Web-Deploying Oracle Character-Based Systems with Oracle Developer Server Rel 6

Technical White Paper
March 2000

Forms 3 to Developer 6 Forms Server (Intranet) Migration Experiences by Motherwell Information Systems

ORACLE

Motherwell Information Systems

Introduction

The ever-growing maturity and acceptance of the Internet and related technologies has provided organisations with a platform from which to deploy applications that are simpler to implement and administrate, and also more accessible to a wider user base. Acceptance of the Web has now reached 'fever pitch', with many organisations having already utilised its associated infrastructure and technology. Much hyped claims such as 'Build once, Deploy Everywhere!' and 'Zero Client Deployment' are commonplace in today's database application environment, where the majority of new and existing applications are considered for Web deployment. Although industrial-strength database application deployment over the Web is relatively immature, much of the industry has already embraced its major strengths. It combines the benefits of Client/Server computing – in terms of GUI richness and flexibility – and the benefits of Server-Centric computing, where ease of maintenance and low client workstation costs can be realised.

Do available and emerging Web technologies work acceptably, with real business-critical applications that require industrial-strength operation? Specifically, are Oracle's Developer and Developer Server products ready for Web deployment in this new Millennium? The purpose of this paper is to present a 'Real World' perspective from the experiences of the author, who has led the Web deployment (Intranet) – over a 3-Tier architecture – of an existing VMS character-based application suite, utilising Oracle's latest offerings – Developer Release 6[1] (Server and Client toolset) and Oracle Application Server Release 4. The paper describes the approaches taken and highlights the experiences of the team in terms of functional areas (where the chosen technology has provided positive business benefits), the changes required to the applications (due to a shift in systems architecture) and the difficulties that were encountered (including workarounds that were adopted). Due to common availability and acceptance of techniques and such issues within the industry, the paper does not address Forms V3 to GUI migration issues in depth, nor does it present a solution for Millennium/Year 2000 compliance of bespoke Oracle applications software.

Since the initiation of the project in April 1999, the author has acted as Technical Architect and Team Leader for this major migration exercise on behalf of Vickers Defence Systems (VDS), a major heavy-engineering firm based in the North East of England. The core business requirement was to migrate a number of business-critical applications from a VMS Server-Centric environment to a Windows NT platform, which would support Millennium-compliance whilst offering a more accessible, feature-rich and intuitive Intranet interface.

At the time of writing, all five applications have recently 'gone live'. Due to the submission deadline for this paper, and the evolving nature of the project, the results reported herein are based on the core software versions as detailed in the 'The Hardware and Software Platforms' section. New Releases (including Software Patches) will be implemented as and when necessary.

Background and 'The Core Business Requirements'

Motherwell Information Systems (MIS), provide software development and consultancy services to a wide range of customers focussing on manufacturing industries and utility companies. As specialists in the Oracle arena and an Oracle Certified Solution Partner, MIS were contracted by VDS to migrate five core manufacturing and engineering applications, collectively known as EDICS (Engineering Department Information Control Systems). Running on two DEC Alpha VMS Servers accessing a geographically distributed database, EDICS technical statistics were as follows:

200 Forms V3[2] modules	400 SQL*Plus Report modules
100 DCL Command modules	30 User Exits (Pro*Fortran/Fortran)
700+ database tables	

The solution had to conform to the following core requirements:

- Millennium compliance.
- Tight integration with existing company Intranet and HCI standards.
- Ease of deployment.
- Low specification workstations.
- Fail-safe and intuitive operation.
- Retention of central control of applications.

[1] Developer Server, as referred to in this paper, is synonymous with Forms Server 6/6i.

[2] Some Forms modules were originally developed using Forms V2.3, meaning that V2-style triggers would need to be addressed during the migration.

Following an initial Design phase, including a 'Proof of Principle', a team of 6 developers was assembled. One of the developers was a VDS IT resource, with little Oracle experience. (This MIS/VDS development partnership would ease the burden of in-house maintenance once the migrated systems were in place, by providing valuable application expertise).

The Challenge

With a deadline of the Millennium, all core EDICS functionality had to be converted within 6 months. Initial planning estimates meant that each module (depending upon its complexity) would have to be converted in an average of 0.5 man day! Fortunately, there was a moratorium on any new development and change requests during the migration period, which aided configuration management.

Why Developer Server?

Oracle's Developer Server was immediately identified as a front-runner, due its scalable and flexible architecture. Fundamentally, its ability to Web-deploy existing Oracle Developer applications with minimal change to application code made it clear favourite over other Internet development technologies such as Java and Microsoft's Active Server Pages. Although the utilisation of such and similar technologies - used with success on other MIS projects - was evaluated, considerable confidence was given in Oracle's Developer Server product suite.

Developer and Developer Server Release 6 (production) was made available in May 1999. This timed nicely with the Design and Proof of Principle phase (see below), although lack of exposure and available technical knowledge resources meant that we were 'blazing the trail' with this hot release. The day it arrived was not soon enough to have it out of the box and ready for installation and evaluation!

Why Oracle Application Server?

Developer Server can co-exist and operate almost seamlessly with most major third-party Web Server products. (e.g. Microsoft Internet Information Server, Netscape Enterprise Server and Apache). In essence, Oracle Application Server (OAS) was chosen in preference, due to its native integration with Developer Server, flexible Application Server Load Balancing and also for single-vendor (i.e. Oracle) support and maintenance issues. OAS has been utilised solely as a Web Listener/daemon. Currently, none of its core Cartridge functionality has been used.

Cartridge or Static Implementation?

A Static non-cartridge Web implementation of Developer Server was chosen. Considerable previous experience of this method was the main influence, along with the required application operation, where a conventional menu system provides centralised and captive control. (See 'Intranet Integration' section for details).

Sockets or HTTP communication?

The Forms Server has been implemented in its default listening mode - TCP/IP sockets communication. (These are widely regarded as simple and efficient). Although HTTP 1.1 is supported in Release 6.0, it is a beta feature only and introduces additional communications overhead.

