the default migration all relevant boiler plate text objects be associated with their respective items. This can be achieved quickly and easily via use of the *'Associate Prompt'* button in the layout painter.

Graphics

It was common in SQL*Forms 3.0 applications to use simple lines/boxes to delineate areas of the screen into different logical zones. Often, these boxes were not created as a single element but rather made up of separate lines. After the migration boxes of this nature will appear as non-contiguous lines. Those created with the ***DRAW BOX LINE*** option will be correctly migrated.

If the 'zone' is based on a single data block the conversion of the box surrounding it to a native frame object (after basic migration) allows for grater control over layout in the new bitmap environment.

Further-more it may be possible to simplify the overall screen layout by moving these 'zones' to separate independent windows, thus removing the need for boilerplate to isolate groups of fields.

DYNAMIC PROPERTIES OF FIELDS

If you have dynamically modified access to a SQL*Forms 3.0 application field using the SET_FIELD built-in, remember that with a graphically based Oracle Forms application, users can access items using either the keyboard or the mouse. In the migrated application, consider using the SET_ITEM_PROPERTY built-in with the appropriate property, to provide the intended behavior.

For example: *navigable, mouse_navigate, enabled*

VIDEO ATTRIBUTES

When migrating an application to the Oracle Forms Server, you must replace the video attributes used in character-mode with named visual attributes.

If necessary, these should reference *charmode logical attributes* that are defined in Oracle Terminal (***Oraterm***). The *charmode logical attribute* ensures that if the application is required to run in both character and Web mode, the screen's appearance will be 100% identical to its appearance in SQL*Forms 3.0

WINDOW TITLES AND ROOT_WINDOW

By default, the migration using the Form Compiler creates a Multiple Document Interface (MDI) with the primary window given the name 'ROOT_WINDOW'. This name may cause un-predictable behavior when running against the Oracle Forms Server. Rename the ROOT_WINDOW to one based on the current functionality or use a nomenclature where each window would be uniquely named. For example, WINDOW_<*i*>, where *i* is the sequence number of the window.

Windows created during the migration inherit the titles of the original SQL*Forms 3.0 pages. If the original page does not have a title, then the new window derives its title from the *name* of the page. For aesthetic reasons it is worthwhile explicitly setting the Window titles.

NON-GUI CONVERSION ISSUES

This section discusses the following forms conversion issues:

- Logic Partitioning
- Data types
- Date Ranges
- Format Masks
- POST-CHANGE and ON-VALIDATE-FIELD triggers
- Reserved Words
- Trigger Style

LOGIC PARTITIONING

The migration to the Oracle Forms Server introduces distributed processing, a significant architectural change over the generally HOST based SQL*Forms3.0 applications. In general SQL*Forms 3.0 applications were designed to run either on the same platform as the database or against a remote server in a TWO_TASK (client/server) mode. In most cases the queries and application PL/SQL were executed within the application itself. In the Oracle Forms Server distributed architecture the server itself runs against the Oracle database in a two-tier client server mode. As such, it is important to optimize the network traffic between the application server and

the database. To achieve this, the migration should look to moving data intensive client side PL/SQL to the database (using stored packages, Functions and Database triggers).

> *Note: PL/SQL 1.1 was used in SQL*Forms 3.0. It provided basic support for the use of Stored procedures (though not stored functions), consequently, it may be possible to make some changes to the existing SQL*Forms 3.0 application before the migration to the Forms Server. The PL/SQL code will be upgraded to PL/SQL8 as part of the compiling process.*

Currently, it is not possible to directly place application trigger code on the client UI tier except through the use of Pluggable Java Component extensions. In a future release of Forms Server it will be possible to locate Trigger code wherever it makes most sense; in the Database, on the application server tier or on the client platform.

DATA TYPES

Determine whether the existing application uses datatypes specific to SQL*Forms 3.0, such as JDATE, EDATE, RINT, RMONEY, and so on. These data types are maintained for backward compatibility and if possible should be converted to one of the data types shown in figure 2, with the relevant format mask applied.

