Macrotone
Consulting Ltd.

Oracle Forms And Reports 10G

Author:	**G S Chapman**
Date:	**12th February 2007**
Version:	**1.1**
Location of Document:	

DOCUMENT HISTORY

Version	Date	Changed By:	Remarks
1.0	12/02/07	G S Chapman	Original Version
1.1	30/07/11	G S Chapman	Sanitised version for the web.

DOCUMENT DISTRIBUTION

Copy No	Name	Role	Organisation

DOCUMENT REFERENCES

Document Name	Originator	Part Number	Version	Date
Oracle Forms Service – Using Run_Report_Object() to call Reports with a parameter form	Oracle Corporation			February 2004
Oracle Application Server 10g – Integrating Oracle Reports in Oracle Forms Service applications.	Oracle Corporation			May 2004
Oracle Forms Developer – WebUtil User’s Guide	Oracle Corporation		1.0.6	October 2005
Oracle Forms – Migrating Forms Applications From Forms 6i. 10g (10.1.2.0.2) for Windows and Unix	Oracle Corporation	B15572-01	10.1.2.0.2	August 2005
Oracle® Application Server Containers for J2EE Services Guide 10g Release 2 (10.1.2) for Windows or UNIX	Oracle Corporation	B14012-01	10.1.2	
Oracle® Application Server Containers for J2EE Security Guide 10g Release 2 (10.1.2)	Oracle Corporation	B14013-01	10.1.2	

TABLE OF CONTENTS

TABLE OF FIGURES

TABLES

PURPOSE OF DOCUMENT

This document summaries the results of investigating Forms and Reports migration from Developer 2000 to release 10G. It includes the initial review and the results of problems encountered during the conversion of various forms and reports to the new version.

The document also, despite the document title provides details of the Business Intelligence [Discoverer] application, covering installation, configuration and preliminary developments.

Glossary

Term	**Description**
DDE	DDE, or Dynamic Data Exchange, is a protocol which allows Windows applications to exchange data. Data is exchanged in the form of short messages, which contain either a short item of data, or a pointer to a location in memory where this data is available. DDE is deprecated.
OLE	OLE stands for Object Linking and Embedding. OLE was originally intended to complement DDE, and to provide a simple method of linking files, so that for example a spreadsheet linked from a word processor would have the latest figures; and embedding the application, so that the user would be able to edit the spreadsheet without exiting the word processor.
COM/DCOM	COM, or Component Object Model, is an extension to OLE which also took in OCX, based on a modified version of DCE/RPC. Previously, object interfaces were specified using ODL (Object Description Language); with COM, these were specified using DCE IDL, with an extended syntax. DCOM stands for "Distributed COM". The only real difference here is that when an object is accessed on the same machine, it is accessed in the same way as a C++ class is, but if the object is on another machine, the RPC mechanism is used.
JPC or PJC	Java Pluggable Component. Pluggable Java Components provide a low level API, which allows you to extend the Forms Java client. Each Oracle Forms native UI component, as defined in the builder – a field, a button and so on, has an equivalent representative in Java. The Java representations of the Oracle Forms native UI components are created using the Java lightweight component model. Using the lightweight component model means that the Oracle Forms Java UI components are rendered completely, rather than relying on the peer UI objects provided by the windowing system of the client operating system. This means that the Oracle Forms Java UI components appear visually the same and have similar behaviour across different client operating systems.
FBean	Although the PJC mechanism provides an extremely useful API it requires that one writes bespoke Java code to extend the client. So in Oracle9*i* Forms a new simpler mechanism was introduced to allow one to integrate JavaBeans and Java Applets as custom components without writing any Java Code at all. This is all handled using a PL/SQL package in Forms called FBean. FBean provides a full PL/SQL API to allow interaction with any public methods, properties and events that a JavaBean or Applet exposes.
EUL	The EUL or End User Layer is a Discoverer component comprising a number of tables and views used to create an abstraction layer on to of a number of database objects. This EUL is then used to hold 'Business Areas' that an end user can then select defined folders from containing the business perspective.

1 Introduction

This paper describes the results of an investigation into the various options available for a new Web enabled forms and reports services.

Oracle web based forms and reports first became available with release 6i of the Developer Suite. This was enhanced with release 9.0.4 (10G) of the application server, which provided the full frame work of the Application Server including Single Sign on (OID), portal etc. However the need is only for a cut down version of the Application Server and Oracle provided a separate CD with only the Forms and Reports components. This effectively removes the whole of the 'Infrastructure' components of the Apps Server.

The current version of Oracle Application Server is 10.1.3. However inspection of the documentation reveals that there is no stand alone version for Forms and Reports server. [This may change as it looks like the Forms and Reports server is usually released later, so it is always possible that there may be a 10.1.3 version available within the time frame.] The previous version 10.1.2 however did provide this functionality, so this seems to be the obvious place to start.

The current version of Oracle Developer Suite is 10.1.2. This provides the developers with 2 possible options:

- J2EE Development: A lightweight installation that allows one to develop Java and Enterprise Edition (J2EE) applications using Java, HTML, XML, and SQL. This includes testing capability with Oracle Application Server Containers for J2EE (OC4J), and allows you to extend applications with business intelligence using Oracle Business Intelligence Beans (also referred to as "OracleBI Beans"). – May not be relevant for some environments.
- Complete: This installs Oracle Forms Developer, Oracle Designer, Oracle Reports Developer, and Oracle10g JDeveloper. This also installs Oracle Application Server Containers for J2EE (OC4J) and the relevant Oracle Application Server runtime services (Oracle Application Server Forms Services and Oracle Application Server Reports Services), and configures OC4J as the default listener for testing purposes.

In the work environment each of the developers would [probably] require the complete installation on their local PCs. The developers could then test their forms and reports prior to deployment upon the Application Server.

There was a patch issued by Oracle for the Application Suite (including Development Suite) on the 26th October 2006, which will bring the version up to 10.1.2.2. This patch has been successfully applied to all installed environments upon the test system.

The quarterly security patch for January 2007 has also been successfully applied.

2 Options

There are a few options available to the Business. The decision as to which will be deployed has yet to be made but will depend upon a number of decisions, not least the financial aspect. There are a few consequences of the decision such as the requirements for the Application server machine specifications. These are as follows:

- Java Developer topology: Intended for Java developers. This topology installs a **J2EE and Web Cache** middle tier, on which to deploy your applications.

- Portal and Wireless Developer topology: Intended for Java developers who want to use OracleAS Portal, Oracle Application Server Wireless, Oracle Internet Directory, or Oracle Application Server Single Sign-On features, in addition to J2EE and Web Cache features. This topology installs a **Portal and Wireless** middle tier and the **OracleAS Infrastructure**.
- Business Intelligence and Forms Developer topology: Intended for developers who want to use OracleAS Personalization, OracleBI Discoverer, OracleAS Reports Services, or OracleAS Forms Services features, in addition to J2EE and Web Cache and Portal and Wireless features. This topology installs a **Business Intelligence and Forms** middle tier and the **OracleAS Infrastructure**.

2.1 System Requirements

	J2EE and Web Cache	Portal and Wireless	Business Intelligence and Forms	OracleAS Infrastructure
Memory (see (1) below)	512 MB	1 GB	1 GB	1 GB
Disk space	400 MB	760 MB	1.5 GB	3.2 GB (see (2) below)
Space in TEMP directory	256 MB (see (3) below)	256 MB (see (3) below)	256 MB (see (3) below	256 MB (see (3) below)
Total pagefile size (virtual memory)	1.54 GB	1.54 GB	1.54 GB	1.54 GB

Table 1 - Minimum System Requirements

Notes:

(1) If you plan to install OracleAS Infrastructure plus either Business Intelligence and Forms or Portal and Wireless on the same computer, you need to have at least 1.5 GB of memory and 2.0 GB total pagefile size.

(2) Includes OracleAS Identity Management and OracleAS Metadata Repository. You can install the data files for the OracleAS Metadata Repository database on a disk that is different from the disk where you are installing OracleAS Infrastructure. If you do this, make sure the disk for the data files has at least 1.6 GB of free space.

(3) 55 MB of free space in the TEMP directory are required to run the installer in addition to the 256 MB of space required for the installation.

2.2 Questions

It is not a supported option to incorporate the BI OC4J component within the Forms/Reports middle tier without the need for the full Infrastructure components. Hence the decision to install 2 separate middle tiers.

From the Oracle documentation B13918 **Oracle® Business Intelligence Discoverer Configuration Guide** it states that it is possible to install BI as a standalone installation (i.e. installed from an Oracle Business Intelligence standalone CD).

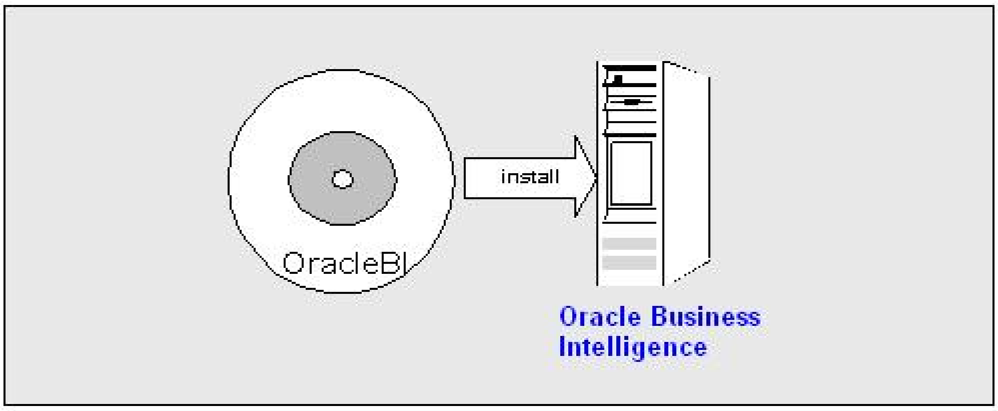


Figure 1 - BI stand alone configuration

An OracleBI standalone installation is not associated with an OracleAS Infrastructure, and therefore has a limited number of components available. An Oracle BI standalone installation contains the following components:

- Discoverer Plus Relational and Discoverer Plus OLAP
- Discoverer Viewer
- Discoverer Portlet Provider (installed but not operational until the Discoverer installation is associated with an OracleAS Infrastructure)
- OC4J
- Oracle HTTP Server
- OPMN
- Oracle Application Server Control
- OracleAS Web Cache

Note: End users start Discoverer using a direct login page. Discoverer connections and OracleAS Single Sign-On are not available unless the OracleBI standalone installation is associated with an OracleAS Infrastructure.

2.2.1 Steps to deploy Discoverer under the Forms and Reports Middle Tier

This is not for the faint hearted. It is not recommended currently as a supported environment and hence has not been used for the installation. It is included here to document the work carried out to evaluate the feasibility of such a configuration.

There are a number of steps to carry out, one of which is to deploy the 'discoverer.ear' file to the Middle tier. Also required is the configuration of the application itself, seen from the console home page. Also require the binaries that would usually be located in the Oracle_HOME/bin dorectory. i.e. eulapi etc.

1. Install the BI release to enable the full code tree to be obtained.
2. Copy the discoverer directory from the BI home to the FR home.
3. Edit the files in the new $OH/discoverer/conf directory [OH = Oracle_HOME directory] to change all references to the old OH to point to the new OH. NOTE: If you omit this

step then once the deployment is done you will be unable to remove the old OH because the various files will still point to the old OH

4. In the Console navigate to the BI-Forms configuration and opt to deploy an EAR file.
5. For the source use $ORACLE_HOME/discoverer/deploy/discoverer.ear
6. For the Name choose 'Discoverer'
7. Click Finish and it should all deploy successfully after a few minutes.

There will still be no entry for the main Discoverer system component upon the front page of the Console. Have tried editing various files in $OH/Apache/Apache/conf, also $OH/opmn/config. Also put a new discoverer directory in $OH/sysman/ias all without much success. This affects the Discoverer Preference Server initialisation.

3 Forms and Reports Services AS instance.

Oracle Application Server Forms and Reports Services enable one to install and configure Forms and Reports Services without the need to install and configure the entire Oracle Application Server. This is called a standalone installation of Forms and Reports Services. One can also configure the standalone Forms and Reports Services instance to use the Oracle Identity Management and Oracle Application Server Metadata Repository services of an Oracle Application Server Infrastructure.

Forms and Reports Services standalone installation is best suited for users who want to upgrade Forms and Reports applications to the Grid environment in two phases. In phase one, the move is to the Grid environment by upgrading their client/server-based Forms and Reports applications to Web-based applications. In phase two, users can then choose to use the services offered by an existing Oracle AS Infrastructure installation

With the Forms and Reports Services, one will have access to the following features:

Oracle Application Server Forms Services

> Oracle Application Server Forms Services deploys Oracle Forms with database access to Java clients in a Web environment. Oracle AS Forms Services automatically optimizes class downloads, network traffic, and interactions with the Oracle database. Forms applications are automatically load-balanced across multiple servers; thus, they can easily scale to service any number of requests.

Oracle Application Server Reports Services

> Oracle Application Server Reports Services provides an easy-to-use, scalable, and manageable solution for high-quality enterprise reporting and publishing. Using Oracle Reports, one can publish data generated by multiple sources in various formats (paper layout, Web, or data interchange format). This provides flexibility in the presentation of data. Oracle AS Reports Services is part of Oracle Application Server.

Oracle HTTP Server

> Oracle HTTP Server, built on Apache Web server technology, is the Web server that Oracle Application Server uses. It offers scalability, stability, speed, and extensibility. It also supports Java servlets, Java Server Pages (JSPs), Perl, PL/SQL, and CGI applications.

Oracle Application Server Web Cache

> Oracle Application Server Web Cache is a server-accelerator caching service that improves the performance, scalability, and availability of frequently used Oracle E-business Web sites that run on the Oracle platform. By storing frequently accessed URLs in virtual memory, Oracle Application Server Web Cache eliminates the need to repeatedly process requests for those URLs on the Web server. It also caches both static and dynamically-generated HTTP content from one or more applications Web servers.

Oracle Application Server Containers for J2EE

> Oracle Application Server Containers for J2EE (OC4J) is a complete set of J2EE containers written entirely in Java that execute on the Java Virtual Machine (JVM) of the standard Java Development Kit (JDK).

Oracle Enterprise Manager

> Oracle Enterprise Manager Application Server Control (henceforth referred to as Application Server Control) provides one with Web-based management tools that one need to monitor, administer, and configure multiple Oracle Application Server instances and its components. By default, Application Server Control is installed with every instance of Oracle Application Server. One can deploy applications, manage security, and create and manage Oracle Application Server clusters.

Application Server Control consists of the following:

- The Enterprise Manager home pages one uses to manage Oracle Application Server and its components. These Web pages provide one with a high-level view of the Oracle Application Server environment. From these pages one can drill down for more detailed information on administration, configuration, and performance monitoring. These pages also let one administer Oracle Application Server, its components, and deployed applications.
- The underlying software technologies that keep track of the Oracle Application Server instances and components. These technologies automatically perform the necessary management tasks. For example, these technologies discover the components of each Oracle Application Server instance, gather and process performance data, and provide access to application configuration information.

Oracle Process and Management Notification

Oracle Process and Management Notification (OPMN) provides process control and monitoring services for Oracle Application Server instances and their components such as Forms and Reports Services. It gathers component status information and distributes the information to the relevant components. Application Server Control uses OPMN for such tasks as starting and stopping the components of the Oracle Application Server instance.