Furthermore, Sockets are acceptable for local Intranet communication, where access through a Firewall is not required.

The Hardware and Software Platforms

The hardware platform was modelled according to the now common 3-Tier Web architecture:

Database Server	-	Dual 450MHz Pentium II 3 x RAID 1 Arrays	512MB RAM 1 x RAID 5 Array
		MS Windows NT4.0, Service Pack 3[3] Oracle 8.0.5.0.0[4] Enterprise Edition Database Server Oracle Net8 Server (Listener) and Client 8.0.5.0.0 (TCP/IP Proto Adapter)	
Application Server	-	Dual 450MHz Pentium II 1 x RAID 1 Array	512MB RAM 1 x RAID 5 Array
		MS Windows NT4.0, Service Pack 3 Oracle Application Server 4.0.7.0.0[5] Enterprise Edition[6] Oracle Developer Server Rel 6.0 (including Client toolset)	
Development PC	-	Pentium II 350MHz 4GB Hard Drive	128MB RAM 800x600 (SVGA) video display res
		MS Windows NT4.0, Service Pack 3 Developer Release 6.0 MS IE 4.01 SP1 Adobe Acrobat Reader 4.0 Intersolv PVCS VM Version 6.0 (WINNT) Oracle JInitiator 1.1.7.11	
Client/User Browser (minimum spec)	-	Pentium 133MHz 1 GB Hard Drive (min 30MB free)	64MB RAM 800x600 (SVGA) video display res
		MS Windows NT 4.0, Service Pack 3 or MS Windows 95 MS IE 4.01 SP1 Adobe Acrobat Reader 4.0 Oracle JInitiator 1.1.7.11	

All 3-Tiers, Single Platform

To facilitate research and development, the author has configured Oracle 8.0.5 EE RDBMS, OAS 4.0.7, Developer Server 6 and Client IE4 browser on a standalone laptop PC (running NT Workstation 4.0 SP3). All can happily co-exist in the same Oracle Home[7]. Performance is quite acceptable on:

Pentium II 366MHz, 128MB RAM, 6GB Hard Drive

[3] Service Packs 4 and 5 are being evaluated for Millennium-compliance and integration with Oracle software.
[4] Strongly recommended database for implementation with Developer Server (Developer 6.0 Doc Addendum).
[5] Awaiting pending release of 4.0.8, which provides full Millennium-compliance and bug-fixes.
[6] Only supported release with Developer 6.
[7] OAS 4.0.7, according to OAS 4.0 Release Notes, can reside in same Oracle Home as NT 8.0.5 RDBMS.

The Design

The first month of the project solely focussed on two deliverables: a Design document - known as the Detailed System Design or DSD - and a Proof of Principle. The purpose of the latter was to confirm the suitability of Developer Server, both technically and functionally. The DSD acted as the main Terms of Reference for the project and included sections detailing the Existing Applications Overview, Proposed System Configuration, Migration Strategy, Application Web Deployment, Y2K Approach, Proof of Principle, Backup/Recovery Strategy, Standards and Conventions and Configuration Management. Specifically, the Migration Strategy would define:

- How each Module Type (Forms, Reports, SQL, Fortran) conversion would be achieved, including Checklists for developers to follow.
- Forms V3 issues (Data Types, Keyboard Mappings, Env Variables, V2-Style triggers, User Exits).
- Style Guide (GUI standards inc Fonts, Colours, Windows, Canvases and structure of Object Library).

Proof of Principle

The Proof of Principle was an extension of the earlier Proof of Concept, where the beta-version of Forms V6 was evaluated before the decision was taken to adopt the technology for the project.

In support of a user demonstration and feedback session, and also to prove the proposed Migration Strategies, sample Forms and Reports were migrated, together with a custom Menu system known as the Launchpad. The main purpose of the Proof of Principle was to confirm successful Web-deployment using the agreed system software (i.e. Developer Server Release 6 and OAS 4.0.7) and proposed deployment mechanisms. Secondly, assessments of GUI standards and Intranet integration were carried out.

The Conversion Process

The conversion process adopted was a manual one, albeit highly mechanistic. The major factors influencing this choice were:

- The need to re-engineer the Forms V3 interface (making extensive use of the newly available GUI widgets).
- Provision for full mouse freedom, replacing Key-Triggers with Event-driven ones.
- Application of Web feature restrictions.
- Enforcement of best practice, ensuring maximum interface usability and optimal Intranet performance.
- Lack of customisable third-party conversion tools.

Prior to each application conversion, user prototyping workshops took place. Interface usability, existing and potential, was analysed in conjunction with the VDS IT and user community. According to the agreed Style Guide from the Design phase, the Forms interface was re-engineered to provide a more feature-rich interface, which facilitated maximum usability and intuitive operation.

In converting the source V3 .INPs to Forms V6 format, the first step was to perform a basic conversion to Forms V4.5 modules. Using Developer/2000 Rel 1.6, this was performed as a one-off, automated batch process using F45GEN32.EXE and MIS' Batch MultiGen conversion utility. (There is no native batch conversion functionality with F45GEN32.EXE and IFCMP60.EXE). F45GEN32.EXE was used due to its track record, robustness and well-documented conversion functionality. (MIS had used this many times in previous conversion projects, whilst having problems with subsequent releases such as the Forms V5 F50GEN32.EXE).

The next step, using the resultant Forms 4.5 module, was to perform another basic conversion to Forms V6 modules. This was achieved, on a module by module basis, by simply opening the V4.5 .FMB module in the Forms V6 Builder tool.

Note that Forms V5 (Developer Release 2) was 'leapfrogged'. It was not required during the conversion process.

When opening a Forms pre-V6 module in Forms V6 Builder for the first time, all PL/SQL is automatically converted to PL/SQL 8. Additional, manual PL/SQL conversion issues were addressed in detail but are beyond the scope of this paper. Additionally, custom Search Engine tools were employed to identify potential Y2K issues in the existing V3 .INP source code. Such issues were addressed manually by developers.