Data Type	Description	Example
CHAR	Supports VARCHAR2 up to 2,000 characters	user_id CHAR(8)
DATE	Contains a valid date	hire_date DATE (dd-mon-yyyy)
DATETIME	Contains a valid date and time	DATETIME (dd-mon-yyyy hh24:mi:ss)
LONG	Contains any combination of up to 65,534 characters	resume LONG
NUMBER	Contains fixed or floating point numbers	annual_sal NUMBER(7,2)

Figure 2: Recommended data-types for deployment with Oracle Forms Server

The data types shown in the proceeding figure relate to items within a form. The range of data types for use in PL/SQL procedures has increased to supporting much of the functionality of PL/SQL 8.

Note1: Limit the use of global variables. They require data to be treated as Character, consume a large amount of memory, and can cause unpredictable behavior if not properly erased after use. (Global variables remain for the entire session unless explicitly removed).

DATE RANGES

In SQL*Forms 3.0 date fields were held within the form as character values, necessitating the use of conversion functions if date arithmetic was to be performed. The current release of Oracle Forms allows for date fields to be accessed directly without the need for conversion. As such, individually assign the appropriate date format mask to each date range.

For date formats, make sure you also consider the implications of the new century within the migrated application. As such it is recommended that the default format masks include the 'RR' or 'RRRR' date formats.

Note: For a detailed discussion on date handling and Y2K compliance please refer to the white paper "Date handling in Developer/2000" available form the Oracle Web site (http://www.oracle.com).

FORMAT MASK

Oracle Forms has greater restrictions on format masks than SQL*Forms 3.0. Formats with blanks, such as "99 999 99", were accepted in SQL*Forms 3.0. However, these generate errors during standard conversions using the Forms Compiler. It is now necessary to place double quotes around blanks. For example, the format "99 999 999" should be expressed as: "99,999.99".

POST-CHANGE AND ON-VALIDATE-FIELD TRIGGERS

Prior to the migration step, investigate whether there are occurrences of both POST-CHANGE and ON-VALIDATE-FIELD triggers having been coded on the same field. If this is the case, split the associated actions between the POST-QUERY and ON-VALIDATE-FIELD triggers, and delete the POST-CHANGE trigger. In this case POST-QUERY to return any required display data (such as descriptor fields) , and ON-VALIDATE-FIELD to specifically validate the users input.

The migration process will convert the ON-VALIDATE-FIELD trigger, which is not supported in Oracle Forms, to the corresponding WHEN-VALIDATE-ITEM trigger.

TRIGGER STYLE

If the character mode application still uses SQL*Forms version 2-Style triggers, consider rewriting them in PL/SQL. This will yield better performance. The version 2-Style trigger will still execute within the Oracle Forms Server, however the code may no longer be edited (or new V2 triggers created). Given the requirement however, to migrate the trigger from the version 2-style macro language to PL/SQL, it is still possible to view the trigger code within the build environment,

> *Note: Release 6.x of Oracle Forms will be the terminal release to support SQL*Forms V2 trigger syntax. As such, it is recommended that such triggers be re-written to take advantage of PL/SQL as soon as possible (the benefits of this change depends on the life expectancy of the converted application).*

RESERVED WORDS

Make sure that the existing SQL*Forms 3.0 code does not contain any Oracle Forms reserved words. Please refer to the Oracle Forms documentation for a list of reserved words.

PLATFORM CONSIDERATIONS

Direct migration of the current Forms 3.0 applications to the Web environment, avoids many of the client side idiosyncrasies which plagued many client/server deployments. The ultimate client is a standard Web browser, virtually independent of the client operating system and therefore protected from the myriad OS compatibility problems that can arise . The server tier however should be determined according to your specific situation and should be based on a platform on which Oracle Forms Server is available. These include, the major UNIX platforms, Windows NT, OpenVMS and several others.