Distributed Configuration Management

Distributed Configuration Management (DCM) manages configurations among Oracle Application Server instances with a common Oracle Application Server Metadata Repository. It enables cluster-wide deployment of Oracle Application Server; thus, enabling one to deploy an application to one instance and have it automatically propagated to the entire cluster. One can also make a single host or instance configuration change to one instance and have it propagated across all instances in the cluster. Application Server Control uses DCM to make configuration changes and to propagate configuration changes and deployed applications across the cluster.

Other software: The following software may well assist in the migration of the forms. I know for a fact that some of the forms make use of the D2KUtil package, which will not work with the web based forms. In addition the some of the current forms make use of functionality that can be ‘duplicated’ easily through the use of WebUtil so if only for that reason it will be required.

WebUtil

WebUtil provides a simple way to achieve client side integration while running Oracle Forms on the Web. It simplifies the upgrade of existing applications to the web, and enhances the capabilities of Web based Oracle Forms applications.

WebUtil Core Features

• Text_IO:	Read and write text files on the client machine.
• File transfer:	Move from between the client, application server and database.
• Tool_Env:	Read client side variables
• File Manipulation:	Manipulate client side files.
• C API on the client:	Interface with client side C.
• Client machine information:	Read information from the client machine.
• Host:	Run Host commands on the client machine.
• READ/WRITE_IMAGE_FILE:	Read and write client side images.
• OLE2:	Integrate with client side OLE (e.g. Word and Excel)
• Get_File_Name:	Use a file selection dialog on the client machine.
• Enhanced Host commands:	Host command can now call back into Forms!
• D2KWUtil features:	Client side interface into the D2KWUtil package.
• Browser functions:	Integrate with the browser.

4 Generic changes made for Forms

4.1 Calling Reports from Forms

Running the Forms migration assistant changes the form and introduces an object called RP2RRO and some code to enable the object to be used to call reports. Some mixed success has been achieved using this generated code and hence it has not been actively pursued in the testing.

There are basically 2 main methods of calling reports from the forms environment. The older (?) one seems to involve using the 'run_report_object' procedure. The second involves the use of the web.show_document procedure. The later is the easiest to implement. These 2 methods are described in a couple of Oracle white papers. 'Oracle Forms Service – Using Run_Report_Object() to call Reports with a parameter form' and 'Oracle Application Server 10g – Integrating Oracle Reports in Oracle Forms Service applications.' Both of these papers are available on Oracle Technet.

The suggestion is that a generic procedure called 'run_report_object_proc' is introduced into the form. This procedure is then called from the various locations within the form with the appropriate parameters. A single 'REPORT' object is required and it seemed sensible to use the 'RP2RRO' report object created as part of the migration program.

The run_report_object_proc is shown below, with some debugging information useful for displaying the run state. These should all be commented out or removed prior to the forms going live.

```
PROCEDURE RUN_REPORT_OBJECT_PROC(report_id REPORT_OBJECT,
                              report_server_name varchar2,
                              report_format varchar2,
                              report_destype_name number,
                              report_file_name varchar2,
                              report_otherparam varchar2,
                              reports_servlet varchar2)
IS
report_message   VARCHAR2(100)  := '';
rep_status       VARCHAR2(100)  := '';
vjob_id          VARCHAR2(4000) := '';
hidden_action    VARCHAR2(2000) := '';
v_report_other   VARCHAR2(4000) := '';
i                NUMBER(5);
c                CHAR;
c_old            CHAR;
c_new            CHAR;
sso_user         VARCHAR2(100);

BEGIN
      /*set Reports properties for run_report_object*/
      SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_COMM_MODE,SYNCHRONOUS);
      SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_FILENAME,report_file_name);

SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_DESTYPE,report_destype_name);
      SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_DESFORMAT,report_format);
      SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_SERVER,report_server_name);

      /* Create string for pfaction parameter */
      hidden_action := hidden_action ||  '&report='
        ||GET_REPORT_OBJECT_PROPERTY(report_id,REPORT_FILENAME);

      hidden_action := hidden_action ||  '&destype='
```

```
   ||GET_REPORT_OBJECT_PROPERTY(report_id,REPORT_DESTYPE);

hidden_action := hidden_action ||  '&desformat='
   ||GET_REPORT_OBJECT_PROPERTY(report_id,REPORT_DESFORMAT);

    -- Not using sso
    hidden_action := hidden_action ||'&userid='
      ||get_application_property(username)||'/'
      ||get_application_property(password)||'@'
      ||get_application_property(connect_string);

-- message('Set up hidden action complete.');
-- report other parameters are passed as key value pairs
-- "key1=value1 key2=value2 ..."
-- the following loop replaces the delimiting blank with an ’&’
-- used on the Web.
-- c_old is initialized with a dummy value
c_old := '@';
FOR i IN 1..LENGTH(report_otherparam) LOOP
   c_new:= substr(report_otherparam,i,1);
   IF (c_new = ' ') THEN
      c :='&';
   ELSE
      c := c_new;
   END IF;
   -- eliminate multiple blanks
   IF (c_old = ' ' and c_new = ' ') THEN
      null;
   ELSE
      v_report_other := v_report_other||c;
   END IF;
   -- save current value as old value
   c_old := c_new;

END LOOP;
-- message('After for loop');
hidden_action := hidden_action ||'&'||v_report_other;
-- report_servlet contains the full path to the Reports Servlet
-- Example1, Forms and Reports are on the same host
-- reports_servlet:=’/reports/rwservlet’
-- Example2, Forms and Reports are on separate hosts
-- reports_servlet:=’http://host:port/reports/rwservlet’
hidden_action := reports_servlet||
   '?_hidden_server='||report_server_name||encode(hidden_action);
-- Set pfaction to the report

message('Calling '||hidden_action);
message('Force display');

SET_REPORT_OBJECT_PROPERTY(report_id,REPORT_OTHER,'pfaction='
    ||hidden_action||' '||report_otherparam);
-- message('Calling run object');
report_message:= run_report_object(report_id);
rep_status := report_object_status(report_message);
-- message('Read status '||rep_status);
IF rep_status='FINISHED' THEN
   vjob_id := substr(report_message,
      length(report_server_name)+2,length(report_message));
   -- message('Calling show_documnet');
   WEB.SHOW_DOCUMENT(reports_servlet||'/getjobid'||vjob_id
       ||'?server='||report_server_name,' _blank');
ELSE
   -- handle errors
```

```
         message ('Error');
       END IF;
END;
```

One additional function is required (called from the above procedure). This is the encode function supplied below:

```
FUNCTION ENCODE (URL_PARAMS_IN Varchar2)
   RETURN VARCHAR2
IS
   v_url VARCHAR2(2000) := URL_PARAMS_IN; -- Url string
   v_url_temp VARCHAR2(4000) :=''; -- Temp URL string
   v_a VARCHAR2(10);
   -- conversion variable
   v_b VARCHAR2(10);
   -- conversion variable
   c CHAR;
   i NUMBER(10);
BEGIN
   FOR i IN 1..LENGTH(v_url) LOOP
      c:= substr(v_url,i,1);
      IF c in (';', '/','?',':','@','+','$',',',' ') THEN
         v_a := ltrim(to_char(trunc(ascii(substr(v_url,i,1))/16)));
         IF v_a = '10' THEN v_a := 'A';
         ELSIF v_a = '11' THEN v_a := 'B';
                  ELSIF v_a = '12' THEN v_a := 'C';
         ELSIF v_a = '13' THEN v_a := 'D';
         ELSIF v_a = '14' THEN v_a := 'E';
         ELSIF v_a = '15' THEN v_a := 'F';
         END IF;
         v_b := ltrim(to_char(mod(ascii(substr(v_url,i,1)),16)));
         IF v_b = '10' THEN v_b := 'A';
         ELSIF v_b = '11' THEN v_b := 'B';
         ELSIF v_b = '12' THEN v_b := 'C';
         ELSIF v_b = '13' THEN v_b := 'D';
         ELSIF v_b = '14' THEN v_b := 'E';
         ELSIF v_b = '15' THEN v_b := 'F';
         END IF;
         v_url_temp := v_url_temp||'%'||v_a||v_b;
      ELSE
         v_url_temp :=v_url_temp||c;
      END IF;
   END LOOP;
   return v_url_temp;
END;
```

From within the form some code as follows is used to make the call to run_report_object_proc. This code is extracted from the TEST application and is provided as a guide for other developers.

```
PROCEDURE which_database_report IS
  report_id Report_Object;
  v_report_other VARCHAR2(4000) := '';
  v_rep_server VARCHAR2(32) := 'server_name';

BEGIN
   report_id := find_report_object('RP2RRO');
   IF :BL_MAIN_MENU.LI_DATABASE = 'STORES' THEN
            v_report_other := 'paramform=no';
            run_report_object_proc(report_id,
v_rep_server,'htmlcss',CACHE,'TEST/Store_List.rdf',
                          v_report_other,'/reports/rwservlet');
  ELSIF :BL_MAIN_MENU.LI_DATABASE = 'FILE_TYPES' THEN
          v_report_other := 'paramform=no';
          run_report_object_proc(report_id,
v_rep_server,'htmlcss',CACHE,'TEST/File_List.rdf',
                         v_report_other,'/reports/rwservlet');
  ELSIF :BL_MAIN_MENU.LI_DATABASE = 'RECORD_TYPES' THEN
          v_report_other := 'paramform=no';
          run_report_object_proc(report_id,
v_rep_server,'htmlcss',CACHE,'TEST/Record_List.rdf',
                         v_report_other,'/reports/rwservlet');
  END IF;
  :BL_MAIN_MENU.LI_DATABASE := '';
END;
```

Note the use of the v_report_option to specify whether a parameter form is required. [An improvement implemented in the actual form is to define a user parameter to hold the report_server specification to centralise it in one place.]

An alternative which just uses the web.show_document syntax is as follows:

```
vc_url := '/reports/rwservlet?server=server_name&report=/TEST/File_List.rdf'||
        '&desformat=htmlcss'||
        '&destype=cache&userid=username/passwd@database'||
        '&paramform=no';

        web.show_document(vc_url,'_blank');
```

where vc_url is defined as a varchar2(200). This code would obviously be placed immediately after the ELSIF referring to the FILE_TYPES report. Either method works without problems.

If the jsp file has been generated and there is the html code enabled for the when display then this can be invoked with similar code. In this case the code would look like the following:

```
vc_url :=
'/reports/STORES/stores_detail.jsp?server=rep_macro_s1_frhome1&desformat=htmlcss'||
        '&destype=cache&userid=username/passwd@database&paramform=no';
```

The call to web_show_document remains the same. Of course if there is no html for the web display in the jsp file a blank page is generated.

The first method is used since it has an advantage in hiding all the calling parameters to the report and ensuring security of user identifiers and passwords etc.

NOTE: When calling the run_report_object_proc the reports_rwservlet there are a couple of situations that may need to be considered. For example it may be desired to initially run reports against the local OC4J instance and later make the call to the Report Server itself. The run_report_object_proc allows this by the report_rwservlet parameter.

If the Forms and Reports are on the same host which will be the situation with the Development Suite and the desire to call the reports as they are developed the parameter should be set to:

```
        reports_rwservlet_'/reports/rwservlet'
```

If however there is the desire to run the reports against the Application server then the parameter should be set to:

```
reports_servlet:='http://server:7777/reports/rwservlet'
```

since the reports server is a separate host. The example is completed above with the name of the application server.

An example of code using a test to determine the calling parameter settings is provided below for reference.

```
PROCEDURE call_report IS
  report_id Report_Object;
  v_report_other VARCHAR2(4000) := '';
  v_rep_server VARCHAR2(32) := :parameter.RP2RROREPORTSERVER;
  -- vc_url varchar2(200);
  v_name VARCHAR2(72);
  v_val varchar2(72);
BEGIN
   report_id := find_report_object('RP2RRO');
   v_report_other := 'paramform=yes';

   tool_env.getvar('COMPUTERNAME', v_name);

   message('Running on Computer '||v_name||' Calling report server '||v_rep_server);
   -- Report Server name will be like rep_host_home or rep_host
   -- so strip off first 4 chars and any chars after last '_'  [small problem if there is a _ in
the name.
   -- Also convert to upper case since the computer name in the environment will be upper
case.
   v_val := substr(v_rep_server,5);
   v_val := upper(substr(v_val,1,instr(v_val,'_',-1)-1));

   IF v_val = 'SERVER' and v_name != 'SERVER' THEN
          run_report_object_proc(report_id,
v_rep_server,'pdf',CACHE,'STORES/STORES_detail.rdf',
                    v_report_other,'http://server.macrotone.local:7777/reports/rwservlet');
   ELSE
          run_report_object_proc(report_id,
v_rep_server,'pdf',CACHE,'STORES/STORES_detail.rdf',
                    v_report_other,'/reports/rwservlet');
   END IF;
END;
```

The above assumes of course that there is no Forms/Reports server installed on the same machine as the Development Suite.

4.1.1 Reports and Directory Structure on Development Suite

If using the directory structure as set up on the Application Server, it is convenient (and probably best) for the Development Suite OC4J reports instance to also use the same structure. Therefore the OC4J source directory needs to be modified.

Within the <ORACLE_HOME>/reports/conf directory there will be a file name rep_*host*.conf where the *host* is the name of the local development PC. Open this file and near the top there will be a line like the following:

```
<!--property name="sourceDir" value="your reports source directory"/-->
```

Change this line to something appropriate. For example on the authors' machine the line was changed to:

```
<property name="sourceDir" value="D:\Projects\Macrotone"/>
```

Where the application directories were located below the D:\projects\Macrotone directory.

4.1.2 Common Errors calling reports from form

Occasionally problems have been seen when trying to run a report from a form button on the Development Suite. These problems seem quite strange in that they do not seem to product consistent results. Investigations do not reveal a single cause for the problem, but rather a multitude of small inconsistencies.

4.1.2.1 Cannot find module psoasyn in library orapls10.dll

This is cause by the environment settings upon the PC. Often changing the order of the PATH variables for the Oracle Homes such that the development suite home is first in the list will resolve this issue. Using the Oracle Installer is the easiest method to use to change the PATH order.

4.1.2.2 Problem calling run_report_object using parameters

If using the written run_report_object procedure described above, problems are encountered using the v_report_other parameter to add other fixed parameters this is most often caused by the way in which the parameters are supplied.

For example the following will work:

```
v_report_other := 'parameterform=no p_value=66';
```

But the following will fail:

```
v_report_other := 'parameterform=no&p_value=66';
```

It is the additional '&' (ambersand) that is causing the problem. The confusion is caused by the need to use an '&' in calls directly to the html if invoking the form or report form the browser window directly.

Within the procedure run_report_object the parameter gets processed prior to calling the report itself and the '&' is believed to be confusing the 'set_report_object_property' routine in some way.

4.1.2.3 Cannot contact report server issues

Problems have been seen where the OC4J instance is started on the Development Suite [either form the forms or reports options, it doesn't matter which], when the report is called form a form.

This problem is thought to be caused by contention upon the PC, particularly if the PC is working hard and low on memory. For example on a PC equipped with 512Mb of main memory this is seen more often than on a PC with more memory.

One trick to get around or test this is to start up a new browser session and enter the following URL:

http://localhost:8889/reports/rwservlet/getserverinfo?

This will display a page of information on the reports server itself if it is running.

Now run the form again and try and launch the report. In most cases this will run the report successfully (unless there are other problems of course).