Following the basic conversions, each developer adhered to a well-proven Forms Conversion Checklist (see excerpt below) as developed during the Design phase, which ensured compliance to all agreed conversion standards, and also provided traceability, consistency and accountability.

The Forms V6 API could have been used as a pre-processor to automate some of mechanical tasks, although we had little time to evaluate and implement it.

Following the complete conversion of a module (ready for Unit Testing), a completed Checklist was recorded. This included 'Before' and 'After' screenshots, as well as a signed summary of the main conversion steps that the developer had performed/adhered to. More importantly, each completed Checklist included any comments applicable to the module, such as Host Calls, OS Dependencies and non-standard module behaviour. Each completed Checklist was reviewed by the MIS Team Leader and, once approved, passed to VDS IT and the user community for final approval. (Approval by user community solely focussed on converted screen cosmetics).

Abstract from Forms Conversion Checklist

Check List Item
Existing Character Screen
Obtain screen dump to aid manual conversion process
New Screen Design
Devise screen prototype mock-up (utilising newly available supported GUI Widgets)
Screen Re-Design
Arrange new Canvas layouts and introduce new GUI Widgets
Remove all existing Box Line graphics
Apply Window and Canvas (Content, Stacked and Tab) Smart Classes (OLB)
Rename ROOT_WINDOW to WINDOW1, and re-engineer any Pop-Up Windows
Set Window Titles (for Main Window - Title plus Module Reference)
Create new Frame Graphics around blocks/logical screen areas (use and subclass STD_FRAME Smart Class from OLB)
Hardcoded References
Remove any hardcoded references (Tables, Help Text/Boilerplate and OS Paths)
Hot-Links
Introduce any necessary Hot-Links (to other Forms) by dragging and subclassing both STD_HOTLINK_FRAME and STD_HOTLINK_LI List Item from OLB.
Action Buttons
Introduce any necessary Action Buttons (for custom functions) by dragging and subclassing both STD_BUTTON_FRAME and STD_BUTTONS block from OLB.
Add any new Action Buttons (subclassing from STD_PUSH_BUTTON in OLB).
Blocks
Apply STD_BLOCK_SCROLLBAR Smart Class (OLB) to all Multi-Record blocks
Items
Migrate Text Item Data Types
Convert any Binary-State Text Items to Check Boxes
Apply Item Smart Classes (Date and non-Date) (OLB) to all Items (Text, List, CheckBoxes...)
Disable (Enabled = 'No', VisAtt = 'NON_EDITABLE_VA') any non-updateable (Insert, Update, Keyboard Navigable = 'No'), displayed Items.
Copy SET_ITEM_VISATTS PL/SQL Prog Unit (OLB) into Form and declare all Visible Items. (Call Prog Unit from W-N-R-I block trigger). Do once for each block
Align Items (Vertical Top, Horizontal Left)
Size Item Widths (see *Recommended Item Width* table)
Space Items (e.g. Iconic Buttons must be Stacked Horizontally with Text Item)
Master/Detail Blocks
Remove all standard V3 Master/Detail block synchronisation code (e.g. KEY-UP, KEY-DOWN, QUERY_DETAILS, CLEAR_DETAILS) and replace with explicit Forms V6 Relation objects. Note, that Single Record Master Blocks (with Dependent Child blocks) do not require change, where only navigation method (in Master block) is keyboard.
PL/SQL - Key to Event Triggers
Replace navigation key-triggers with event-triggers to cater for both keyboard and mouse navigation Replace POST-CHANGE and ON-VALIDATE-FIELD functionality with POST-QUERY and WHEN-VALIDATE-ITEM functionality.
PL/SQL - V2-Style Triggers
Rewrite any Forms V2.3-Style triggers as native Forms PL/SQL.
Operating System Dependencies
Replace HOST commands with RUN_PRODUCT when calling Reports modules
Replace User Exits with native Forms PL/SQL (in place of Fortran modules). (Utilise common code in PLL where possible)
Report any OS System Dependencies.

Forms Object Library

Having already utilised Object Libraries in Release 2 of Developer (i.e. Forms V5) on previous projects with great success, a Forms V6 Object Library (OLB) was defined at Design time. This repository of standard re-usable objects was implemented in time for the Proof of Principle exercise, allowing for on-the-fly, what-if analysis during the user presentation itself. On the spot changes such as colour and font preferences were quickly implemented through changes to centralised standard objects.
The aim of the OLB was to adopt a 'Subclass Everything' philosophy. This would mean that, potentially, any cosmetic interface changes required, either during or post-migration, would be painlessly implemented through a standard Object Library class. For example, in the middle of the migration, a new custom Smartbar Icon and Menu Item were required in the standard Menu module. Because all of the previously migrated forms had inherited from a Form-level Property Class in the OLB (which defined the standard Menu Module property), all that was required was a quick change to the OLB Form-level class and then subsequent generation of the migrated forms modules. Objects subclassed included Forms, Windows, Canvases, Graphics, Boilerplate, LOVs, Text Items, Blocks, List Items, Check Boxes, Push Buttons and Alerts.

Inheritance in the Object Library

There is no inheritance support within an OLB itself. Whilst it is preferrable to define standards in a hierarchical fashion (loose definitions at the head of the inheritance tree such as Standard Font then Standard Text Item and so on…), this approach was not adopted. Inheritance involving several standard objects has to be maintained externally of the OLB in a 'Scratch Pad' form. The concept of OLB inheritance (and its future upkeep) can therefore become complicated. To simplify the development and maintenance, it was decided that all objects in the OLB would have no dependencies on other objects in the OLB. (A single level of hierarchy was implemented).

It is important to note that the primary strength of OLBs was still utilised. That is, standard, reusable objects defined centrally in one place.