If the new deployment platform is not the same as the platform for which the application was developed, the migration will have to allow for the differences between the platforms. Differences can occur in such areas as;

- Host commands
- User exits
- Environment variables
- Keyboard mappings

HOST COMMANDS

With the migration from the character mode runtime to the distributed web environment, it is important to take into account that the application itself and any subsequent calls to the operating system will run on the middle tier rather than the client tier where the user interface is displayed. As such, the application can not rely on the need for user interaction with the hosted program as the user interface is controlled by the Oracle Forms Server.

For example, if the Oracle Forms Server is running on a platform that differs from the original runtime environment (e.g. Windows NT) and the application requires execution of an program on the original server platform, it is possible to implement a remote host call. This may be achieved through use of rexec or rsh (UNIX to UNIX) or via the use of a resident Daemon and the DBMS_PIPE package.

The latter uses a simple daemon on the host server to listen for requests sent via a DBMS pipe. Having received a message from the application tier it would perform the Host command as in the current application.

> *Note: The end user would not receive visual feedback in either implementation. The execution of the called program is occurring on a server tier and is therefore invisible to the client.*

USER-EXITS

One of the many benefits of the SQL*Forms environment was the ability to extend the basic product through the use of user supplied exits. These user-exits were written in a standard 3GL (such as C' or Pascal) and were statically linked to the Forms Runtime Engine. Thus were executed as part of the main SQL*Forms runtime process.

In the migration to the Oracle Forms Server, the same restrictions are placed on user-exits as are placed on the Host command. That is, the code is executed on the server platform and not the client platform where the UI is running. This means the user-exit may not perform output to the display (e.g., sending a message to the status line).

Because they are written in a 3GL, user-exits are not easily portable and imply additional administration and delivery tasks. Given the increased functionality found in the current release of PL/SQL it may now be possible to eliminate the need for the user exit itself. If this is not possible it will be necessary to port the underlying 3GL code to the new Oracle Forms Server platform.

Having ported the code, it is necessary to link the user exit to the Web runtime engine (***ifweb60***) which is part of the Oracle Formsr Server. For more details on linking user exits please refer to the Oracle Forms documentation.

> *Note: On MS-Windows NT user exits are linked to a stub DLL (xtb.dll) rather than directly to the runtime executable (*.exe).*

A more portable (and sustainable) solution for user exits is through the use of ORA_FFI or foreign function interface. This allows for the dynamic loading and registration of functions within a shared library (*.dll or *.so) via a PL/SQL interface. As the use of ORA_FFI allows for the dynamic linking of a library to the runtime environment it does not require a rebuild of the executable if the 'user-exit' should require further modification.

Statically linked user exits support the ability to pass data directly to/from Forms fields via the IAP_FORM_GET/PUT built-ins whilst ORA_FFI requires all data to be passed as arguments to the called function. The need to directly get and set Forms variables from with the user exit will therefore dictate whether it will be possible to migrate the code to the more flexible dynamically linked model.

> *Note: If the user exit code requires access to the UI platform UI directly (e.g. accesses client side system time) or has some UI of it's own (e.g. pop-up window) it may be possible to implement it through the use of Java and the Pluggable Java components (PJC) capability of Oracle Forms Server R6.*

ENVIRONMENT VARIABLES

The use of operating system Environment variables by an Oracle Forms Server application is twofold:

- To allow the executable to find both application components and executables by searching a series of defined paths (FORMS60_PATH, ORACLE_PATH,...) .
- For retrieving application specific information from the operating system. IT is worth using the tools_util package procedure to return the environment variable in a manner that is not tied to a specific operating system.

Under Windows NT, Oracle Forms Server specific environment variables are defined in the registry under the default ORACLE node. As such, you may only have one active set of Oracle Forms Server environment variables on a given machine, at a given time. On a UNIX platform however, the environment variables are defined in the shell used to launch the Oracle Forms Server. Hence it is possible to have as many different configurations as is required. The syntax of the environment

variables on this platform will likely remain the same as they were in the original SQL*Forms 3.0 application.

KEYBOARD MAPPINGS

If the SQL*Forms 3.0 application took advantage of a custom Keyboard to Function mapping it will be necessary to rewrite these for web deployment. The resource file used by the Oracle Forms Server to map Key to Triggering Functions is ***fmrweb.res*** (found in %ORACLE_HOME%/Forms60). Unlike the binary resource files used with SQL*Forms 3.0 this is a text file which can be edited via a regular text editor. That is, Oracle Terminal is not required.