4.1.3 Calling Reports from a menu bar

This can use the same methods as above, but some extra caveats apply. The name of the report server needs to be passed into the calling routines and because the menu does not have any parameters, will need to be set in each menu or in a called routine within the form. It can not be null.

4.1.4 Testing Menu called reports.

There is a small feature that results in the inability to test reports calling from menu items from within the Forms Builder. The immediate result is that the menu items do not get invoked. If the form is compiled and then executed via the usual browser on the Forms Middle tier application server the reports called from the menu items work perfectly well. The advice is to check out the menu calls from the middle tier rather than assuming that because they do not work in the Development Suite Forms Builder they are wrong.

The error most often presents itself as a 'FRM-21011: PL/SQL unhandled exception ORA-06508';

4.2 Closing the Forms main page on exit

Oracle left out a feature for Web forms, one that will close the browser window for you. Currently, if you just exit the main form, you're left with a dull-coloured applet window. This is not that useful, so we make use of a post form trigger and ensure there is an 'Exit' button on the main form. Add the following code to a post-form trigger:

```
web.show_document('html/close.html','_self');
```

In the ORACLE_HOME/tools/web/html directory on the Application Server machine, create a file called close.html and add this to the file:

```
<html>
  <body onload="self.focus(); closeWindow()">
    <SCRIPT LANGUAGE="JavaScript">
      function closeWindow()
      {
      //Firefox
        if (navigator.appName == 'Netscape')
        {
          window.open('','_self','');
          window.close();
        }
        else //Internet Explorer
        {
          win = top;
          win.opener = top;
          win.close ();
        }
      }
    </SCRIPT>
  </body>
</html
```

This works since the code will call a form called http://*host:port*/forms/html/close.html which used the alias 'forms/html' which points to the correct directory. This will also work with the Development suite.

In Mozilla (and Firefox) by default the "dom.allow_scripts_to_close_windows" value which controls the Java Script close window is set to "false". In order to fix this issue change this value to "true"

1. Location of the File:

    i.e. C:\Program Files\Mozilla Firefox\greprefs\all.js

change "dom.allow_scripts_to_close_windows" from "false" to "true"

Example:

Default value:	pref("dom.allow_scripts_to_close_windows", false);
Change it to:	pref("dom.allow_scripts_to_close_windows", true);

2.Close Mozilla browser

3.Try accessing HTML page which has window.close() code snippet.

If it is the last tab that is being closed the browser will close as with Internet Explorer.

This works with Firefox 2, but Internet Explorer 7 has yet to be tested.

4.3 A Macrotone specific JAR file.

There are some definite benefits to having a Macrotone specific JAR file for forms work. These include the use of standard company icons and reusable java code that can be applied across all of the forms development. To achieve this end the sample application icons described in this report have taken the approach of a single Macrotone.jar file containing both iconic images and also some simple java components.

To demonstrate and provide some additional functionality to Macrotone forms applications some of the Oracle supplied code as using in the 'Forms Demos' has been has been modified and placed in the macrotone.jar file. The specific PJC available within the macrotone.jar file are documented in section 8.2. Some of the build details are provided below for the RoundedButton and the RollOverButton features and the user is advised to look at the library code to build upon what is currently there and thus provide the Macrotone users with a 'richer' usage environment.

4.3.1 Using Iconic Images in a JAR file in Forms/Reports

When running an Oracle Forms 10g application the icon files used must be in a web enabled format such as JPG or GIF (GIF is the default format) (This is unlike older versions of forms running in client-server mode when the file format is .ico.) See also Metalink Note 232413.1

Icon image files can either be retrieved by Forms as individual files on the file system or from a Java Archive (JAR file). If an application uses lots of icon images it is recommended that they are stored in a JAR file to reduce the number of HTTP round trips. Note that some modifications are likely to be required on the icons after they are converted to GIF (or JPG) format so that they display 'nicely' within the form. Resizing is the most common modification required. For the sample applications described in this report, all the icons have been combined into a single jar file usable by all three applications. Where duplication was encountered suitable renaming (and changing of the icon display name in the forms) was made.

Steps to achieve this in both the 10gDS and 10g Application server environments are given below:

4.3.1.1 Oracle 10gDS (Forms Builder / Runtime)

This example assumes an install of 10gDS on a MS Windows machine.

1] Copy all the icons files (gif or jpg) to a folder e.g. c:\icons folder

2] Within a MS Dos/ Command prompt window

Change to the target icons folder and jar the icon files

```
cd c:\icons
```

```
jar -cvf icons.jar *
```

3] Copy icons.jar to <oracle10gDS_home>\forms\java

4] Changes have to be made in formsweb.cfg [ORACLE_HOME/forms/server]

a) archive_jini=xxall_jinit.jar,icons.jar

b) imageBase=codeBase

This signifies that the jar file is placed under the forms/java folder. (Be careful about case as java is case sensitive)

5] As a test create a simple form in Forms Builder

a. File -> New Form

b. Add a form level ON-LOGON trigger, syntax null;

(For a simple test no need for a database connection)

b. Create a canvas, add a button(s)

c. In Object Navigator - highlight button

d. Right mouse click and bring up property palette

e. Change properties as follows:

Iconic= yes

Icon Filename = <icon filename which exists in jar file>.<gif or jpg> (Again be careful that the case matches the filename and extension)

6] Run this form - the button(s) should appear with icon showing

No changes are necessary in the orion-web.xml.

Changes to this file would only be necessary if deploying images as separate gif, jpeg files not stored in a JAR file.

<u>Additional Notes:</u>

The following method could also be used to store the icon files in a JAR file

```
cd c:\
```

```
jar -cvf icons.jar icons
```

This will store the icons under a directory or folder called /icons within the actual jar file. An extra change would therefore need to be made to the registry.dat

```
Open <10gDS_home>\forms\java\oracle\forms\registry\registry.dat
```

In a wordpad/notepad Edit the following line:

```
default.icons.iconpath=icons
```

The string 'icons' has to be specified here because the icons to be found in the \icons directory of the jar file. Consequently it is necessary to tell Forms Runtime that the search path will be:

```
http://<host_machinename>/codebase/icons
```

4.3.1.2 Oracle Forms Services

This example assumes an install of Oracle AS 10g R2

On UNIX the instructions are almost identical to deploying iconic images in the 10gDs (Forms Builder / Runtime) environment. Telnet and log in to the target box.

1] Copy all the icons files (gif or jpg) to a directory

```
e.g. /myapplication/icons
```

2] Change to the target icons folder and jar the icon files

```
cd /myapplication/icons

jar -cvf icons.jar *
```

3] Copy icons.jar to <oracle10gAS_middletier_home>/forms/java

4] Changes to be made in the formsweb.cfg (e.g use vi editor)

```
a) archive_jini=XXall_jinit.jar,icons.jar

b) imageBase=codeBase
```

This signifies that the jar file is placed under the forms/java folder. (Be careful about case as java / UNIX is case sensitive)

5] Copy / ftp the simple form (fmb) e.g. as created in Oracle DS steps to the target box.

6] Generate the fmx file

7] Run this form e.g. the button(s) should appear with icon showing

4.3.2 Using Macrotone specific JAVA code in Forms

The logic extension to using Macrotone specific Icons in a form is to use Macrotone specific java code within a Form. Designing a Forms application to run in a three-tier environment using WebForms, allows one to extend and customize an application's functionality using one of the following methods:

- Link JavaBeans into the design, which can then be called from the application at runtime.
- Replace standard user-interface components in the form with customized versions, or add specialized user-interface components.

These JavaBeans and customized components use a special Forms-defined interface, called the IView interface. They also use certain properties, built-ins, triggers, listeners, and system variables for communicating with the Forms application.

At runtime, Webforms communicates between the client applet and server using a model-view-controller (MVC) scheme, in which its user interface handlers use its IHandler interface to invoke Forms components (implemented as views) that use its IView interface. One can use this same IView interface for the companies own beans and components to integrate them into Macrotone Forms application.

Areas that can be modified include:

- The Item Type property, to which a item type called Bean Area has been added.
- A property, called Implementation Class property, which connects an item type to the Java component through its class name. [This is used as an example below.]
- A new built-in, called Set_Custom_Item_Property, for communicating with the Java component.
- The When-Custom-Item-Event trigger, for activating the Java component.

Using a modified example from the Oracle supplied demos the following section makes use of java code to generate a rounded button look for forms.

The java code for the RoundedButton class is as follows:

```
/* Copyright (c) Oracle Corporation 1998.  All Rights Reserved. */

package oracle.forms.macrotone;

import oracle.forms.ui.VButton;
import oracle.forms.handler.IHandler;

import java.awt.*;

/**
 * Class that implements a button that decides whether or not it needs to
 * be rounded by searching for buttons of the same class with the same parent.
 * The button assumes that buttons with the same y position belong to the
 * same group if the gap between the buttons is less than sMaxGap.
 */
public class RoundedButton extends VButton
{
   // Apps coding standards require a 0.1" gap between buttons in a group and
   // 0.5" between groups.  24 pixels works for every reasonable DPI.

   private static int sMaxGap = 24;

   public RoundedButton()
   {
      super();
   }

   public void setVisible(boolean vis)
   {
      super.setVisible(vis);
      roundButtons();
   }
   /**
    * Find all the buttons at the same height as this one, work out how they
    * are grouped together and then go through and round the buttons at either
    * end of each group.
    */
   private void roundButtons()
   {
      Container parent = getParent();
```

```
if ( parent != null )
{
   Component comp[] = parent.getComponents();
   int i, j, n, start, end;
   Point pos = getLocation();

   // Find all visible RoundedButtons with the same y position as this
   // one.

   for ( n = 0, i = 0; i < comp.length; i++ )
   {
      if ( comp[i] instanceof RoundedButton &&
           pos.y == comp[i].getLocation().y &&
           comp[i].isVisible() )
      {
         comp[n++] = comp[i];
      }
   }

   // Sort the components according to their x position, using an
   // insertion sort.  This should be fast enough for any reasonable
   // number of buttons.

   for ( i = 0; i < n; i++ )
   {
      for ( j = i + 1; j < n; j++ )
      {
         if ( comp[i].getLocation().x > comp[j].getLocation().x )
         {
            Component temp = comp[i];
            comp[i] = comp[j];
            comp[j] = temp;
         }
      }
   }

   // Go through the sorted list of buttons, finding each group and
   // rounding the end buttons.

   for ( start = 0; start < n; start = end + 1 )
   {
      for ( end = start; end < n - 1; end++ )
      {
         Rectangle rect = comp[end].getBounds();
         pos = comp[end + 1].getLocation();

         if ( pos.x - (rect.x + rect.width) > sMaxGap )
            break;
      }

      // We have a group of buttons, remove all rounding and then
      // round the end buttons.

      for ( j = start; j <= end; j++ )
      {
         ((RoundedButton) comp[j]).setLeftmost(false);
         ((RoundedButton) comp[j]).setRightmost(false);
      }

      ((RoundedButton) comp[start]).setLeftmost(true);
      ((RoundedButton) comp[end]).setRightmost(true);
   }
```

```
      }
   }
}
```

The only change made in the above code was to rename the package name itself to incorporate the macrotone name. The code has to be placed in the same directory as the ICON/oracle/forms/macrotone directory. The code is compiled using the javac compiler.

It may be necessary to ensure that the jar.exec and javac.exe executables are in the search path. Also to compile the Java file the CLASSPATH variable needs to be set. For the test machine the CLASSPATH was set as follows:

```
Set CLASSPATH=D:\oracle\product\DS10102\forms\java\frmall.jar
```

Then to compile the code issue the command:

```
javac RoundedButton.java
```

Once the class file is generated it can be added to the jar file.

To ease the number of jar files that are Macrotone specific it was decided to use the same file for the icons and the java class files. This is not mandatory but reduces the number of jar files to be specified.

```
jar cvf Macrotone.jar oracle\forms\macrotone\*.class *.gif
```

Once the macrotone.jar file is generated, copy (or place it in the ORACHE_HOME\forms\java directory.

Ensure that the formsweb.cfg file has the appropriate entry for the macrotone.jar file in the archive_jini parameter.

To make this Java class available to the Forms applet, it must be accessible on the virtual CODEBASE mapping that is defined in the HTML file. The CODEBASE mapping for Developer Server application deployment should point to <oracle_home>/forms/java. This will be the default.

This means that when the RoundedButton class is loaded, the Java Virtual machine contained within the appletviewer or JINITIATOR uses the CODEBASE mapping <oracle_home>/forms/java as the top level directory and then looks in the directories that correspond to the package name, oracle/forms/macrotone.

All that remains is to test the file. Place the implementation class on the form item to be 'oracle.forms.macrotone.RoundedButton' and recompile and run the form. The new class file will apply to the button items marked with the implementation flag.

4.3.2.1 Pluggable Java Components PJC

Some mixed success has been encountered trying to use demo pluggable java components. For example the calendar example (different from the example described above) worked very easily, but the 'rolloverbutton' example gave a few issues.

Of particular note is the RollOverButton code which does not quite match its documentation. It seems that the Button needs to be 'NON ICONIC' and the '[ROLLOVER]' naming needs to be applied to the normal label. The display will then be iconic in form. This a little misleading but is my excuse for why it took me a while to work it all out.

4.3.3 Putting it all together

Take the macrotone.jar file and place it into the ORACLE_HOME\forms\java directory.

Then edit the ORACLE_HOME\forms\server\formsweb.cfgfile to add the macrotone.jar file name to the end of the archive_jini parameter. A restart of the OC4J instance on the Development Machines is then required, or the Application server forms service to enable the pick up of the new library.

To make use the icons all that is required is to put the icon name (which incidentally is a gif file) as the icon file name in your form. It is not necessary to specify the '.gif' suffix.

To make use of the 2 JPC (Java pluggable components) all that is required is to specify the name of the required component as the java implementation call on the button property sheet.

i.e. To use the roundedbutton class just set 'oracle.forms.macrotone.RoundedButton' as the java implementation class to the button property sheet.

Similarly set the implementation class to 'oracle.formsmacrotone.RolloverButton' and specify the label as '[ROLLOVER]exit' etc on the button property sheet. In essence this references two icons in the macrotone,jar file called exit_on.gif and exit_off.gif. There are icons for the obvious names such as prvrec, nxtblk etc in the macrotone.jar file. More can be added as required.

5 Configuration

This section describes the configuration tasks carried out on the Middle Tier Forms/Reports installation. This configuration is virtually the same if a full installation of Middle Tier and Infrastructure is performed, so this section effectively applies to both. Where differences are significant they are marked as being so.

All configuration files can be changes either through the Enterprise Manager Console or directly by editing the files with a text editor. Using the Console is generally easier and the emphasis will therefore be upon this method.

Access to the Console is achieved via the browser by using the address URL:

http://host:port/emd/console

Where the host name is the name of the machine upon the network, and the port number is the port of the Enterprise Manager itself. If this port is not known, then it can be obtained by looking in the ORCALE_HOME/install directory for a file called portslist.txt, which will list all the port numbers used by that particular` installation of the software. [This file exists for virtually all Oracle installation and can be checked for allocations. Any changes to the port configuration after installation will NOT be reflected in this file, unless manually changed.]