Problems with the Object Library approach

The definition of Form-level common properties (such as Menu Module and Validation Unit) could not be stored in the OLB as one, single 'true Forms object'. Rather, these common properties had to be encapsulated within a Property Class, which was then itself placed in the OLB.

Possibly a Forms V3 conversion issue, but occasionally when subclassing an existing item, Colour and Font properties would not be inherited. (Regardless of the subclassing method – either through the Property Sheet or SmartClassing through the Right-MouseClick Pop-Up). The only workaround for this was to re-inherit the Visual Attribute Group of the existing item. This would then have the effect of correctly inheriting the Colour and Font common properties from the OLB.

A Standard Date Text Item object was defined and placed in the OLB. Amongst other standard Date properties, such as Format Mask and Maximum Length, a Lowest Allowed Value (previously known as Range Lo) of '01/01/1900' was defined. This property alone could not be successfully inherited.

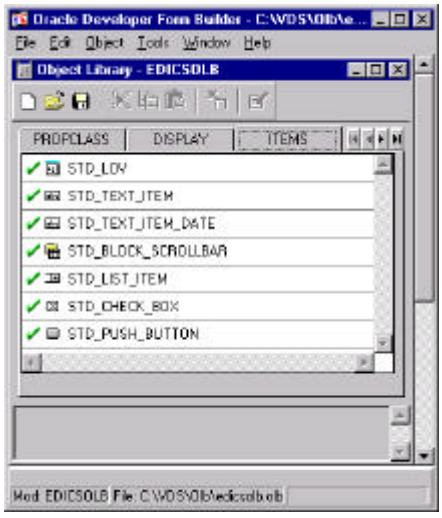
SmartClasses

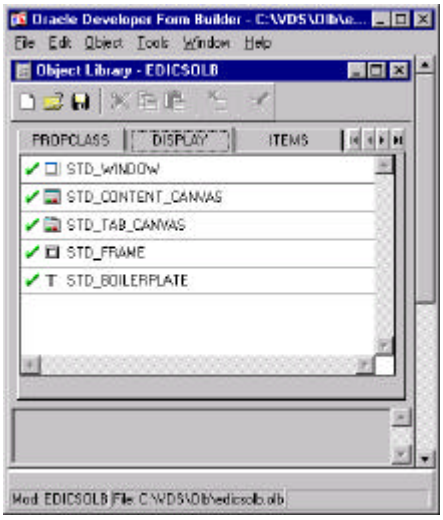
The usefulness of SmartClasses for such projects is highly recommended, especially when applying reusable GUI standards to existing legacy systems. Without SmartClasses, the subclassing of an object has to be performed on an individual object-by-object basis. (Something that can become extremely time consuming). By defining all OLB classes as SmartClasses (indicated by the 'green tick'), common OLB objects can be assigned to multiple 'like' objects in a single action. (Right-MouseClick Pop-Up). When SmartClassing, only OLB objects that are applicable to the object(s) selected are available.

Visual Attributes

Although standard objects can be defined in the OLB to declare common visual attributes such as font and colour, Visual Attributes were defined in the OLB. A main HCI requirement of the migrated application was to provide the user with dynamic visual cues. (i.e. Pale Yellow background for mandatory items, Pale Green background for display only items and so on…) Unlike OLB Objects and Property Classes, Visual Attributes can be assigned programmatically at runtime, allowing for the dynamic visual effect of Items when in the various System Modes (i.e. <Enter Query>, New Record and Update Existing Record).

Sample Object Library Classes





Intranet Integration

The front-end navigation/log-in form was designed to closely integrate (in 'look and feel' terms) with the VDS Intranet. Instead of relying on Oracle's standard Database Login/Authentication dialog, a custom Login screen was developed, which exhibits the VDS Intranet Look and Feel, including a company logo. (The overriding of the Forms-level ON-LOGON trigger and use of the LOGON Built-in allows such an implementation). In simple terms, the invocation of the application is achieved through the introduction of a standard HTML Hyperlink in the company Intranet Home Page. This Hyperlink directly references the application 'Base' HTML file, which specifies the activation of the Oracle JInitiator (with the appropriate startup parameters) and subsequent loading of the Forms Client applet.

Windowing within the Web Browser and Applet Area

The utilisation of Oracle JInitiator and agreed screen design made use of the Full-Screen mode in MS IE4. (Running in 800x600 64K colour video mode). Full-screen mode was chosen in order to view the entire border-less Applet area (or Forms displays) without the need for scrolling within the browser window. (It also provided more available 'real estate' for screen design). The Forms Applet is run in the same IE4 window that spawned the application. (This is default JInitiator behaviour). It was agreed that the overall ‘Look and Feel’ should be as far away from grey Client/Server as possible. (Using the ‘oracle’ setting of the lookAndFeel parameter and ‘teal’ setting of colorScheme helped achieve this aim).

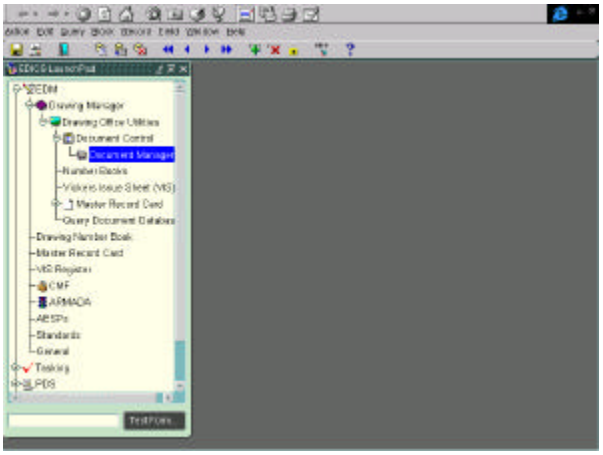
Sample Base HTML file (static, non-cartridge implementation)