When defining a custom keyboard mapping it is important to choose keys which are common across the platforms with which users are accessing the Web. That is, avoid hard coded references to keys such as 'Option' and 'Meta'.

MENUS AND REPORTS CONVERSION ISSUES

The migration of an application must also include any related menus and reports that run from the SQL*Forms module.

MENUS

In character mode applications the use of full screen menus to navigate from one module to another was a common user interface. Within a GUI environment it is standard for menus to be implemented as hierarchical pull-down menus. In keeping with this standard, Oracle Forms Server does not support the use of Full screen menus.

With SQL*Forms 3.0 menu was performed with a separate tool, SQL*Menu 5.0. In the current build environment both Forms and Menus are created within the same executable. This is also true for the upgrade process. By use of ***ifcmp60*** the menu will be extracted from the database and an *.MMB file created.

If the current menu has significant levels of security, it is recommended that the migration direct the security structure to use the database role support that is now available.

REPORTS

In concert with the Oracle Reports Server, Oracle Reports delivers a complete database publishing environment by providing the ability to generate and distribute reports dynamically to the web. Through the ability to dynamically generate the standard document formats of the web, that is, HTML, HTML with cascading Style Sheets (HTMLCSS) or Adobe's Portable Document Format (PDF) the migrated applications may maintain the on screen display of related reports.

The reporting tool that was used in the original application (SQL*Report [rpt/rpf] or SQL*ReportWriter) will determine how much of the original reporting structure would be able to be migrated.

RPT

No migration tools are currently available to assist in the process of converting from SQL*Report to Oracle Reports 6.0. A number of third party vendors have migration methodologies that may help. Given the ease of use and high productivity found in the current versions of Oracle Reports it may be more efficient to re-create the reports from scratch rather than attempt to convert the RPT/RPF codebase.

SQL*ReportWriter

Reports developed using SQL*ReportWriter may be migrated without significant modification by use of the Oracle Reports ***MOVEREP*** utility. This utility was shipped with Oracle Reports up to and including Release 2.5. Unlike SQL*Forms modules, SQL*ReportWriter modules must be migrated to Release 2.5 before it may be used with the Oracle Forms Server. Once converted to the RDF format, the file may be opened directly within the build environment of Oracle Reports.

The conversion of character mode reports to a bitmap format is outside the scope of this paper.

MIGRATION METHODS

Having analyzed the SQL*Forms 3.0 application and determined what functionality can be enhanced, the actual migration of the application generally takes place two distinct stages :

- Conversion of the application module's source file from the text based INP format used in SQL*Forms 3.0 to the binary FMB format used by the Oracle Forms Builder.
- Enhance the functionality of the application to take into account a graphical, Web based deployment environment.

STAGE 1 : CONVERTING THE INP FILE

The conversion of the INP file is performed by the Forms Compiler (ifcmp60 with the upgrade option). The following options are recommended

Parameter	Description
MODULE=<name>	Name of INP file to convert
OUTPUT_FILE=<name>	The name of the FMB file to be created
UPGRADE=YES	Tells the compiler that the file is to be converted to the latest format
VERSION=30	Notifies the compiler which version of Forms to upgrade.
BUILD=NO	Tells the compiler to create an executable file (*.FMX). If OUTPUT_FILE is defined and BUILD=YES the output file created will be the compiled Form and not an FMB source file.
BATCH=YES	If a number of forms are to be migrated setting this argument to YES allows for background processing.
WIDEN_FIELDS=YES	Adds an extra character space to fields to account for GUI effects such as bevel.

STAGE 2 : TUNE THE APPLICATION, AND ADD ORACLE FORMS SERVER FUNCTIONALITY

There are a number of ways of completing the second migration stage. Some of these involve;

Oracle Tools and services:

- Oracle Forms (manual enhancement)
- Oracle Designer

Third party tools and services that are promoted by Oracle :

- FormsGenie (Sierra Atlantic, Inc.)
- GUI*Converter (Kumaran Systems, Inc.)