5.1 Forms

Setting up a new application:

From the Console Home, navigate to the Forms -> Configuration page. A number of Forms Configurations are provided. These can all be either changed for ones own purpose, or more usually used as a template for the particular installation that is to be performed. The following example uses the TEST application as a guide. The 'default' configuration is 'duplicated. The new name, i.e. TEST is chosen as being suitable. Clicking into the Description area for the 'new;' TEST configuration enables one to change the description to match that desired to accurately describe the configuration. i.e 'Macrotone TEST Configuration'.

Now select the 'TEST' application and click the Edit button. A new page will be displayed enabling the individual setting to be changed. The primary ones to change are the main called for: i.e. form = test_main.fmx. [Note the chosen form can be specified as a '.frm' or a.'.fmx' form. Either will work successfully.] The userid can be specified if it is desired that the initial form be displayed without requiring the user to login in. The 'pageTitle' presents the details specified at the head of the browser and in this example it was made 'Macrotone TEST Application'. It was also chosen to change the default window size to 'width = 950' and 'height=600'.

Since we chose to put all the Macrotone icons into a single jar file, it was also necessary to specify the location of this file, and this was achieved by adding the name of the icon jar file to the 'archive_jini' parameter. i.e. archive_jini = frmall_jinit.jar,macrotone_icons.jar.

The final parameter chosen to be changed in this section was the 'clientDPI' setting which was added as a new parameter and was set to a value of '96'. This parameter affects the screen scaling and enabled the page display to more easily fit onto the screen.

Click on the 'Apply' button to save the changes.

The next step is to pick the Forms -> Environment option. By default there will only be a single environment file called 'default.env'. Others can be created if desired and to use these, and extra parameter would be required when the form is invoked. In the environment being used this is a little overkill so the main default.env file is chosen to be edited. Therefore click on the Edit button and the user is presented with a new screen showing the parameters. Most of these are correctly set as expected, but we desire to set up our forms by a specific application directory, so we have to add the path of where the forms are located to the 'FORMS_PATH' variable. Click in the Value box, next o the variable and add the path to the end of the existing values. i.e. 'C:\oracle\products\FRHome_1\forms;D:\MACROTONE\TEST'. NO other parameters need to be changed, so click on the 'Apply' button to save the changes.

Note that the FORMS_PATH will get extended with every new 'form application' installed in this way.

The forms are now configured.

5.1.1 Configuring jar files to be used within forms

If it is desired to make use of a jar file within a forms application the jar file needs to be configured. i.e. The jar file may contain a set of java class files to modify the shape of the buttons within the form.

1. Place the jar file into the forms90\java folder.
2. Add an entry for the jar file to the archive_jini parameter of the formsweb.cfg. i.e. Assuming we are adding a jar file called hyperlink.jar the entry would become: archive_jini=f90all_jinit.jar,hyperlink.jar.
3. If this is an OC4j environment such as the Forms developer, restart the OC4j instance and the Forms Builder.

5.2 Reports

The reports are not configured in the same way as the forms described above. The reports are most easily configured using the existing configuration, the chosen method described here.

Within the ORACLE_HOME/j2ee/OC4J_BI_Forms/applications/reports/web directory reports located here will be picked up automatically without any further action required. So a new directory was created called 'TEST' reports are located within that directory. To invoke the report all that is required is to specify the directory name immediately before the report name. i.e. report=TEST/Store_List.rdf.

5.3 WebUtil Configuration

WebUtil is supplied with the Development suite but not as part of the Forms/Reports stand alone middle tier or as part of the full application suite. There are few extra configuration items that need to be completed.

1. Download http://prdownloads.sourceforge.net/jacob-project/jacob_18.zip. This archive supplies the core OLE functionality provided by WebUtil.
2. From the JACOB Zip file, extract both jacob.dll and jacob.jar into the ORACLE_HOME\forms\WebUtil and ORACLE_HOME\forms\java directories, respectively.

3. You need to sign both frmwebutil.jar and jacob.jar with the same digital certificate. This is a one-time operation which allows your end-users to trust that the JACOB routines can access client side resources. If you do not have an existing signing certificate, or if you are not sure how to sign Jar files, a script is in the `forms\WebUtil` directory to help you. This script is called sign_webutil.sh for UNIX and sign_webutil.bat for Windows.

   To sign the Jar files:

4. Open a Command window and change to the ORACLE_HOME\forms\webutil directory.
5. Check that ORACLE_HOME/jdk/bin is in the path. If it is not, add it.
6. In Windows, Issue sign_webutil.bat ORACLE_HOME\forms\java\frmwebutil.jar (or the path to where you installed WebUtil). On UNIX, issue sign_webutil.sh ORACLE_HOME/forms/java/frmwebutil.`jar` (or the path to where you installed WebUtil).
7. In Windows, Issue sign_webutil.bat ORACLE_HOME\forms\webutil\jacob.jar. In UNIX, issue sign_webutil.sh $ORACLE_HOME>\forms\java\jacob.jar.

   This only needs to be done once since the certificate will be stored in the JInitiator keystore.

8. Because in this release the JACOB code is in an external Jar file and not incorporated into frmwebutil.jar, it needs to be downloaded. To do this, change the WebUtilArchive setting to read: WebUtilArchive=frmwebutil.jar,jacob.jar

Note: The latest version of Jacob is release 1.11 which unfortunately does not appear to work.

One additional task to be performed is that the webutil binary needs to be recompiled against an installation of the database objects. In our situation the webutil package had been installed in the 'webutil' schema upon the database so this was used to compile the pll library form within the Forms Builder environment.

Within the Development suite it is still necessary to obtain the Jacob distribution since it is not possible for Oracle to distribute it with the release of Developer Suite. Thus some of the steps described above still need to be carried out to enable webutil based application to be developed successfully.

5.4 WebUtil Demo

It is possible to obtain a sample form that demonstrates the use of the WebUtil code within a 'real' form. The demo comes complete with its own installation instructions and it is not necessary to repeat them here. The reader is referred to those documented steps. It is important to mention that the demo has been successfully used on the Development Suite installation and it does indeed work as expected. The interested user is encouraged to try it themselves, particularly if the forms they intend to migrate contain functionality that interacts with the client.

5.5 Forms 10G demos

It is worth mentioning that there are some distinct problems with the supplied code for the 10G forms demos. The compiled executable will work however the code supplied is of 9G and NOT 10G. This means that the code will not work without extensive changes. It is sufficient for viewing techniques only.

Typical problems encountered include:

- Sub classed Object libraries that are not connected within the Forms Builder. This means that a number of programmable objects will not be found such as 'SHOWHELP', MSGBOX etc. This is basically a problem resolving the Object library. See section 11.2 for more details.
- Hard coded paths to the 9G path 'forms9demos'.
- Buttons that are positions at location 0,0 with a size 1,1 which is effectively non displayable.
- Buttons that are defined as text areas.
- Canvas's not linked to a base window and visa versa.
- Displayed items with a height of 1.
- Console Window not specified on form property

Solutions for the sub classed objects basically means 'copying' the object libraries objects and then modifying the code.

All the code can be made to work (or at least all those I tried which was a representative sample) but a lot of effort would be expended upon the way. Of course it is always a reasonable learning opportunity.

6 Developer PC configuration

The speed of the PC processor and the amount of memory available will directly affect the responsiveness of the Developer PC particularly with the use of the browser. A minimum of 512M of memory is required and preferably 1GB or more.

Each developer will require a full installation of the Oracle Developer Suite 10G Release 2. These are currently upon a DVD which will need copying to the PC prior to unzipping. Once installed this needs to be patched with the latest patch set which currently will bring it up to release 10.1.2.2.0. This patch set is also upon a DVD and will also require copying and unzipping prior to installation.

Some configuration changes will then need to be performed. These are described in specific detail in other parts of this document but are listed here as a checklist.

Set up the tnsnames.ora entries as appropriate and/or modify the Windows registry to add TNS_ADMIN entries as required for the Oracle Homes.

Copy the latest version of the macrotone.jar file to the forms/java directory.

Edit the forms/server/formsweb.cfg file to add the name of the macrotone.jar file to the archive_jini variable.

If it is desired to mimic the reports and forms configuration as used on the apps server on the PC, then edit the Reports server configuration file parameter 'sourcedir' to point to the location of the Reports. At this time it is useful to set up the PC with the same directory structure as well.

Modify the browser settings to ensure that all local domain resolution for Macrotone.org and macrotone.local nodes do not go out onto the outside web for resolution.

If necessary modify the browser setting to allow active x applications to run.

Ensure no web browser control software is preventing access to the machine nodes to be used. For each application converted create appropriate sections within the forms/server/formsweb.cfg file for the application. Use these for testing prior to final deployment to the apps server.

It is useful to set up a local directory for the apps server accessible as a drive upon the PC. This will then be used to copy the completed forms and reports to the apps server. A user name and password will be required to enable write access, which will be required by the developers.

After an application has been migrated, the Apps server application needs to be created. This can be done by the Oracle DBA, or a competent developer. See elsewhere in this document for details. In addition the APPLAUNCH form will need to be modified to add details of the application.

7 Problems encountered

7.1 Web Browsers

7.1.1 IE7 and Firefox

There are a few issues related to the use of the Forms and Reports web pages reliably with the Firefox (and probably Internet Explorer 7) browser.

This also has important implications for the future release of Internet Explorer 7. Microsoft has pushed out Internet Explorer 7 (IE 7) as a 'High Priority' update through its 'Automatic Updates Patch Distribution Service' for Windows XP users later this year. This may have ramifications for Oracle Applications users, who may unknowingly have their IE 6 installation upgraded to IE 7 before it is fully certified by Oracle.

Microsoft have released a blocker toolkit that may be installed by administrators to prevent the automatic delivery of IE 7 to their users desktops. Although the application of the blocker software prevents the automatic download of IE 7, it does not stop users manually downloading and installing it themselves. It would be a good idea for System Administrators to advise their users not to download IE 7 until it has been certified by Oracle.

If the Microsoft 'Automatic Updates' facility is set to 'Automatically download recommended updates for my computer and install them', it is highly recommended that one installs the blocker toolkit. If your Microsoft 'Automatic Updates' facility is set to 'Download updates for me, but let me choose when to install them', it may still be useful to install the blocker toolkit as an extra safety measure.

Companies that have their own Windows patch update server (e.g. Windows Server Update Services (WSUS) or SMS 2003) do not need this blocker. They should just manage the IE 7 update so that it is not deployed until certification of IE 7 has been announced by Oracle.

Until such time as Internet Explorer 7 has been validated for the Application Server by Oracle caution is advisable.

One particular observation concerns the opening of new windows within tabs option. It seems that when the first report is invoked from a Forms window a new tab is opened. Subsequent reports will always open in the same tab, even if the tab itself has been used subsequently to view some other web page (or site.)

This can be inconvenient and mildly annoying sometimes. Moving of the tabs to different positions doesn't seem to make much change.

7.2 http://%%20 when running from Forms Builder

When trying to launch the run form from the Forms Builder the page displayed shows the URL http://%%20 at the start of the URL. The exact cause of the problem is not known, but setting the preference for the browser seems to resolve the problem. This has only been seen on one PC.

Workaround 1

1. Exit the Builder and OC4J
2. Open cauprefs.ora (located in the Oracle Home root directory)

3. Locate the lines labelled "Forms.html ..."
4. Make your changes here. Any values you do not want displayed in the Builder replace with (). Example below.
5. Save the file and restart the Builder

Example of a correct cauprefs.ora entry

```
Forms.html = "http://myMachine:8889/forms/frmservlet"
Forms.html1 = ()
Forms.html2 = ()
Forms.html3 = ()
Forms.html4 = ()
Forms.html5 = ()
```

The machine.domain name should NOT contain spaces. This is poor naming practice regardless of the use of Forms.

Workaround 2

Go to the preferences dialog box and click on the Runtime tab. Set Web Browser (i.e, C:\Program Files\Internet Explorer\IEXPLORE.EXE). To make the change within the Forms Builder, click on Edit, Preferences, Runtime and then set Application Server URL, and the Web Browser Location. Both are required.

Workaround 3

Go to the preferences dialog box and click on the Runtime tab. Make sure your Application Server field is correct.

Workaround 4

Go to the preferences dialog box and click on the Runtime tab and then make Application Server field Blank.

Workaround 5

%20 is the escape for blank. Maybe there is a blank character in the runtime settings.

7.3 Running Run Form from Browser URL: HTTP://127.0.0.1:1276/xqSvRNEFF5n2k4g20EfiQ0rNzLbD7zKNhrSqw4y98gwef8t3

The application server URL is correctly set to 'http://machine:port/forms/frmservlet' but when run the strange looking URL is displayed. This may or may not occur with the earlier URL http://%%20. This has only been seen on one PC.

The problem will probably not be seen with the Mozilla or Firefox browsers.

The probably causes are that the browser security is set higher than the default level of medium, or the registry entry FORMS_HIDE_OBR_PARAMS needs to be changed.

There are several ways to work around the problem. In some cases, it may be necessary to implement all.

OPTION 1

Assuming you are using Internet Explorer, do the following in the browser Preferences:

Tools > Internet Options > Advanced

In the section titled "Security", CHECK the box labelled "Allow active content to run in files on My Computer"

Also, it is a good idea to CHECK the box labelled "Empty Temporary Internet Files folder when browser is closed"

Accept the changes and retest. It may be necessary to exit and reopen the browser.

OPTION 2

Add a Windows registry entry to the iDS environment to disable the function that creates the temp files and causes the problem. Note that this will cause the complete URL to be displayed in the browser when running a form from the Builder and will display username and password information in the URL.

FORMS90_HIDE_OBR_PARAMS = 0

For Forms 10.1.2.x, the registry key name will be:

FORMS_HIDE_OBR_PARAMS

OPTION 3

Assuming the use of Internet Explorer, do the following in the browser Preferences:

Tools > Internet Options > Security

And reset the security level for the zone. Accept the changes and exit the browser and then reopen it.

7.4 Drop-down menus fail to function correctly.

This problem was first seen after IE7 was installed upon the test system. Detailed investigation revealed that this is a Windows problem rather than an Oracle Forms/Reports type problem.

To resolve please ensure the following:

- Make sure popup blockers are off.
- If Customer has two monitors, drop-down menus will only work on the primary monitor.

It is the encountered situation it was the second situation that was the main cause upon the test machine.

Rechecking of the use of the Firefox browser using this solution also works. [Note that the automatic closing of the browser window by using the close.html script as described in section 4.2 does not close the Firefox window upon exit from the top form. This will be investigated further when circumstances permit.]

Retesting of Internet Explorer 7 has revealed that the popup blocker was indeed the cause of the original problem. Preliminary tests seem to show no show stoppers but some further exposure is required before a strong recommendation can be given.

7.4.1 Oracle DBA Toolbar and Popup Blockers

An associated problem involved the use of a 'Oracle DBA Toolbar' obtainable from Oracle Technical Network. This toolbar effectively prevented the spawning of additional browser windows due to an inbuilt pop up blocker. Unfortunately this is not apparent on the web site and some time was lost investigating the 'problem'. Disability the toolbar allowed the new windows to be generated. This toolbar is unlikelyto be installed by the vast majority of users so can itself be reasonably ignored, however the use of such blockers is common, so the problem may be seen with out such products.

7.5 Printing from the Report Builder.

Printer icon versus File/Print pull-down menu

To print a report directly from ReportBuilder, one must first run the report 'on screen' then select either the File/Print pull-down menu or the Printer icon.

Either choice takes the user to the Printer dialog box.

Selecting the File/Print pull-down menu causes the report to print properly.