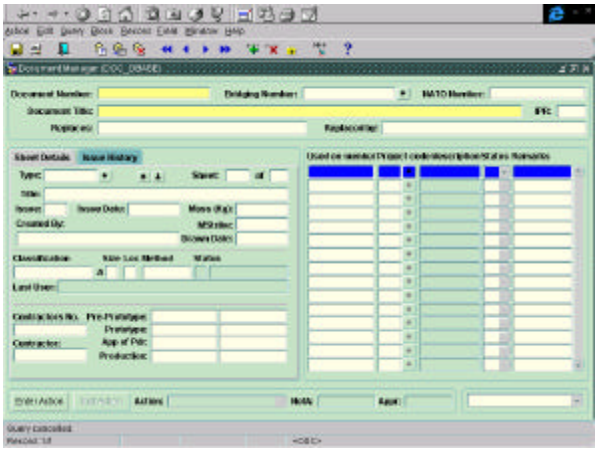
```
<HTML> <!-- FILE: edicsbase.html -->
<!-- Oracle Static (Non-Cartridge) HTML File Template (Windows NT) -->
<HEAD><TITLE>EDICS Web Site</TITLE></HEAD>
<BODY TOPMARGIN=0 LEFTMARGIN=0 BOTTOMMARGIN=0 RIGHTMARGIN=0>
<OBJECT classid="clsid:9F77a997-F0F3-11d1-9195-00C04FC990DC" WIDTH=803 HEIGHT=575
codebase="http://m_watson.mother.co.uk/webhtml/jinit.exe#Version=1,1,7,11">
<PARAM NAME="CODE"          VALUE="oracle.forms.engine.Main" >
<PARAM NAME="CODEBASE"      VALUE="/forms60code/" >
<PARAM NAME="ARCHIVE"       VALUE="/forms60code/f60all.jar" >
<PARAM NAME="type"          VALUE="application/x-jinit-applet;version=1.1.7.11">
<PARAM NAME="serverPort"    VALUE="9000">
<PARAM NAME="serverArgs"    VALUE="module=edics_menu.fmx">
<PARAM NAME="serverApp"     VALUE="default">
<PARAM NAME="splashScreen"  VALUE="splash_edics.gif">
<PARAM NAME="colorScheme"   VALUE="teal">
<PARAM NAME="lookAndFeel"   VALUE="oracle"> </OBJECT> </BODY> </HTML>
```

One of the main customer requirements for Intranet Look and Feel was not to have traditional Client/Server style Windows and grey Pull-Down menus. Therefore, on an authenticated login, a custom application navigation Launchpad form is displayed. This Launchpad has been implemented using the native Tree Control (akin to

Windows Explorer), which is one of the new features in Developer Forms V6. The main function of this Launchpad is to remain resident in the background whilst other Forms are in focus. At almost any time, control can be switched to the Launchpad, from which the user may launch other forms. (The modeless operation of any open forms is established through the use of the OPEN_FORM Built-in from the Launchpad Forms module. To achieve transaction independence the SESSION parameter was specified). The Tree Control was seamlessly deployed by Developer Server without problem.

Sample Launchpad and Forms screenshots





The Launchpad menu structure/contents is entirely data-driven by a new menu hierarchy database table, which defines the menu hierarchy alongside calling module references, icons etc. The introduction of a custom menu system meant that the native Database Roles security integration, as offered with standard .MMB Menu Modules, could not be utilised. Therefore, a new Menu Roles table was created. This is manipulated at runtime (in conjunction with standard Oracle User Role Privileges dictionary views) to allow for the dynamic availability of Menu Tree Items, depending upon the user logged in and associated granted Roles.

MDI or SDI?

A Multiple Document Interface (MDI) was implemented. This allowed for the fixed position of the Menu and Toolbar and, more importantly, the ability to navigate around open forms (including the Launchpad) using native MDI Window menu functionality. Additionally, the MDI background can be set by using the background parameter when invoking the Oracle JInitiator in the Base HTML launch file.

Default Menus and Toolbars

Each form is assigned a standard menu and toolbar. The implementation of a standard toolbar was achieved through the use of the native SmartBar functionality provided with Menu modules. Although, with Developer Server, Toolbar Canvases cannot be assigned to the MDI (parent container) window, the attachment of such a Menu module at form-level gives an MDI-level toolbar effect. Additionally no separate Toolbar canvases were required, which can prove cumbersome where multiple windows are concerned.

The console was assigned to the main application window of each form (normally WINDOW1), as it cannot be assigned to the MDI window when running Developer Server.

Unfortunately, Developer Server does not currently support Magic Items (e.g. Print Setup, Cut, Copy and Paste) and also Menu Item Separators as Client/Server implementations do.

Keyboard Mappings

There was no need to map legacy KEY-Fn triggers as these were replaced by Action Push Buttons. Nonetheless, logical to physical key mappings can be easily defined through editing of the Fmrweb.res file. Unlike Client/Server, this file can be directly manipulated using a standard Text Editor. This is fortuitous, as a bug in Oracle Terminal (as provided with the base production release of Developer 6) prevents the addition of any new key mappings (e.g. KEY-Fn keys)!

Fonts and Colours

The Arial 9 Point font was chosen for its accurate mapping from Design-time to Web runtime. Additionally, it was the font having the nearest 'look and feel' of standard text on the company Intranet. (The Arial Windows font is mapped to the Java SansSerif font at Web runtime).

The company corporate colour was adopted (Pale Green background) along with dynamic Visual Cue-ing. (Item background colours dynamically change at runtime according to their status and current System Mode). This colour coding highlighted Editable Mandatory, Editable Optional and Display-Only fields. This functionality was implemented using WHEN-NEW-RECORD-INSTANCE triggers, generic SET_ITEM_INSTANCE_PROPERTY code and OLB Visual Attributes. Unlike Client/Server implementations, Push Buttons could be coloured. The background colour of Checkboxes (White check-area), however, could not be controlled dynamically.