In house Solutions

- Customized Tools

ORACLE TOOLS AND SERVICES

Oracle Forms (manual enhancement)

The most direct method of enhancing the application is through the Forms Builder environment. Before proceeding with this mode of migration it is recommended that a set of rules and UI standards for the migrated application be first defined. Through the use of standard libraries of Object Groups, Property Classes and Visual Attributes, an application may quickly be converted to take on the desired GUI look and feel. For example, the definition of a standard Text item Smart Class (including Font, Fill Color, Bevel etc.) would allow other items to be sub-classed from this, giving a standard look to all text items.

As the Form is enhanced to include new GUI style interface elements the following guidelines should be followed. These simple rules will help to reduce the network traffic between the Forms Server and the Forms Client:

- *Mouse triggers* : Including mouse triggers management, like WHEN-MOUSE-CLICK, WHEN-MOUSE-DOUBLECLICK, WHEN-MOUSE-DOWN and WHEN-MOUSE-UP will impact speed and performance. Each time one of these triggers fires the Forms Client must communicate with the Forms Server (necessitating a network round trip). In particular, the WHEN-MOUSE-MOVE trigger is not supported by the Oracle Forms Server. This is due to the high number of network round-trips required it would generate.

- *Timers* : Each time a defined timer expires it will generate a round trip from the Forms client to the server. Limit the use of timers and select realistic intervals on which they are to expire. Fore example, if a form included a timer that fired every 100th of a second, the end user would face the performance ramifications of 60,000 network round-trips every minute.

- As a rule of thumb it is recommended that the expiry interval be limited to those over 1 minute. For expiry periods less than this then it is recommended that the use of a Pluggable Java component be investigated.

- *Image item and background images*: Each time an image is displayed to application users, the image must be streamed from the Forms Server to the Forms UI Client. As such it is advisable to minimize the total number of images contained within the form. Use the Oracle Forms Server's splash screen functionality to display a company or application logo as the application starts up. (this has the added benefit of masking the Java class download time)

- As the form's GUI progresses, the need for an iconic navigation bar is likely to increase. Create the icons in GIF format and size them as needed. For example, for an MDI toolbar, icons should not be larger than 32x32 pixels; GIFs for application level buttons can be much larger. The GIF files must either reside in the same directory as the HTML page or be specified within the registry.dat file. Please see the Oracle Forms Documentation for information about the registry.dat file.

Based on previous migration efforts both by external organizations and within Oracle, the amount of time involved in performing manual migration to a full GUI Web environment is between 0.6 and 1.5 person days per form, assuming an average complexity of UI.

As with all manual code methodologies there must also be significant time set aside to test the new configuration after the conversion. This is in contrast to more automated solutions.

Oracle Designer

Oracle Designer - with its latest release, Release 6.0 - is a highly recommended migration solution in that it offers both automatic generation capabilities of Oracle Forms applications and automatic reverse-engineer support for custom modifications made after the initial migration step. Oracle Designer supports model-based development of database applications, which is the most productive approach for building enterprise-class information systems. Model-based development defines the new economics of application development.

Application and server definitions, designed as business and functional models, are stored in the Oracle Repository independent from their final implementation. By defining the application once, it may be generated as to different physical incarnations, such as in Oracle Forms, dynamic HTML, Visual Basic, or C++.

Note: This discussion applies only to applications which have not been built, either partially or wholly, with a CASE tool. For applications which have been wholly built with a CASE tool, it is relatively easy to migrate the existing repository to the Oracle Repository and then re-generate the modules.

With Oracle Designer it is possible to quickly;

- Retrofit data from physical to conceptual levels
- Enrich and modify the conceptual data model
- Generate a new logical and physical data model
- Reverse engineer the pre-migrated modules (after stage one of the migration)
- Enrich the module design with new widgets, updated logic partitioning, mouse navigation support, and all the required actions to ensure the same behavior as provided in the original SQL*Forms 3.0 application.
- Generate the new application as an Oracle Forms application and deploy it on the Web with the Oracle Forms Server.