However, if the dialog box comes from the Printer icon, the report throws the following error messages:

REP-0069: Internal error

REP-57054: In-process job terminated. Executed successfully but there were some errors when distribute the output

REP-50159: Executed successfully but there were some errors when distribute the output

This is using ReportBuilder v10.1.2.0.2

7.6 Creating Templates for Web Layout to be used in Report Builder

Based upon Metalink 230282.1

You can use templates to define common characteristics and objects that you want to apply to multiple reports. For example, you can define a template that includes the company logo and sets fonts and colours for selected areas of a report.

7.6.1 Creating web templates

Steps to create an Oracle9i Reports Web Layout template:

1. Create a template file (.tdf file).

2. Create an HTML file that describes the layout.

3. Update the Web Template description file (rwTemplates.xml).

4. Test the new template.

5. Deploy the template in Oracle9i Reports and Oracle9i Application Server.

6. Add the template to the Template Library.

7. Modify the default Web source file.

An Oracle9i Reports template is comprised of a paper layout template (.tdf file), a Web layout template (.html file), and a CSS file.

The TDF file is used to register the template in reports and to make the relationship between the HTML and the CSS file in the rwTemplates.xml file.

It can also be used as a paper template.

The HTML file is used to create the layout of the template (default header and footer, position of the data area.)

The CSS file is used to define the HTML font of the different parts of a report.

Step 1: Create a template file. First, you must create a Reports template file:

In Oracle9i Reports Builder, choose File > New > Template.

In the Object Navigator, under Templates, your new template name displays.

Right-click the name, then choose Property Inspector from the pop up menu.

In the Name field, type a name for your template, for example "MyCompany". This name will be used by Oracle9i Reports to create the relationship between the HTML and CSS files. Press Enter to make your changes.

Save the file as MYCOMPANY.tdf in the $ORACLE_HOME/reports/templates/ directory.

You have now created a template. You can edit this template for paper reports by using the Reports wizard.

Step 2: Create an HTML file that describes the layout

In an HTML editor of your own choice, (i.e. Macromedia Dreamweaver, Microsoft FrontPage etc.), create an HTML page that will define the areas of your Web layout template.

A. Add the JSP header and taglib, then add the rw:report tag to reference the report environment:

```
<%@ taglib uri="/WEB-INF/lib/reports_tld.jar" prefix="rw" %>

<%@ page language="java" import="java.io.*" errorPage="/rwerror.jsp"
session="false" %>

<%@ page contentType="text/html;charset=ISO-8859-1" %>

<!--

<rw:report id="report">

<rw:objects id="objects">

</rw:objects>

-->
```

B. Put all your generic HTML here:

```
<html>

<head>
```

C. This tag is used to create the link between the template and the CSS (Cascading Style Sheet) that we created and references into the rwTemplates.xml file:

```
<rw:style id="pageStyle">

</rw:style>
```

D. Put the entire generic HTML here, such as the Logo of the company:

```
<title> My Company Report </title>

<meta content="text/html; charset=iso-8859-1" http-equiv=Content-Type>
```

```
<body>

<h1>My Company Report</h1>
```

E. Put the <rw:dataArea> tag here, where the Wizard will insert the report block:

```
<rw:dataArea id="defaultLayout">

        <!-- Report Wizard inserts report block here! -->

</rw:dataArea>
```

F. Put the entire generic HTML here, such as copyright and contact information:

```
</body>

</html>
```

G. Close the report reference tag:

```
<!--

</rw:report>

-->
```

H. Save this document in the $ORACLE_HOME/reports/templates/ directory.

Step 3: Update the Web layout template description file

The Web layout template description file

($ORACLE_HOME/reports/templates/rwTemplates.xml) references Oracle9i Reports templates with their associated styles.

Add a new entry to register your new template.

The webTemplate tags associates the paper layout template, Web layout template and CSS files.

For our MyCompany sample, we can add a line like:

```
<webTemplate id="myCompany" cssFile="rwbeige.css" classSet="portletclassSet"
htmlFile="myCompany.html"/>
```

where:

id references the name of the module in the Reports template file

(mycompany.tdf). This is not the name of the file, but the name that you see in the Object Navigator.

cssFile references the name of the CSS file that Report Wizard will place in the <LINK> tag.

classSet references the classSet id into the current XML file. The classSet makes the relationship between each field type and the CSS Class.

htmlFile references the file name of the HTML template, in our example myCompany.html.

If there is a specific CSS file with new class names, then add a new classSet entry in the file.

The classSet tag contains the class name (css) for each field type.

```
<classSet id="portletclassSet">
  <class name="TableStyle" value="PortletTable"/>
  <class name="ColumnHeader" value="PortletHeading1"/>
  <class name="RowHeader" value="PortletHeading1"/>
  <class name="CellText" value="PortletText1"/>
  <class name="CellNumber" value="PortletText1"/>
  <class name="CellDate" value="PortletText1"/>
  <class name="TotalText" value="PortletText1"/>
  <class name="TotalNumber" value="PortletText1"/>
  <class name="GroupAboveHeader" value="PortletHeading2"/>
</classSet>
```

Each XML element class, contains 2 attributes:

- name: type of field (TableStyle, ColumnHeader, RowHeader, ...)
- value: name of the class in the css associates to the template.

There is a list of valid names:

TableStyle: specifies the class name of the <table> tag.

ColumnHeader: specifies the class name of the <th> tag used for the column header.

RowHeader: specifies the class name of the <th> tag used at the row header in matrix reports.

CellText: specifies the class name of the <td> tag used for alphanumeric data.

CellNumber: specifies the class name of the <td> tag used for numeric data.

CellDate: specifies the class name of the <td> tag used for date data.

TotalText: specifies the class name of the <th> tag used for the summary empty cells.

TotalNumber: specifies the class name of the <td> tag used for the summary non empty cells.

GroupAboveHeader: specifies the class name of the <th> and <caption> tags used for the group above section.

Note: If the class set is not complete, Oracle9i/10g Reports will put "(null)" as class name in the HTML document.

Save the file and restart Reports Builder.

Step 4: Test the new template

Now let's test your new template.

Create a new report using the Report Wizard, and choose the MyCompany.tdf template from the file system.

When you click Finish, you see the new report with your template code.

The Report Wizard replaces the empty <rw:style> and <rw:dataArea > tags with the generated code.

Step 5: Deploying the template

Configure Reports Builder All associate elements like images and css files must be in the correct place on the file system. The default root directory of Oracle9i Reports Builder is $ORACLE_HOME/reports/docroot/.

Default images and CSS are placed in $ORACLE_HOME/reports/docroot/images and $ORACLE_HOME/reports/docroot/css directories.

Deploy on Oracle9i Application Server To deploy a Web report on Oracle9i

Application Server, you need to make sure that all associated elements (like images, css, etc) are in the correct place on the server.

Step 6: (Optional) Add the template to the Oracle Reports template library

If you want your new template to always display on the Template page of the Report Wizard, see Contents tab > How to > Define a template > Adding a template to the predefined templates list in the Reports Builder online help.

Step 7:Modify the default Web source file

You can also modify the default Web source template. This template is used by Reports Builder when the report developer builds a Web report but does not choose a template.

This file is: $ORACLE_HOME/reports/templates/blank_template.jsp.

7.7 Destination Type Screen running Builder to Web with Parameter Form

Based on Metalink Note:254161.1

When an Oracle Reports Developer 9/10g has a parameter form has been created with the "Parameter Form Builder" and the parameter DESTYPE is present in the parameter form, an error message 'REP-56092: No class defined for destination type Screen' is encountered when running the report from report builder, using the 'Run Web Layout'

The problem is caused since when the parameter DESTYPE is selected in a parameter form built with the "Parameter Form Builder", the parameter is set in the URL used to call the "Web Layout" report causing the error.

To resolve the problem:

1. Call the "Parameter Form Builder"

2. Unselect the parameter DESTYPE

3. Answer Yes to the Warning

4. If it is required to add the parameter DESTYPE in the parameter form:

   a. Edit the parameter Form

   b. Add a field

   c. Set the source of the field to DESTYPE

Hint: It is possible to check the URL used to launch the "Web Layout" reports:

A. In the file <ORACLE_HOME_9IDS_R2>\reports\docroot\orion-conf\http-web-site.xml

Change the line:

From

```
<!--access-log path="../log/http-web-access.log" /-->
```

To

```
<access-log path="d:\temp\http-web-access.log" />
```

B. Check the file d:\temp\http-web-access.log after execution to verify the "Web Layout".

7.8 Error 404 Not Found running JSP Report in Builder

A problem was encountered when trying to run a JSP Report in the Builder, which returned a message 'Error: 404 Not Found Oraclejsp: Java.Io.Filenotfoundexception'

Evidentially this problem can occur upon any platform. The sequence of events was to a) Open a report in builder, b) Click on Run to Web Layout. The browser will now show the 404 error. Displaying paper layouts work as expected.

The cause of the problem is that there is a space in the name which is causing the URL call for the jsp not to be valid. When the URL is generated, the space is treated as the need for a new parameter.

To resolve the problem it is necessary to rename the report without a space in the name.

7.9 FRM-30457 Maximum length ignored for character-datatype subordinate mirror item

In the process of migrating some forms from release 4.5 or 6i to release 9i or 10g the error: 'FRM-30457 Maximum length ignored for character-data type subordinate mirror item' was encountered.

This is caused by the change in behaviour of synchronized items (mirror items). The warning indicates that though the subordinate item has a maximum value assigned, this value will be ignored and taken from the master item.

To resolve this issue:

1. Specify the Maximum Length property in the master mirror item.

2. In the mirrored item inherit the maximum length property by pressing the "Inherit" button on the property palette toolbar.

7.10 What are the DSA, dcm-daemon and log loader components in IAS?

What are the DSA, dcm-daemon and log loader components in IAS and why aren't they started automatically by opmnctl startall command?

The DSA is the OracleAS Guard server. The dcm-daemon is used by Oracle Application Server Control and is started when the ASC web page is accessed (i.e. ias_admin user).

The log loader component compiles log messages from various log files into a single repository.

These by default are not started with the opmnctl startall command.

The processes can be started by the "opmnctl startproc ias-component..." command if desired.

For more details on the function of these components, please refer to the Oracle® Application Server Administrator's Guide 10g Release 2 (10.1.2). See the following sections:

1.4.1 Getting Started with Oracle Process Manager and Notification Server (OPMN)

5.5.1 Starting and Stopping Log Loader

B.1.3 DCM Daemon Does Not Start As Expected

7.11 FRM-92150 Web Client Version too new

This problem was encountered after installing a patch to the Development Suite. The patch applied successfully and indicated everything was fine. When running the Forms Builder however, the error message was encountered when trying to run a form. The form itself was unchanged from before the patch application so was given a clean bill of health.

The problem is caused by the patch installing a 'new' or later version of JInitiator. In the described situation this was from version 1.3.1.17 to 1.3.1.26.

The error message implies the 'Wrong path and/or codebase setting'.

The solution is to check the `CODEBASE` entry in your HTML file or forms configuration file [$ORACLE_HOME/forms/server/formsweb.cfg] which may point to older versions of the Jar file. Then either modify the codebase entry in your configuration file or replace the jar file in the codebase path with the appropriate jar file.

If the above are both correct {in our situation they were) then clearing the Oracle Jar cache in the user profile directory of the client computer ensures that the fresh Forms Jar files are downloaded.

i.e. C:\Documents and Settings\username\Oracle Jar Cache

Where *username* is the login name of the user upon the PC.

7.12 BI TOOLS installation upon an 'old' PC with Oracle Developer 2000

An installation of the BI Tools upon a Macrotone PC already installed with Developer 2000 caused the Developer forms and reports to give TNS errors when they were invoked. The problem was complicated by the 'tnsping' utility within the Oracle Home failed to work and consistently provided messages about Message repository not found. Whilst not serious this tends to indicate a bad installation of the product. If this PC was created from an install image, it is likely that the install image was not tested sufficiently to reveal this problem. Tests showed that the BI Tools installation had obviously placed entries in the PATH variable which were causing the problem. Moving the entries to the end of the PATH variable did not resolve the problem and they had to be removed in their entirety. The solution was to create a TNS_ADMIN registry entry in the top Oracle_HOME, which happened to be the ORAWIN95 directory home.

NOTE: This problem also occurs when the 10G Developer Suite is installed on a PC with the older Developer version installed into the orawin95 directory. [Now seen on at least 2 PCs.]

Need to investigate whether putting the path entries back continues to let the application work.

7.13 Use of ON_NEW_FORM_INSTANCE triggers

A number of forms have been encountered which make use of a 'ON_NEW_FORM_INSTANCE' trigger at the data block level. This results in a number of problems with using PARAMETER lists on called forms, and also on returns from a called form to the invoking form.

Resolution has involved relocating the trigger to the form level, and introducing a 'ON_NEW_BLOCK_INSTANCE' trigger at the data block level to hold any specific block level code.

7.14 Form screen resizing

A number of issues were encountered within a number of the forms that were causing issues with the displayed output within the Web environment. Most of these issues were resolved by commenting out (or removing) the resizing calls for the form.

One specific problem was encountered within a button trigger [Invoice Form] calling another form (Stock_Movement). The form call was fine and called the form but upon return a blank screen was displayed. Only upon removal of the additional code resizing the form within the browser window after the call_form routine was the normal calling form redisplayed.

7.15 Migration assistant fails

Depending upon the configuration of the Oracle Home for the Forms environment, attempts to use the migration assistant upon forms may fail IF there are attached libraries without the full path specified. It is usual NOT to attach the full path to the libraries within the Forms themselves since this is best configured upon the Forms Middle tier itself.

7.16 Report Server Could Not Print on Default Printer with RUN_REPORT_OBJECT

The report does not get printed to the default printer set on the machine hosting Oracle Application Server while calling from Forms using RUN_REPORT_OBJECT on Windows. The DESTYPE is set to Printer while DESNAME is not set. The problem happens only when using in-process Report Server and only on Windows.

For in-process Report Server in Windows, it is not possible to get the default printer of the currently logged in user because the service that runs the in-process Report Server is logged in as Local System. Thus, HKEY_USERS/.DEFAULT gets mounted to the HKEY_CURRENT_USER for the in-process Report Server and it doesn't have any default printer. While for the standalone Report Server, the process runs as the current log on user.

7.16.1 Workaround 1

Please follow the below mentioned steps:

a. Open the Registry by running regedit.

b. Go to HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows.

c. Copy the string value for "Device". It will be like *"\\abc\sierra,winspool,Ne02:".*

d. Open the HKEY_USERS\.DEFAULT\Software\Microsoft\Windows NT\CurrentVersion\Windows. The "Device" string value will be empty. Paste the value copied from previous step.

Note: Whenever the default printer is changed on the machine hosting Oracle Application Server, this workaround has to be reapplied. Also, backup the existing registry before applying this workaround.

7.16.2 Workaround 2

Pass the default printer name of machine hosting Oracle Application Server as a value to DESNAME parameter. On a Windows host this requires the printer name as defined on the machine to be specified within double quotes. See below for more details.

7.17 Printing from Web Reports

A few issues have been discovered when trying to print directly to a printer:

- The specification for a printer name (desname) is the name of the printer as defined in the Windows Printer setup enclosed within double quotes. The double quotes are required if there are any embedded blanks in the printer name.
- When printing direct from a report with the desname specified the output is shifted up the output page slightly. All of the output is printed but it is relocated on the page differently than when printed from a browser display.
- The printer definitions need to be known OR there needs to be a method to allow the user to specify a printer.