Once more, all font and colour properties were encapsulated into the OLB, allowing for quick, 'on-the-fly' changes during GUI standards assessments with the users, and also for longer term maintenance.

Reporting Mechanism

Within the browser, the report publishing was implemented through the use of the Adobe Acrobat Reader 4.0 Plug-In. All report output is in PDF (Adobe's Portable Document Format). See Reports section below.

The Reports

All existing SQL Reports modules were rewritten as new Reports V6 modules. (In a similar manual Checklist fashion to the Forms migration, alongside a Reports Style Guide). For report output, Adobe's industry standard PDF was chosen in preference to native HTML. PDF offered much more high-fidelity reporting features, which retained all advanced formatting at Web Runtime. The mapping of a formatted report to HTML was not as accurate and realistic as PDF. Furthermore, the Developer Server HTML Reports output does not truly support the concept of pages, meaning that rudimentary End Of Page Graphic Markers are inserted between report pages.

Integration with the Forms Server

It was decided that all reports would be called directly from the Launchpad menu system or from other forms. Unlike Client/Server, Developer Server does not currently support a Java implementation of Reports module Parameter Forms as, during Application Server processing, there is no user dialog such as processing indicators and Parameter Forms. For this reason, each report required a new front-end Parameter form, from where the user could specify any report parameters and then invoke the report.

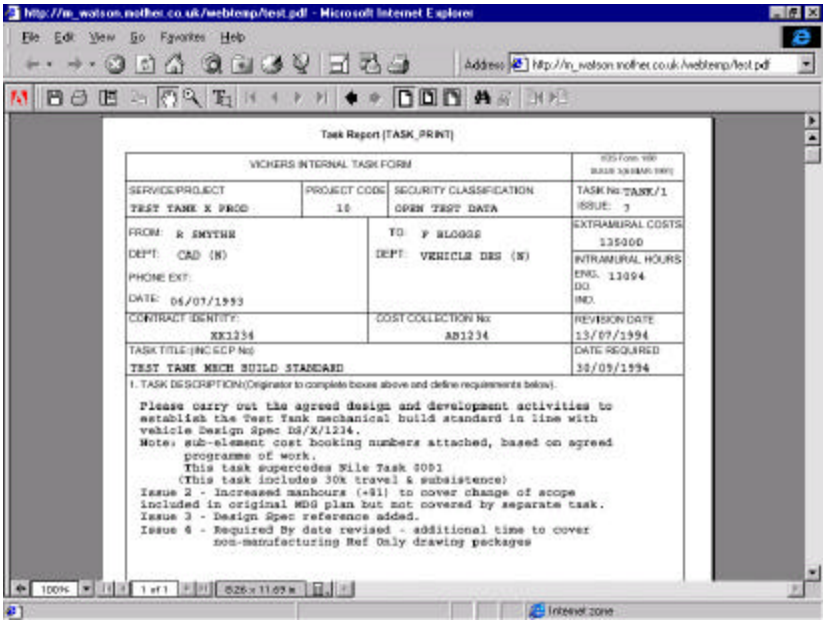
The RUN_PRODUCT Built-in was used with the appropriate parameters (see below) to call the report. Because this takes place in the context of the Forms Server, it was discovered that there was no need for a separate Reports Server (process or NT Service). It can only be assumed that the Forms Server has built in Reports Server functionality! Like the Reports Server, however, the Forms Server NT service must Log On As an account with access to a default printer. Otherwise the 'REP-3002 Error initializing printer…' occurs. Additionally, some of the Reports Server environment variables (Registry entries) still need to be created. (The Reports Server is obviously required for Reports Server URL CGI Run Requests). For completeness, the Reports Server was installed and configured as standard.

Report Output

Each front-end report parameter form gave the user two choices of Report Output – Preview or File. With Preview (DESTYPE='PREVIEW'), once the Reports Server (or really the Forms Server) constructs the PDF file (on the Application Server), a new IE window is opened (with the Adobe Reader 4.0 Plug-In), seamlessly displaying native Adobe functions and the report itself (in true WYSIWYG). From this window the user can preview the report, simply print it to any configured and accessible Windows printer, Save To File, Email, Search or Copy Text.

The RUN_PRODUCT <commode> parameter must be set to 'SYNCHRONOUS' to automatically download the report to the browser. (Although control does not return to the calling form until the report is produced).

Sample Report Output in browser



The purpose of File output (DESTYPE='FILE') was to allow the users to output all reports to unformatted ASCII report files, which could then be manipulated and imported into other applications/tools (such as MS Excel). The output files themselves are created and stored on the Database Server in a central user area (through use of the DESNAME parameter). In achieving File output, there were two main options:

- DESFORMAT = 'DELIMITED' (new option available to Reports V6)

- DESFORMAT = 'Generic / Text Only' (name of standard Windows NT Printer Driver for printer redirection to basic, unformatted ASCII file). This standard driver requires installation on the Application Server (and visibility from the Forms Server NT Service Log On As Account).

The new native 'DELIMITED' option was adopted due to its more accurate output. (Especially for standard tabular reports).

Host Operating System (OS) Integration

One of the main difficulties in migrating the applications from VMS to Windows NT was the extensive OS reliance the existing system had. With over 100 VMS DCL Command scripts and 30 Pro*Fortran/Fortran User Exits, complete re-design of some existing functionality was inevitable. Typically, the DCL scripts acted as parameter input and validation wrappers for the reports. (The new PDF Reports browser mechanism and front-end report parameter forms replaced most of these). The Pro*Fortran/Fortran modules typically performed OS file handling such as copying, moving and deleting files. (These User Exits were subsumed into native PL/SQL and Forms HOST commands). Although having used the Windows API access functions in the 'bonus' D2KWUTIL.PLL and associated DLL on previous projects, it was decided to use the more traditional HOST commands as they were more widely supported. (The use of D2KWUTIL in the Middle Tier is not supported by Oracle).