In support of Internet migration, Oracle Designer has the ability to generate a module into two different interface environments;

- Java based (via The Oracle Forms Server)
- Oracle Forms modules are produced with the Forms and Reports Generators. These modules can then de deployed on the Web with the Oracle Forms & Reports Servers.
- HTML (via the PL/SQL agent)
- The PL/SQL Web Generator can generate PL/SQL packages deployed as PL/SQL cartridges. These packages generate dynamic HTML pages on the fly which incorporate both the database access and the layout defined using the templates and standards set up inside Oracle Designer;

When looking at a migration to the web it is important to review the requirements for the application and to choose the appropriate implementation model for each of SQL*Forms 3.0 module

In addition, using Oracle Designer for a migration project allows you also to benefit from the other advantages of a modeling approach in terms of documentation, maintenance, and application reusability.

THIRD PARTY SOLUTIONS

FormsGenie

FormsGenie is a tool developed by Sierra Atlantic, Inc. to migrate SQL*Forms 3.0 applications to the Oracle Forms Server. It also has the ability to migrate applications built in other 4GL environments, such as Powerbuilder and Ingres ABF, such that they too may run against the Oracle Forms Server. FormsGenie has many built in features which insure that the migrated application will perform well in a Web environment, however it may be used as well for client-server migration projects.

In its current version (2.0), FormsGenie:

- Performs the initial upgrade from SQL*Forms 3.0 to Oracle Forms using the Oracle Forms migration option.
- Integrates the Year2000 management issue.
- Offers specific options to migrate to the Oracle Forms Server.
- Adds widgets to the application. These widgets include visual attributes, messages in Alert boxes, vertical scrollbars, check boxes, pop-lists, horizontal and vertical toolbars, and so on.
- Treats all pop-up pages specifically. You can decide which ones you want to remain as content canvasses (i.e. not pop-up) and which ones you want to have in stacked views (i.e. pop-up).
- Searches for Oracle Forms reserved words which may have been used in the SQL*Forms 3.0 application, and makes substitutions for those words.
- Offers an "out-of-the box" product that can be used without assistance.
- Changes all non-enterable fields to display items. Note that display items do not accept format masks. You may have to remove the format masks, where applicable, from the display items.
- Provides a solution based on the published Oracle Forms APIs (Application Program Interface).
- Can migrate forms with names which are longer than eight characters.
- Integrates a full mouse navigation capability inside the migrated application.

See the Appendix for information on contacting Sierra Atlantic, Inc.

GUI*Converter

GUI*Converter from Kumaran Systems, Inc provides a migration path for SQL*Forms 3.0 applications to the Web both in terms of GUI objects and code conversion. It is available in 2 versions:

- GUI*Converter Pro, that fully supports mouse navigation,
- GUI*Converter Lite, for applications that do not need mouse navigation support.

In its current release (3.8), GUI*Converter Pro:

- Performs the initial upgrade from SQL*Forms 3.0 to the Oracle Forms Server using the Oracle Forms migration option.
- Provides full mouse navigation support.

> *Note: The mouse navigation support implemented by GUI*Converter Pro requires the use of mouse management triggers,. These will generate extra network round-trips between the Forms Client and the Forms Server. GUI*Converter Pro offers the delay firing checks involved by mouse navigation.*

- Adds widgets to the application. These widgets include visual attributes, messages in Alert boxes, vertical scrollbars, check boxes, pop-lists, horizontal and vertical toolbars, and so on**.**
- Takes into account specific format masks that are no longer allowed in Oracle Forms. Changes all non-enterable and non-queriable fields to display items.
- Modifies the title of the MDI window.
- Provides a quick and flexible solution.
- Performs the migration in a manner that is transparent to the end user.
- Offers the ability to convert old pop-up pages in stacked views.

See Appendix for information on contacting Kumaran Systems, Inc.