The simplest way around these issues is to permit the report to display to the screen and then allow the user to print to a specific printer from the usual method. Not all that elegant but will work from all locations without the server needing to know all the printers.

7.18 Java crash in fontmanager.dll

The problem of a crash within the Java Machine involving the fontmanager has been seen a few times when using the Application server, although it can probably happen on any machine.

The symptoms are that the browser crashes out and a file is generated upon the users desktop with a name like 'hs_err_pid2616.log' where the pid number is a variable. The problem may or may not be reproducible. Tests seem to indicate that it often disappears.

Opening the text file reveals text like the following:

```
An unexpected exception has been detected in native code outside the VM.
Unexpected Signal : EXCEPTION_ACCESS_VIOLATION occurred at PC=0x6D202A87
Function name=Java_sun_awt_windows_DrawGlyphVectorGDI_drawGlyphVectorGDI
Library=C:\Program Files\Oracle\JInitiator 1.3.1.26\bin\fontmanager.dll

Current Java thread:
        at sun.awt.font.NativeFontWrapper.getFontMetrics(Native Method)
        at sun.awt.font.FontDesignMetrics.initMatrixAndMetrics(Unknown Source)
        at sun.awt.font.FontDesignMetrics.<init>(Unknown Source)
        at sun.java2d.SunGraphics2D.makeFontMetrics(Unknown Source)
etc…….
```

Investigation reveals that a number of different versions of Jinitiator show the problem. Several solutions are suggested which include:

Rebuild windows fonts directory.

Reportedly the problem was with the Windows font files, in C:\WINNT\Fonts, or with their permissions. The user ran the FIXACL.EXE program from the NT4.0 resource kit utilities CD and reports that this fixed the problem. On a Windows 2000 Pro (SP2 machine the Window's font database was corrupt and fixed by running the OS self-repair.

Use Sun JVM (at least 1.4.2_5) instead of Jinitiator.

Run Oracle Forms with Sun JRE. i.e. this needs to be the same as the default version used by the browser.

7.19 Configuring Forms to use Sun JVM.

Here are the changes required to make use of Sun's JVM instead of Oracle Jinitiator.

- In formsweb.cfg change baseHTML, baseHTMLjinitiator and baseHTMLie to "basejpi.htm". One could also make a copy of basejpi.htm to create a custom base file and point to that file.
- The setting of IE=native or IE=JInitiator is no longer relevant. Basically, what this does is determine if the setting for baseHTMLjinitiator (IE=JInitiator) or baseHTMLie (IE=native) should be used. If one changes both base-parameters to point to the same file, this is no longer relevant.
- The basejpi.htm uses several variables from the formsweb.cfg in the EMBED and OBJECT tags. These are: jpi_download, jpi_classid, jpi_codebase and jpi_mimetype.

Note that not all versions of Sun's JVM are certified against Oracle Forms 10G Release 2. For example, Oracle Forms 10g Release 2 (10.1.2.0.2) is supported / certified to work with Sun JDK plug-in 1.4.2 and 1.5.

There are four settings in the formsweb.cfg file that are relevant to the Sun JPI and to determine which version to use and where to download it:

- jpi_classid – The ClassID of the Sun JVM to use (Internet Explorer specific)
- jpi_codebase – Download location of the CAB file for the Sun JVM (IE specific)
- jpi_mimetype – Used primarily for Netscape/Firefox but also passed on to IE
- jpi_download_page – Download location of the JRE installer for Netscape/Firefox

There is the option for static or dynamic versioning. Static versioning mandates an exact version of Sun JPI to be installed on the workstation, whereas Dynamic versioning just states a minimum version and every higher version will also do. The use of dynamic versioning provides the greatest flexibility to users.

JPI_CLASSID should be set to clsid:8AD9C840-044E-11D1-B3E9-00805F499D93 for dynamic versioning or to clsid:CAFEEFAC-<major version>-<minor version>-<patch version>-ABCDEFFEDCBA for static versioning. For example, the classID for Sun JPI v1.4.2_06 would be clsid:CAFEEFAC-0014-0002-0006-ABCDEFFEDCBA. It is probably best to use the dynamic value of "clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"

With static versioning JPI_CODEBASE should be a URL to the .CAB file to download this exact version. A list of possible URLs is available at Sun. For dynamic versioning the URL consists of two parts, separated by a #-sign. The last part specifies the minimum required Java version. An example is Version=1,4,2,6 for Java v1.4.2_06. The first part of the URL should point to the CAB file to download when the required (or a higher) version is not installed on the workstation. One can point to the CAB file of specific version (e.g. http://java.sun.com/update/1.5.0/jinstall-1_5_0_02-windows-i586.cab for Sun Java v.1.5.0_02). One can also just use the version number of a family to automatically download the latest version of this family (e.g. http://java.sun.com/update/1.5.0/jinstall-1_5-windows-i586.cab will now download v1.5.0_03).

If we wish to use the application over HTTPS we need to change all external URLs in the HTML to also use HTTPS, otherwise Internet Explorer will raise a warning about combining secure and non-secure items on the same page. Luckily Sun's download sites are also available through HTTPS, so one can use https://java.sun.com/update/1.5.0/jinstall-1_5-windows-i586.cab#Version=1,4,2,4 to use at least v1.4.2_04 and if that's not available the latest version of the v1.5 family will be downloaded and installed.

JPI_MIMETYPE is a mime-type in a format like "application/x-java-applet;<version_type>=<implementation_version>". For static versioning, <version_type> should be set to "jpi-version". For dynamic versioning, this should be set to "version". For static versioning, the <implementation_version> should note the required Java version. For dynamic versioning, this should state the minimum required version. Appendix 6 of the Developer Guide shows the syntax of this <implementation_version>. One can just require a family or a very specific version/patch. Use "version=1.4.2_04" to require at least v1.4.2_04. This makes the entire mime-type "application/x-java-applet;version=1.4.2_04". However, this does not seem to work with Firefox, so it is better to use the more generic "application/x-java-applet;version=1.4.2".

JPI_DOWNLOAD_PAGE is the URL where Netscape/Firefox users can download the JRE. Sun's examples show this as http://java.sun.com/j2se/1.5.0/download.html. If using https this must be changed to prevent warnings in your browser, making it https://java.sun.com/j2se/1.5.0/download.html

Then run the application by making use of theconfig section in the formsweb.cfg called [jpi] e.g. http://.../forms/frmservlet?config=jpi

Alternatively if one wishes to run all forms with the Sun JVM change the setting in the formsweb.cfg file:

```
baseHTMLJInitiator=basejpi.htm
```

The latest Oracle supported version of the Sun JVM as of January 15th 2007 was 1.5.0_08.

7.20 Using Beans in DevSuite

There are a few issues with using Beans in the DeveloperSuite, whilst not serious they do require a few work arounds.

- The first relate to the positioning of the Bean within the Layout Editor. It has only been found possible to locate the bean area correctly by changing its location within the Property Sheet item for the bean item. In essence the bean area is just an item on which the Java Class is implemented so, one could say the position of the Bean area is not important. One could also play about with the item as a different type and then change it to a Bean, but this is messy and not an ideal solution.

- The second is the annoying FRM-13008 message that the Layout editor gives saying that it cannot find the Bean implementation class when entering the Layout area when the data block contains the bean with an assigned class.

  This can be resolved by changing the Windows registry and entering the name of the jar file to the end of the FORMS_BUILDER_CLASSPATH entry.

  I.e. FORMS_BUILDER_CLASSPATH on the author's machine is:

  D:\oracle\product\DS10102\forms\java\frmbld.jar;D:\oracle\product\DS10102\jlib\importer.jar;D:\oracle\product\DS10102\jlib\debugger.jar;D:\oracle\product\DS10102\jlib\utj.jar;D:\oracle\product\DS10102\jlib\dfc.jar;D:\oracle\product\DS10102\jlib\help4.jar;D:\oracle\product\DS10102\jlib\oracle_ice.jar;D:\oracle\product\DS10102\jlib\jewt4.jar;D:\oracle\product\DS10102\jlib\ewt3.jar;D:\oracle\product\DS10102\jlib\share.jar;D:\oracle\product\DS10102\forms\java\frmwebutil.jar;D:\oracle\product\DS10102\forms\java\frmall.jar;D:\oracle\product\DS10102\forms\java\Macrotone.jar;

  It is probably a good idea if using any possible beans within the Macrotone.jar file for developers to include this jar in the registry entry as in the author's setting displayed above.

8 Generic Forms/Reports Changes

The following changes have been made in more than one of the following applications. It is detailed here to avoid repetitive documentation.

1. A number of the forms have a procedure called 'set_window_size'. This procedure duplicates an existing Forms procedure held within the MACROTONE_LIBRARY.PLL file performing a similar function. To avoid duplication and to retain the supplied procedure, the Macrotone generated procedure within the forms has been renamed to set_mac_window_size.

A new form called APPLAUNCH has been written to enable a single front end form from where all Macrotone Forms based applications can be launched. See section 10 below for details.

8.1 Generic Settings

There were a few settings that were found to be applicable to all the forms encountered to date and these are specified here to aid others in their own conversion exercises.

In the server/formsweb.cfg file change the following settings:

```
width=950

height=600

splashscreen=no
```

A new parameter to assist in font scaling was useful:

```
clientDPI=96
```

This will affect font scaling and may help the display of fonts a little.

If one desires the full screen every time for a form set both the width and the height to 100%.

8.2 Useful PJC within the MACROTONE.JAR file.

Section 4.3.2 referred to the use of Java components within the Forms environment. It also provided mention of a couple of converted JSP that could be easily used within generic forms, which after all is the purpose of them being supplied in the first place. This section describes the JSP components present within the MACROTONE.JAR file and how they are used. The following is taken from the supplied Oracle documentation suitably modified where appropriate.

8.2.1 RoundedButton PJC

To use the Rounded Button PJC you must first create a normal Forms button, with the *Iconic* property set to **No**, and set the *Implementation Class* property for this item to **oracle.forms.Macrotone.RoundedButton**. Note that this property is case sensitive and must be entered exactly as shown.

There are no custom properties or any other instructions required for its use. Note that the RolloverButton PJC includes this functionality and more and may be preferred to be used.

8.2.2 RolloverButton PJC

To use the Rollover Button PJC you must first create a normal Forms button, with the *Iconic* property set to **No**, and set the *Implementation Class* property for this item to **oracle.forms.Macrotone.RolloverButton**. Note that this property is case sensitive and must be entered exactly as shown.

8.2.2.1 Using the PJC to create Oracle Look and Feel Buttons

To create a rounded edge to the button, simply include a leading or trailing rounded bracket in the label property (you can define this at build time). The brackets will be stripped out of the final label. So an instance of the PJC with the label "(Press)" will create a button with both ends rounded and the label "Press". "(Press" would create a button with just the left side rounded and the right side squared.

8.2.2.2 Using the Rollover Button Feature

To use rollover images with the PJC One of three methods can be adopted:

1. The standard button images as used in the Forms demos are all included in the rolloverbutton.jar file. To use one of these images simply set the Label property of the button to a value which consists of the keyword [ROLLOVER] followed by the name of the function required - e.g. exit (use lowercase). The PJC will then automatically load the correct images for the button from the JAR file. So in this case **[ROLLOVER]exit** would load the on and off images for an exit button. No code is required in PL/SQL to set up the button. The following function icons are defined in the JAR file.
    - add
    - delete
    - firstrec
    - lastrec
    - prevrec
    - nextrec
    - prevpage
    - nextpage
    - exit
    - help
    - print
    - query
    - save
    - rollback
2. Defining Images Declaratively - Rather than using the hard-coded images you can also explicitly define the On and OFF images in the label, using a comma to separate the pair with the ON image first and the OFF image next. You still use the [ROLLOVER] tag to prefix the label. The image names you use should include the path to the images (relative to the JAR or Codebase) and the extension for the image e.g. .gif. So the declaration in the label property for an exit button might be:

```
[ROLLOVER]/image_btn/exit_on.gif,/image_btn/exit_off.gif
```

3. Defining Images at Runtime - The images for a button can be changed at runtime in two ways.

    i. Simply use *SET_ITEM_PROPERTY(<name>, LABEL, <value>)* where value is a string in one of the formats prefixed with [ROLLOVER] as above.

    ii. Use SET_CUSTOM_PROPERTY with the custom properties **IMAGE_NAME_ON** and **IMAGE_NAME_OFF** to define the on and off images respectively.

```
SET_CUSTOM_PROPERTY ('CONTROL.EXIT_PB', 1,
   'IMAGE_NAME_ON', '/image_btn/exit_on.gif');
```

Property	**Get**	**Set**	**Valid Values / Return Value**	**Purpose**
IMAGE_NAME_ON	Yes	Yes	String	Defines the name of the image that is shown when the mouse rolls over the button
IMAGE_NAME_OFF	Yes	Yes	String	Defines the image shown when the mouse is not over the button
DEBUGMESSAGES	No	Yes	'true' | 'false'	Enables/disables debugging to the Java Console for this instance of the control.
DEBUGMESSAGES_ALL	No	Yes	'true' | 'false'	Enables/disables debugging to the Java Console for all instances of the control.

Table 2 - RolloverButton PJC Custom Properties

8.2.3 Calendar PJC

To use the Calendar PJC you must first create a normal Forms bean area.

Then set the *Implementation Class* property for this list item to **oracle.forms.Macrotone.CalendarWidgetWrapper**. Note that this property is case sensitive and must be entered exactly as shown. At runtime, this Bean will render as a simple button which the user can press to display the calendar.

8.2.3.1 Setting Properties on the PJC

The Calendar supports custom properties for setting the label of the launch button, and the initial date to display in the popup window. For instance the code to set the launch button label to "Get Date" would be:

```
SET_CUSTOM_PROPERTY('PJC.CALENDAR',1,'BUTTONLABEL','Get Date');
```

The first argument to Set_Custom_Property() is the name or id of the PJC enabled item. The second parameter defines the instance of the control that you wish to set the property on. This index number is one based and represents the physical control in the user interface (rather than the record number in the underlying block). You can use the constant ALL_ROWS to set the property on all instances of the PJC for this field. The third argument is the custom property that is being set on the PJC and the forth the value.

The property **setDate** is used to set the initial date for display. The date is expected in the format DD.MM.YYYY

8.2.3.2 Getting the Selected Date from the PJC

When the user selects a date in the PJC and Custom Item event is sent back to the Form which should be handled in a WHEN-CUSTOM-ITEM-EVENT trigger associated with the PJC bean area.

The event sent back from the PJC will be called **DateChange** and has a parameter **DateValue** which contains the selected date a sample When-Custom-Item-Event to return this value into a date field would:

```
Declare
   paramType number;
   newDateVal varchar2(80);
   newDate date := null;
begin
   if(:system.custom_item_event = 'DateChange') then
      get_parameter_attr(:system.custom_item_event_parameters,
      'DateValue',ParamType, newDateVal);
      newDate := to_date(newDateVal,'DD.MM.YYYY');
   end if;
   :EMP.HIREDATE := newDate;
end;
```

Property	**Get**	**Set**	**Valid Values / Return Value**	**Purpose**
BUTTONLABEL	No	Yes	String	Label to display on the launch button
setDate	No	Yes	String	Date to initialize the Calendar with.

Table 3 - Calendar PJC Custom Properties

8.2.4 InfoButton PJC

This PJC is used within the APPLAUNCH form. You must first create a normal Forms button, then set the *Implementation Class* property for this button to **oracle.forms.Macrotone.InfoButton**. Note that this property is case sensitive and must be entered exactly as shown.