HOSTing on the Web

One major advantage of the Web – or Developer Server to be specific – is that any HOST commands are executed on the machine and Operating System on which the Forms Server is running (i.e. the Application Server). This made the migration much simpler as, in essence, the move was from a VMS Server-Centric architecture to a Windows NT Server-Centric one. Much access to central OS files was required (through HOST commands and common NT Command scripts), meaning that a Client/Server solution would have proved more problematic, with the usual choices being daemon processes (e.g. DBMS_PIPE) or mapped drives. Furthermore, with the HOST command taking place in the context of Windows NT and spawning full 32-bit applications, the FORM_SUCCESS Built-in was successfully used to test for success or otherwise of HOSTed OS commands. (Avoiding the invalidity/unpredictability of FORM_SUCCESS with Windows 95 and 16-bit applications).

The main problem with the HOST Built-in and Forms Server is that no screen output/interaction is supported locally at the user-presentation Tier, as the Application Server performs the command. Any screen interaction was, therefore, subsumed into native Forms PL/SQL, with any parameters passed through via the HOST commands. It is good practice to specify the HOST command with the NO_SCREEN parameter, preventing any separate window output on the Application Server console.

Operating System Environment Variables

Obviating the need for any hard-coding in the application (such as output/log directories and users areas), OS environment variables were defined. This presented a slight problem in that we needed one central place where they could be defined and accessed by all tools. Two choices were available - System Environment Variables as defined in System Properties Environment Panel; or the Windows Registry. The problem with the Windows Registry is that although both Forms and SQL*Plus tools have visibility, NT Command scripts do not. Any Registry entries were, therefore, accessed via Forms and their values passed as parameters when HOSTing to NT Command Scripts.

Oracle JInitiator

Oracle JInitiator Version 1.1.7.11 was utilised with MS IE4. (This version of JInitiator is bundled with both the Developer Server 6 and Developer 6 CDs). On installation, though, the JINIT.EXE setup file is placed in the ORACLE_HOME (e.g. C:\ORANT) and not the ORACLE_HOME\Jinit directory! For completeness and local testing, JInitiator was also installed on the Application Server, although this is not strictly necessary.

Client Deployment

JInitiator was seamlessly deployed to client browsers through the use of the ActiveX model as supported by IE. (JInitiator is automatically downloaded and installed to a client machine from the application server, the first time the client browser encounters the Base HTML file that specifies the use of JInitiator). Furthermore, by using the standard MIME type in the Base HTML file to specify the JInitiator version, the browser detects a request for a later version and notifies the user for subsequent download and installation of the new version. This was successfully implemented with the later version of JInitiator – 1.1.7.15 – as provided on the Developer 6 04 Jun 1999 Patch. Furthermore, JInitiator supports a silent installation mode, which does not require end user intervention.

Performance

Without doubt, Developer Server works best on a fast direct LAN connection (as was the case with our Intranet deployment). Running on an FDDI network (10 Megabits to the Desktop), we were able to take advantage of the relatively unconstrained network bandwidth and make the following decisions:

- Default Field-Level validation, although incurring more Client to Developer Server network roundtrips, was implemented, with obvious user benefits.
- Dynamic background colouring of Items according to their status via the use of SET_ITEM_INSTANCE_PROPERTY Built-in. Extensive modification of item attributes during runtime is not normally recommended for Internet deployment.

Utilisation of the pre-configured f60all.jar Java Archive (JAR) file. Despite its 1.5+MB size, this gave good application start-up time and minimised class demand loading during subsequent application execution. (The time to download this JAR file was negligible with the main delay comprising the unzipping of the JAR and then subsequent class loading and security authentication by Java). Future areas of investigation are to include the feasibility of locating the Developer Server class files locally on each client PC (using CLASSPATH) avoiding the need for JAR files and class downloading from the Application Server.

The default 20MB JInitiator cache size proved more than adequate in ensuring that all necessary JAR files remained valid in the local cache. To confirm what was downloading from the Application Server and what was being referenced from the local client cache, HTTP status codes were reviewed in the standard OAS Web Site log files for each referenced class and GIF file.

Optimisation
Developer Server optimisation included:

- Placement of Database Server in close vicinity of Application Server, due to their Client/Server relationship.
- Inclusion of custom GIF files (splash screen, custom toolbar and Launchpad/Tree Control icons) in f60all.jar. This allowed their local caching and, hence, avoided downloads from the server each time they were required.
- Minimal use of GIF image files (<=26Kb) for obvious network traffic reasons.
- Zero-use of 'heavy' mouse triggers, such as WHEN-MOUSE-ENTER/LEAVE, and also Forms Timers.

Without much re-work to legacy code, the full promotion of Developer Server's 'message diff-ing' could not be realised. (e.g. use of the Prompt property for Item boilerplate, and 'like' display order of Items in Object Navigator).

Hardware levels and Performance
It was discovered at an early stage that the more raw processing power available to Java, the better the performance (especially application start-up time) of the application on the client.

Comprehensive PC client benchmarking resulted in the recommendation of a minimum Pentium II 200MHz 64MB RAM hardware specification for deployment with Developer Server 6.

Testing
Ease of deployment came into its own during Unit and Integration Testing. Simply broadcasting the application's URL meant that we could feasibly enlist more testers, without the need for extra high-spec development workstations. (Basic thin-clients were all that was necessary, requiring minimal configuration). Leveraging this simplicity, we were able to allow the users to beta-test the application prior to it going live, allowing them visibility of the product well in advance of the planned go-live date.

Prior to formal testing, however, basic cursory testing of each unit (or isolated module) was performed in native Client/Server mode. Occasionally, the testing of partly-migrated modules led to the freezing of the browser session. From the application server, it was difficult to identify and terminate Forms Server runtime sessions, which sometimes meant that the Forms Server and OAS core processes required restarting.