CUSTOM BUILT MIGRATION TOOL

If the number of Forms to be migrated is high, and there is sufficient Information Systems resources available, it may be cost effective to create a specific custom migration tool. Oracle Forms publishes an API to the internal structure of the FMB file. With this API it is possible to read, update or create an FMB file through a set of pre-defined functions in the C language. This approach is fully supported by Oracle Worldwide Support and allows the creation of an in-house migration solution. It is recommended that before the decision to build a custom tool is made the

availability of a suitable 3rd party solution be first determined, as the development effort will dramatically increase both the time and workload required to complete the overall migration.

CONCLUSION

The migration of a character mode application to the web allows for the cost benefits of server side computing, with the user productivity of the GUI environment. The Oracle Forms Server allows for the migration of an existing SQL*Forms 3.0 application to the Web by the implementation of a thin Java client for UI and a middle tier for application execution. In this new configuration, the application remains easy to maintain and to deploy, but end-users can now benefit from a much more user-friendly application.

There are several options available in order to migrate legacy SQL*Forms 3.0 applications to the Web, from a purely manual conversion to the use of highly automated tools from third parties. The method of choice will be dependent on the number of modules, the degree of GUI enhancement required and the life expectancy of the migrated application.

APPENDIX

ORACLE FORMS COMPILER

Usage: ifcmp60 Module=<INP filename> Userid=<Userid/Password> [Parameters]

Relevant Parameters	Description (defaults indicated)
MODULE_TYPE=FORM *	Module type (FORM, MENU).
MODULE_ACCESS=FILE *	
STATISTICS=NO	Show Statistics
LOGON=YES *	Logon to database
BATCH=YES * (recommended)	Don t display messages on the screen
OUTPUT_FILE=<FMB filename>	Write output to file
GRANT_ADMIN	Grant a user administrate privileges for Forms Menus
GRANT_DESIGN	Grant a user design privileges for Forms Menus
GRANT_EXECUTE	Grant a user execute privileges for Forms Menus
UPGRADE=YES *	Upgrade module to current version
UPGRADE_ROLES=NO	Upgrade SQL*Menu 5.0 role information.
VERSION=30 *	Version from which to upgrade (23, 30, 40, 45 or menu 50)
CRT_FILE=<crt_file>	CRT file for version 2.x form upgrade
BUILD=NO *	Build a FMX or MMX file when upgrading.
ADD_TRIGGERS=NO	Add Key-UP/Down triggers during the upgrade.
NOFAIL=NO	Add the NOFAIL clause to version 2.x style triggers.
DEBUG=NO	Build/Run with debug information
STRIP_SOURCE=NO	Strip PL/SQL source from library file.
WINDOW_STATE=MINIMIZE *	MS-Windows MDI state (Normal | Maximize |

	Minimize)
HELP=NO	Display command line arguments
OPTIONS_SCREEN=NO	Display options window (bitmap only)
WIDEN_FIELDS=YES*	Add one character to the display width of Text items

(*) Recommended parameters with values for performing initial upgrade from SQL*Forms 3.0

MIGRATION SOLUTION CONTACTS

The following lists the company name and contact information for the conversion tools and solutions referred to in this paper. In addition to the partners mentioned in this document there are a number of vendors who have solutions which can help in the migration process. These include;

Asymptote: Provides a migration tool from SQL*Forms 3.0 up to Oracle Forms,

Sinorg: Provides Free Mouse, a migration solution from SQL*Forms 3.0 to Oracle Forms client-server and to the Oracle Forms Server.

Company	Migration Solution	Web Site
Sierra Atlantic (USA)	FormsGenie	http://www.sierraAtl.com
Kumaran Systems, Inc. (Canada)	GUI*Converter	http://www.kumaran.com
Asymptote (France)	ASY2345	Email: asymp@clubinternet.fr
Sinorg (France)	Fee Mouse	http://www.sinorg.fr

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

Worldwide Inquiries:
+1.650.506.7000
Fax +1.650.506.7200
http://www.oracle.com

Migrating SQL*Forms 3.0 Applications to Internet computing

Technical White Paper: EIT-DE-WP-0010