The enhanced button has the capability to have either or both edges rounded in an Oracle Look and Feel style. To round a button edge, simply prefix or suffix the button label with an opening or closing round bracket. For instance (Press Me) as the label property for the button would result in a button with both edges rounded and the label "Press Me" (the brackets are removed in the final label)

8.2.4.1 Setting up the InfoButton

To set up the actual tool-tip functionality up on an InfoButton PJC you need to set two custom properties:

Set the text for the "tool-tip": Use the custom property **INFOBUTTON_TEXT** with a value of the required test to be displayed when the mouse moves over the button.

SET_CUSTOM_PROPERTY('MENU.BUTTON1',1,'INFOBUTTON_TEXT','The is the first choice in the Menu');

The first argument to Set_Custom_Property() is the name or id of the PJC enabled item. The second parameter defines the instance of the control that you wish to set the property on. This index number is one based and represents the physical control in the user interface (rather than the record number in the underlying block). You can use the constant ALL_ROWS to set the property on all instances of the PJC for this field. The third argument is the custom property that is being set on the PJC and the forth the value.

Define the position of the Display text item: Use the custom property **INFOBUTTON_FIELDPOS**. This property should be set to a string containing a X,Y comma separated pair providing the location of the target field in pixels relative to the top left hand corner of the window. The X,Y pair simply has to define a point anywhere within the target field, it doesn't have to exactly define the corner.

```
SET_CUSTOM_PROPERTY('MENU.BUTTON1',1,'INFOBUTTON_FIELDPOS','100,20');
```

Property	**Get**	**Set**	**Valid Values / Return Value**	**Purpose**
INFOBUTTON_TEXT	No	Yes	<string value>	The hint text to display in the target field
INFOBUTTON_FIELDPOS	No	Yes	X,Y pair as a String	The position of the target field to display the hint text into.
DEBUGMESSAGES	No	Yes	'true' | 'false'	Enables/disables debugging to the Java Console for this instance of the control.
DEBUGMESSAGES_ALL	No	Yes	'true' | 'false'	Enables/disables debugging to the Java Console for all instances of the control

Table 4 - InfoButton PJC Custom Properties

8.2.5 LabelledIconButton PJC

This PJC is an enhanced version of the RolloverButton described above, which will permit the placing of both an ICON and a label on a push button. To use the Rollover Button PJC you must first create a normal Forms button, with the *Iconic* property set to **No**, and set the *Implementation Class* property for this item to **oracle.forms.Macrotone.RolloverButton**. Note that this property is case sensitive and must be entered exactly as shown. The icons are stored within the Macrotone.jar file.

8.2.5.1 Using the PJC to create Oracle Look and Feel Buttons

To create a rounded edge to the button, simply include a leading or trailing rounded bracket in the label property (you can define this at build time). The brackets will be stripped out of the final label. So an instance of the PJC with the label "(Press)" will create a button with both ends rounded and the label "Press". "(Press" would create a button with just the left side rounded and the right side squared.

8.2.5.2 Forms configuration

Copy the Macrotone.jar to the /forms/java directory if it is not already there. Also if adding it anew ensure that the /forms/server/formsweb.cfg file has the jar file specified as part of the archive_jini variable

Within the form on the button peroperty sheet specify the implementation class as 'oracle.forms.Macrotone.LabelledIconButton'. Case is important.

The following custom properties can be set:

On image

```
Set_Custom_Property( 'BLOC.BEAN_ITEM',1, 'IMAGE_NAME_ON', 'icon_name');
```

Off image

```
Set_Custom_Property( 'BLOC.BEAN_ITEM',1, 'IMAGE_NAME_OFF', 'icon_name');
```

Turn debug on/off

```
Set_Custom_Property( 'BLOC.BEAN_ITEM',1, 'DEBUGMESSAGES', 'true');
```

Turn debug all instance of this PJC on/off

```
Set_Custom_Property( 'BLOC.BEAN_ITEM',1, ' DEBUGMESSAGES_ALL', 'false');
```

Property	**Get**	**Set**	**Valid Values / Return Value**	**Purpose**
IMAGE_NAME_ON	Yes	Yes	String	Defines the name of the image that is shown when the mouse rolls over the button
IMAGE_NAME_OFF	Yes	Yes	String	Defines the image shown when the mouse is not over the button
DEBUGMESSAGES	No	Yes	'true' | 'false'	Enables/disables debugging to the Java Console for this instance of the control.
DEBUGMESSAGES_ALL	No	Yes	'true' | 'false'	Enables/disables debugging to the Java Console for all instances of the control.

Table 5 - LabelledIconButton PJC Custom Properties

8.2.6 ProgressBar PJC

The ProgressBar Pluggable Java Component (PJC) provides a simple progress bar widget. The bar supports the setting of the background and foreground colours of the bar to a limited range of colours, and displays the percentage completed in the centre of the bar.

8.2.6.1 Using the PJC in your Form

To use the ProgressBar PJC you must first create a normal Forms bean area.

Then set the *Implementation Class* property for this list item to **oracle.forms.Macrotone.ProgressBarPJC**. Note that this property is case sensitive and must be entered exactly as shown.

8.2.6.2 Setting Properties on the PJC

The Progress Bar is incremented using the custom property **PROGRESS_PERCENT_VALUE**. this can be set to any value between 0 and 100. The Bar will then be rendered with the correct size of fill and with the percentage number written in the bar. For instance to set the bar to reflect a value of 45% the call would be:

SET_CUSTOM_PROPERTY('PJC.PROGRESSBAR',1,'PROGRESS_PERCENT_VALUE',45);

The first argument to Set_Custom_Property() is the name or id of the PJC enabled item. The second parameter defines the instance of the control that you wish to set the property on. This index number is one based and represents the physical control in the user interface (rather than the record number in the underlying block). You can use the constant ALL_ROWS to set the property on all instances of the PJC for this field. The third argument is the custom property that is being set on the PJC and the forth the value.

As well as setting the percentage value of the bar you can also set the colour value for the background and foreground. The control only supports a limited number of string representations of colours as it stands but this could be enhanced to support more colours or RGB codes to increase the colour range. The supported colours are (not case sensitive):

- Blue
- Red
- Green
- White
- Yellow
- Orange

8.2.6.3 Custom Property Summary

Property	**Get**	**Set**	**Valid Values / Return Value**	**Purpose**
PROGRESS_PERCENT_VALUE	Yes	Yes	Number between 0 and 100 /String	The percentage represented on the bar
PROGRESS_BACKGROUND_COLOR	No	Yes	String (see above for valid values)	Background colour for the bar
PROGRESS_FILL_COLOR	No	Yes	String (see above for valid values)	Foreground (fill) colour for the bar

Table 6 - ProgressBar PJC Custom Properties

8.3 DDE and the Web

There is at least one form that is built using Dynamic Data exchange (DDE).

To briefly summarize, the report accepts a start date and an end date and generates an Excel spread sheet on the users PC that they can access and use.

Using the web this is best achieved by opening a browser window containing an Excel spreadsheet allowing the user to then save the spreadsheet on their PC if they desire. Only by using a JSP is this possible and DDE is not an option since that assumes a client/server approach and there is no desire to save the spreadsheets upon the Application Server itself.

The approach taken was to initially develop the JSP report and then to build a JSP parameter form around this to pass in the date values to the JSP report which would perform the browser actions. This is an enhanced version of the examples seen upon the web and builds upon the lessons of others.

8.3.1 The JSP Report

A fresh start had to be taken to develop the report.

The basic steps are as follows:

- Create an MS Excel template

  The first step is to open an Excel session and create a sample template, containing generic information such as the title, logo's column headers etc. At the same time enter a single row of typical data formatted in the desired manner. Then save the Excel spreadsheet as a Web page.

- Open the Web Page in Reports Builder
  - Having opened the report, double click on the Web source node to display the HTML code for the Excel spreadsheet. Note how most of the XML and HTML information has already been provided, including the Report JSP tags.
- Add a data source
  - Now click on the Data Model icon in the toolbar to display the DATA Model View of the report. Opt to add a data source. In our situation we already had the SQL query since we could extract it from the 'old' version of the form, otherwise one would code in the SQL code required. Once we have the query we can proceed to modify the Web source to display the report in Excel.
- To Modify the Web Source
  - Click on the Web Source icon and then we can make the desired changes. The first thing to change is the HTTP content type to the MIME Type by adding the line:
    - <%@ page contentType="application/vnd.ms-excel" %>
  - Delete the blank lines above the <html> tag to respect Excel format.
  - Use the Oracle Reports JSP tags to add the data retrieved by the SQL query to the report. Search for the sample lines added to the Excel report above. Each row is saved as an HTML Table row tag <tr>.
  - Using the Reports JSP Tags, add a Reports repeating frame tag to loop around the Data Model Group. Move the closing repeating tag after the </tr> tag.
  - Change the name of the group in the opening repeating tag <for each>, and provide a unique identifier for the id.
  - Map the cells of the Excel spreadsheet to the corresponding field from the data model. It might help to change the screen display to view the Object navigator and the Web source side by side. Repeat for each filed of the report.
  - Save the report as a 'reports JSP'.
  - Run the report using the Run Web Layout icon in the toolbar. The execution creates a temporary HTML file and launches the browser. The browser does not

launch Excel because the file is saved as an HTML file. To launch Excel from the browser requires the run to be from the Reports Server.

- Test the report with the Reports Server

8.3.2 JSP Parameter form

One of the most useful tools in use on the Web is JavaScript. With JavaScript at your side, one can process simple forms without invoking the server. And when submitting the form to a program is necessary, you can have JavaScript take care of all the preliminary requirements, such as validating input to ensure that the user has dotted every 'i' and crossed every 't'.

8.3.2.1 Creating the form

There are few differences between a straight HTML form and a JavaScript-enhanced form. The main one being that a JavaScript form relies on one or more event handlers, such as onClick or onSubmit. These invoke a JavaScript action when the user does something in the form, like clicking a button. The event handlers, which are placed with the rest of the attributes in the HTML form tags, are invisible to a browser that doesn't support JavaScript. Because of this trait, one can often use one form for both JavaScript and non-JavaScript browsers.

Typical form control objects -- also called "widgets" -- include the following:

- Text box for entering a line of text
- Push button for selecting an action
- Radio buttons for making one selection among a group of options
- Check boxes for selecting or deselecting a single, independent option

It is not worth enumerating all the attributes of these controls (widgets), and how to use them in HTML. Most any reference on HTML will provide one with the details. For use with JavaScript, it should always be remembered to provide a name for the form itself, and each control used. The names allow one to reference the object in your JavaScript-enhanced page.

The typical basic form looks like this. Notice the provided NAME= attributes for all form controls, including the form itself:

```
<FORM NAME="myform" ACTION="" METHOD="GET">
Enter something in the box: <BR>
<INPUT TYPE="text" NAME="inputbox" VALUE=""><P>
<INPUT TYPE="button" NAME="button" Value="Click"
onClick="testResults(this.form)">
</FORM>
```

- *FORM NAME="myform"* defines and names the form. Elsewhere in the JavaScript one can reference this form by the name *myform*. The name given to the form is up to the developer, but it should comply with JavaScript's standard variable/function naming rules (no spaces, no weird characters except the underscore, etc.).
- *ACTION=""* defines how it is desired that the browser should handle the form when it is submitted to a program running on the server. As this example is not designed to submit anything, the URL for the program is omitted.

- *METHOD="GET"* defines the method data is passed to the server when the form is submitted. In this case the attribute is buffer as the example form does not submit anything.
- *INPUT TYPE="text"* defines the text box object. This is standard HTML markup.
- *INPUT TYPE="button"* defines the button object. This is standard HTML markup except for the onClick handler.
- *onClick="testResults(this.form)"* is an event handler -- it handles an event, in this case clicking the button. When the button is clicked, JavaScript executes the expression within the quotes. The expression says to call the testResults function elsewhere on the page, and passes to it the current form object.

8.3.2.2 Getting a value from a form object

Load the page and type something into the text box. Click the button, and what you typed is shown in the alert box.

```
<HTML>
<HEAD>
<TITLE>Test Input</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function testResults (form) {
   var TestVar = form.inputbox.value;
   alert ("You typed: " + TestVar);
}
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME="myform" ACTION="" METHOD="GET">Enter something in the box:
<BR>
<INPUT TYPE="text" NAME="inputbox" VALUE=""><P>
<INPUT TYPE="button" NAME="button" Value="Click"
onClick="testResults(this.form)">
</FORM>
</BODY>
</HTML>
```

Here's how the script works. JavaScript calls the testResults function when one clicks the button in the form. The testResults function is passed the form object using the syntax `this.form` (the '`this`' keyword references the button control; `form` is a property of the button control and represents the form object). The form object has been given the name *form* inside the testResult function, but it can be any name you like.

The testResults function is simple -- it merely copies the contents of the text box to a variable named TestVar. Notice how the text box contents are referenced. The form object to be used has been defined (called *form*), the object within the form to be used (called *inputbox*), and the property of that object required (the *value* property).

8.3.2.3 The developed JSP Parameter form

Using the above the form below was developed. The RunReport function is built to display the input parameters (using the alert function) and then to call the developed JSP report. The form name is MY_PARAMFORM and the parameters named P_1 and P_2. Note that they are case sensitive. The username and password are deliberately not shown in the example below.

The developed JSP parameter form is shown below. There is still a number of improvements that could be made, particularly in the screen layout, but it is sufficient to illustrate the process steps.

```
<HTML>
<title>JSP Parameter Form</title>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
 function RunReport(form)
 {
   var START_DATE = form.P_1.value;
   var END_DATE = form.P_2.value;
   alert("Running report between " + START_DATE + " and " + END_DATE);

   var URL = "http://localhost/reports/test.jsp?"
   + "server=rep_macro_s1_frhome1&"
   + "report=test.jsp&"
   + "userid=username/password@database&"
   + "p_start_date='" + START_DATE + "'"
   + "&p_end_date='" + END_DATE + "'";
   runWindow = window.open(URL);
 }
</SCRIPT>
</HEAD>
<BODY>
<body bgcolor="#FFFFFF" text="#000000">
<p><font size="6" color="#0066FF"><b><font face="Arial, Helvetica, sans-serif"
color="#000099">Parameter
 Form</font></b></font></p>

<FORM NAME="MY_PARAMFORM" METHOD="GET">
<TABLE>
<TR>
<p>Enter the start and end dates in DD-MON-YY format then click the <b>Run
Report</b>
button to display the report in Excel.</p>
<font face="Arial, Helvetica, sans-serif">Start Date:
<input type="text" name="P_1" value="">
</font><br>
<font face="Arial, Helvetica, sans-serif">End Date:
<input type="text" name="P_2" value="DD-MON-YY">
</font><br>
<TD>
<INPUT TYPE="SUBMIT" NAME="RUN_REPORT" VALUE="Run This Report"
onClick="RunReport(this.form)">
</TD>
</TR>
</TABLE>
</FORM>
</BODY>
</HTML>
```

9 TEST Conversion

The TEST forms/reports application consists of a single Oracle Form that in turn calls a number of different reports. The structure of the form is reasonably simple; however there are a few features that do not appear to be used.

9.1 Directory Monitoring option.

There is an option to monitor directories that would have been invoked from a button on the main form. This button, whilst present is not enabled so it is doubtful whether it has ever been used. A second reason for think this, is that the code associated with the button has a number of direct operating system calls which would involve a lot of rework where the functionality to be enabled upon the applications server. For that reason this code has not been checked or modified.