Web Look and Feel
The Web Developer Server runtime (real 3-Tier mode) was always used to confirm GUI standards and presentation details. The Forms Builder native Run Form Web facility, however, was not used. Initially, it appeared not to fully resemble the look and feel of Developer Server and JInitiator over the Intranet, without some custom configuration. It proved simple enough to use the Base HTML file from a browser (as end-users would) for incremental testing.

Known Problems with Developer Release 6.0 and OAS 4.0.7

During design and applications conversion, some problems were discovered with the Oracle software. However, none of these posed any serious risk to the success of the project. To follow is a list of the major problems/issues encountered, together with their workarounds. It should be noted that new releases (including Software Patches) of the core software versions as detailed in the 'The Hardware and Software Platforms' section may not exhibit these problems:

- 'Hidden' Server Configuration Wizard (workaround: configure Developer Server manually).
- 'REP-3002 Error initializing printer…' (workaround: Forms Server NT Service must have visibility of a default printer when integrating Reports with Forms via the RUN_PRODUCT Built-in).
- ifsrv60 –listen port=<nnnn> –install <servname> - Forms Server not installing (workaround: ifsrv60 –install<servname> - Forms Server can only be installed as a Windows NT Service with default port of 9000, or run as non-NT Service process).
- 'FRM-99999 Network error – java.net.SocketException: Conection reset by peer…' when attempting browser connection to Developer Server (workaround: remove FORMS60_TIMEOUT Registry entry).
- Cannot view V2-style triggers (workaround: use Forms 4.5 or view source .INP).
- No Magic Items and Menu Item Separator support on Web (workaround: none).
- No Combo Box List Style for List Items on Web (workaround: none, Oracle doc error).
- Truncation/clipping of icons in Web buttons (workaround: resize buttons and GIF icons).
- Missing Oracle Terminal on Custom Installation of Developer 6 (workaround: perform standard Development installation or modification of NT.PRD).
- Cannot add custom keys (i.e. KEY-Fn) with Oracle Terminal (workaround: none for Client/Server, Web – edit fmrweb.res instead).
- Developer 6 Patch 04-Jun-1999 (inc JInitiator 1.1.7.15) – mouse navigation not changing forms internal focus and thus not firing event triggers (workaround: remove patch).
- OAS 4.0.7 and Developer Server 6 base production releases with WindowsNT4 Service Pack 4 (workaround: copy DLL from Developer Server CD, and acquire OAS Patch - TAR#2242195.1 - for required OAS DLL).
- OAS 'OracleStartOAS 4.0…' NT Service not correctly starting OAS on reboot (workaround: manual start or copy psapi.dll to SYSTEM32 area).
- OAS Manager JDK 1.1 Tree Control Applet not always loading on IE4.0 SP1 (workaround: none).
- PDF corruption when downloading report from Application Server to browser (workaround: reported fixed in OAS 4.0.8).
- Invalid Page Faults/Dr Watson when integrating Intersolv PVCS VM Version 6.0 (WINNT) (workaround: do not open files after Check-Out…).
- Temporary files accumulation in webtemp area on Application Server (workaround: implement batch job to auto clear down on frequent basis).

The Future…

Plans are underway to evaluate the running of the migrated applications natively within MS Internet Explorer 5.0. Although some extra client configuration is required, the benefits of not deploying and using Oracle JInitiator are nonetheless attractive.

The implementation of custom controls using JavaBeans and the new Java Pluggable Component interface is to be explored.

Plans are underway to reverse engineer the entire migrated application into Designer. We currently have a Designer Release 2 repository defining the application database. At the end of the project, all modules (Forms, Reports, SQL and utilities) will be Design-Captured into Designer, allowing for an easier future maintenance path.

Conclusion

Overall, we have been impressed with the new Developer Server and Developer 6 client software. Although earlier versions of the product (i.e. 1.3.2, 1.6 and 2.1) were slightly hindered by performance and configuration issues, Release 6 has addressed these areas and provided developers with a far more suitable and stable platform from which to deploy applications to the Web. That said, however, we feel Oracle could improve their documentation on both Developer Server and JInitiator. Pertinent information is sometimes distributed across core documentation, release notes, example files and on-line help. (Rather than in one core central place). Furthermore, crucial configuration details, like example JInitiator Base HTML files and environment variable setup, were incorrect and caused much frustration during the early days, when it was often difficult obtaining support for the latest release.

We were especially impressed with the lack of Invalid Page Faults during design-time. It appears that Developer 6 is extremely stable with Windows NT. The exception to this was the PVCS integration, which occasionally resulted in crashes.

Finally, 'Zero Client Deployment' and 'Build Once Deploy Everywhere' were almost achieved. Ease of deployment cannot be emphasised enough, especially during testing.

Bibliographical References

- Oracle White Paper – Migrating SQL*Forms Applications to Developer/2000 Server November 1997 Claire Dessaux.
- Oracle Developer Server 6.0: Client Platform Support MS IE 5.0 Technical Note April 1999 EIT-DE-TN-002.
- Oracle White Paper – Oracle Developer and Object Libraries October 1998.
- Oracle Developer Server: How to Tune for the Deployment of Internet Applications Techn White Paper April 1999.
- Oracle JInitiator Technical FAQ March 1999.
- Developer Forms Server FAQ Technical White Paper April 1999.

Contact Information

If you have any questions with reference to this paper or queries in general regarding Oracle Internet deployment then please contact Motherwell Information Systems, Tel +44 (0)191 5163000, email info@mother.co.uk.

Disclaimer

All views expressed in this paper are those of author, and do not necessarily reflect those of MIS.

About the Author

Mark Watson is a Principal Consultant working with Motherwell Information Systems (MIS) in the North East of England. With over seven years Oracle development experience and an Oracle Certified Professional, Mark has worked at MIS for over four years, where he has enjoyed key positions on many Oracle design and implementation projects.

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

Web-Deploying Oracle Character-Based Systems with Oracle Developer Server Rel 6
Technical White Paper
Author: Mark Watson, Motherwell Information Systems