9.2 Testing

The options that involve creating new store details file type etc., have not been extensively tested, due to the authors lack of knowledge of the impact of testing these on the live system. A code inspection reveals that there should be no problems, but these need to be checked more thoroughly before the decision is made to go live.

All other features have been tested both upon the Development Installation and upon a Form/Reports Applications Instance upon a local PC.

9.3 TEST Forms

Form Name	**Form Type**	**Comments**
TEST_main	Form	Only form.
TEST_mnu	Menu	Part of the TEST_main form

Table 7 - TEST Form Objects

9.4 TEST Reports

The Oracle reports fall into 4 categories as invoked from the main TEST form. There appear to be 2 reports that are not used.

Report Name	**Parameter Form**	**Report Type**	**Comments**
Configuration Details	YES		Does not appear to be called
Control_by_sf	YES	Control report	
Daily_Full_Log	YES	Error report	
Error_by_sf	NO	Error report	
Exceptions	YES	Error report	
File_List	NO	Database report	
Full_Log	NO	Error report	
Misc_Full_Log	NO		Does not appear to be called

Report Name	Parameter Form	Report Type	Comments
Next_Excepted	NO	Control report	
Outgoing	YES	Error report	
Record_List	NO	Database report	
Store_List	NO	Database report	
Vendor_Details	NO	Vendor report	
Vendor_not_confirmed	NO	Vendor report	

Table 8 - TEST Report Objects

9.5 Outstanding TEST problems

The following are the known outstanding problems with the TEST conversion to 10G.

There is a problem with some reports and the formatted output when the desformat is set to be 'htmlcss'. If the format is set to 'html' and the cascade style sheets are not being used the output is displayed correctly. Needs to be reported as a bug. Workaround is to change the output format in the form [Program units 'which_vendor_report' and 'which_error_logs_report' in the TEST_MAIN form] making the report request.

The problem is an Internet Explorer 6 problem, where it cannot display (format) the html correctly. The problem does not exist with the Firefox browser, and if one takes the generated html output that does not display correctly on IE 6 and runs it directly into Firefox 2.0, it also displays correctly so it is not the generated output from Oracle.

This is a problem with the output format. Use of a desformat of 'html' or 'pdf' bypasses the problem.

1. Printed output line up still needs to be addressed. Work around is to use the PDF output format and then no other changes relating to printing need to be considered.

10 APPLAUNCH form

This is a new form written especially to enable the launching of all Macrotone Forms based applications. It was derived form a sample form and enhanced as appropriately for Macrotone usage.

The two buttons perform very specific functions. The question mark or HELP button will call a second browser window (or an additional tab in a multiple tabbed browser such as Internet Explorer 7). The door or Exit button will close the browser window. The browser will close if it is the last window open. Moving the mouse cursor over the two buttons will cause the button icons to change slightly.

The menu items on display are changed within the form itself so as additional applications are converted or written, the form will need changing to reflect the new form within the launcher.

10.1 APPLAUNCH Usage

The form is accessed upon the application server directly from the following URL:

http://server.macrotone.local:7777/forms/frmservlet?form=applaunch

Pressing the return key will display the form in a browser window.

No logon is required for the use of this form.

10.2 Installation of APPLAUNCH

Place the applaunch.fmb and applaunch.fmx files in the D:\Macrotone\forms directory upon the Application server. Provided this directory is in the default.env file in the forms/server directory the forms will be found.

The information text file that is displayed via the help icon needs to be placed in the D:\Macrotoneforms\doc directory. Then a configuration file needs to be changed to allow the file to be displayed via the web browser. Depending upon the system will determine how this is configured.

On an OC4J system i.e. Developer Suite a change has to be made to the OC4J configuration:

Modify the orion-web.xml file:

open %ORACLE_HOME%\j2ee\DevSuite\application-deployments\forms\formsweb\orion-web.xml

Add the following lines within the <orion-web-app> tags, making sure to replace the %MACROTONE_HOME% with the correct value:

<virtual-directory virtual-path="/Macrotonedoc" real-path="%MACROTONE_HOME%\forms\doc" />

For example MACROTONE_HOME might be ‘D:/projects/Macrotone/forms’.

On the Application server the change has to be made to a couple of configuration file.

Change the forms.cfg file to add a line similar to the following making sure to replace the %MACROTONE_HOME% with the correct value:

```
AliasMatch ^/forms/Macrotonedoc/(..*)
"%MACROTONE_HOME%\forms\doc/$1"
```

Also modify the file %ORACLE_HOME%\j2ee\OC4J_BI_Forms\application-deployments\formsapp\formsweb\orion-web.xml and add the line:

```
<virtual-directory virtual-path="/Macrotonedoc" real-
path="%MACROTONE_HOME%\forms\doc" />
```

A restart of the service is required.

10.2.1 Problems Determination.

1. Icons not displayed on forms:

   Check that the Macrotone.jar file in present in the archive_jini parameter in the webconfig.cfg file in the forms/server directory.

2. Default documents on applaunch is not displayed.

Check the configuration of the Macrotonedoc directory in the orion-wel.xml file and if on the Application server check the setting in the forms.cfg file.

10.3 Adding new applications.

A new application is added to the list by changing the APPLAUNCH form itself. The first step is to open the form in the Forms Builder application. Once open a new item needs to be added to the PJC data block.

Once the object is subclassed, then the code for the BUTTONDEFS package specification needs to be changed to add some additional code for the new item being added, such as the informational text and the form configuration being used.

The code will look something like the following:

```
 setButtonData('PJC.STORES',
                     'Stores to HQ Confirmation',
                     'STORES',
                     'STORES',
                     False,
                     The Macrotone STORES application implements viewing (and control) of
                     the confirmation files sent form the Stores to the Head Office. Primarily
                     used by Stores support the application displays reject messages and other
                     rejection criteria upon goods received by the stores.');
```

The input parameters to SetButtonData are as follows:

buttonName in varchar2	Name of button as defined in data block.
buttonLabel in varchar2	Label to appear on Form Button
moduleDir in varchar2	Directory in which form is located on server
callModule in varchar2	The module calling name.
JRE13only in Boolean	Specifies whether a specific Java version is required.
buttonInfo in varchar2	Textual information displayed on the form as the mouse cursor is positioned over the button.

The Macrotone STORES application implements viewing (and control) of the confirmation files sent form the Stores to the Head Office. Primarily used by Stores support the application displays reject messages and other rejection criteria upon goods received by the stores.');

The code needs to be added to the SetUPButtons routines. All other code should not need to be changed unless there is some specific requirement for the new application that is required.

Now recompile the form and test, and the new item(s) should be shown on the running form.

NOTE: There are 2 possible configurations set up in the BUTTONDEFS package to enable calling of web forms either by use of forms configuration or a base form. This code may need modification depending upon where the form is located. It is intended that a form configuration is set up for each Macrotone application to make installation and configuration easier and more precise.

10.4 Get Client Info Form

This form is a simple 'hack' of an Oracle demo form modified to be used in the Macrotone environment. It consists of a single form, together with a java jar file (getclientinfo.jar). Icons were modified to be those contained within the Macrotone.jar file. The clientinfo.jar file has to be placed in the forms/java directory and the formsweb.cfg file modified to accept the jar on the archive_jini parameter. Forms are located within the standard C:\macapps\forms directory upon the application server.

The form is intended to be called from the APPLAUNCH form, but could be called directly if required. It was retained to enable users, and support personnel to obtain information easily about client machines when trouble shooting possible problems.

No login is required for the use of this form.

11 Additional Useful Information

This section contains generally useful information that I have discovered but not found documented anywhere within the Oracle documentation.

11.1 To convert FMB source code to XML

Oracle Forms 10g Release 2 has changed the binary format of the source code particularly in masking trigger code. If you load a Form from Release 2 to Release 1, you cannot open the form. One way around this problem is to convert the form to XML and then convert it back. Another use is to change the object link criteria, perhaps removing any hard coded path information.

Step 1: On the PC with Developer Suite Release 2 installed

To convert the xml back to fmb in 10g Release 2, in DOS prompt:

To convert all FMB in the same folder

```
C:\YourCodeFolder> frmf2xml *.fmb
```

To convert a particular FMB file, just specified the filename

```
C:\YourCodeFolder> frmf2xml mbank.fmb
```

Step 2: On another PC with Developer Suite Release 1 installed

To convert the xml back to fmb in 10g Release 1, or to convert all XML back to FMB

```
C:\YourCodeFolder> Ifxml2f90 *.xml OVERWRITE=Y
```

Please note that if there is an existing FMB file, it will be overwritten with Release 1 format. Release 1 FMB can be accessed from both Forms Builder Release 1 and 2.

11.2 Usage of Object Libraries [OLB files]

Some fun was had with the use of object libraries in forms. It seems that the libraries need to be pointed to by a number of different parameters depending upon the environment in which one is running.

For example the Forms Builder requires that the library is contained within the Windows registry parameter 'FORMS_PATH' [at 10G version]. The OC4J environment requires that the library is contained within the FORM_PATH as specified in the default.env file [or other environment file that may be in use, located within the Forms/server directory upon the PC.

Business Intelligence

12 Business Intelligence and Discoverer

Whilst not strictly belonging to this document, this will serve as a convenient location to save information until such time as it becomes a document in its own right.

12.1 Connections to Discoverer

To connect to the Discoverer service use the following URLs:

http://server:7780/discoverer/plus	For the Discoverer Administrator

OR

http://server:7780/discoverer/viewer	For the Discoverer Viewer

Most users will use the viewer access. Note that ports 7779 could also be used and will display the same information. The main difference is that the lower numbered port connects to the Web cache where as the higher numbered port connects to the HTTP server itself directly.

12.2 Discoverer OLAP catalogue

The OLAP catalog is owned by the D4OSYS user. Access to the administration of the OLAP catalog is through the Enterprise Manager console accessed through the URL:

http://server:7780/emd

One will be prompted for the administrator userid and password. Once connected, navigate to the Discoverer -> Manage Catalog page and enter the connection details for the catalog. Once completed the page will return a pick list to which one can add user etc. A role is defined for catalog users which is the D4OPUB role. Any user with this database role should be able to connect to the Discoverer OLAP catalogue.

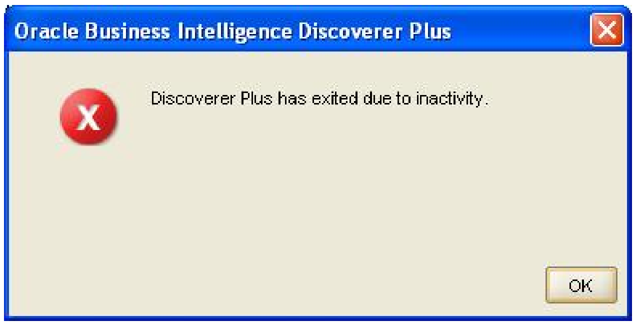
To connect to the Discoverer OLAP catalogue from the main connection page, use one of the URL's specified above and change the drop down 'Connect To' information to choose the 'OracleBI Discoverer for OLAP' option.

At the current time there is no usage of Oracle OLAP within any of the Macrotone databases. This feature is an extra cost option and it is unknown as to whether Macrotone have subscribed to this Oracle feature. It is strongly suspected that they have not. Consequently there are no OLAP database objects and hence no requirement to use the Discoverer OLAP tool.

Although the Discoverer OLAP catalog has been created and hence exists within the databases, there is and are no plans to develop or use this feature.

12.3 Discoverer Timeout

The user may receive the following alert after a period of inactivity.



Discoverer Plus has exited due to inactivity.

Figure 2 - Inactivity Timeout

The default timeout on the Application server is currently set to 30 minutes.

To change the Discoverer Session timeout to a different value:

1. Edit the file $ORACLE_HOME/discoverer/util/pref.txt.
2. Under the heading [Session Manager] there is a parameter called timeout.
   Timeout = 1800 (Number of Seconds before timeout)
3. Save the pref.txt file.
4. Run the applypreferences script.

When changing the timeout value to be higher than 30 minutes you will need to change the Discoverer servlet timeout as well. The servlet timeout must always be higher or equal to the Discoverer session timeout.

Steps to change the Discoverer Servlet timeout:

1. Edit the file ORACLE_HOME/j2ee/OC4J_BI_Forms/applications/discoverer/WEB-INF/web.xml
2. Add the following entry to the file:
   ```
   <session-config>
   <!-- Session timeout in minutes -->
   <session-timeout&gt60
   </session-config>
   ```

   The file should look something like:
   ```
   <web-app>
   <servlet>
   <servlet-name>...</servlet-name>
   <servlet-class>...</servlet-class>
   </servlet>
   <session-config>
   <session-timeout>60</session-timeout>
   </session-config>
   </web-app>
   ```
3. Save the web.xml file.
4. Run command 'dcmctl updateConfig -ct oc4j -v' to update the configuration.
5. Restart OC4J with command 'dcmctl restart -ct oc4j -v'.

The same can be achieved through the OEM website.

Using the OEM website is the Oracle recommendation for changing the configuration files. Navigation path: OC4J_BI_Forms > Application:discoverer > Web Module:web > Properties
Set the Session Timeout box. Click Apply and then restart the OC4J_BI_Forms

12.4 Problems encountered

1. The demo installation of the bi_samples revealed a few problems with the installations scripts. These were easily resolved but indicate a lack of testing on someone's part. Once installed on the JDADEV environment a further problem revealed itself in that the SH.PRODUCTS_DIM (dimension) was invalid. Installation of the 'demo_dim' package enables a description of the dimension to be displayed (use DBMS_DIMENSION in Oracle 10G) which indicated that there was on the initial statement. Code inspection indicated that the underlying SH.PRODUCTS table was different in column definitions form that supplier as standard in Oracle 9. Using the sample Oracle 9 code the dimension creation statement was successfully recreated, with the new column definition. Whether this will have an effect upon the running of the samples is unknown at this time.

   There is no apparent effect found so far. The scripts that try to install the EUL and the OLAP samples in one go, are the most incorrect. Running the individual scripts worked fine.

2. The installation of the D4OSYS schema for the Discoverer schema requires a specific execute grant on DBMS_LOB to work correctly. The install will initially fail, but will create the user, so this is the opportunity to grant execute permission upon dbms_lob, and then the installation can be retried which worked successfully on the test instance.

3. The Discoverer EUL owner will require 'SELECT ANY TABLE' privilege to enable the creation of the End User Layer. Allow a good 30 mins for a creation of an EUL.

4. The following selects were granted to the discoverer users:

Privilege	**Object**
SELECT	V$sql
SELECT	V$session
SELECT	V$sesstat
SELECT	V$parameter
Execute	Dbms_job

Table 9 - BI grants

5. A blank page is displayed when trying to use the plus connection to the EUL. This is a configuration issue. The first is that the trial installation is on a machine that has a dynamically assigned IP address which can change and confuses the Apache setup. The browsers seemed to have a problem loading the java applet (oracle.disco.DiscoApplet) with using the discoverer plus address although the viewer page works fine. The Apache config was incorrect since it still wanted to connect to the actual hostname. Needed to modify the httpd.conf Apache file to specify the host name as 'localhost', which then enabled the applet to be downloaded successfully. This was a local configuration issue based upon using a DHCP machine for development.

6. If the database instance has timed_statistics set to FALSE timing information will not be available for queries.

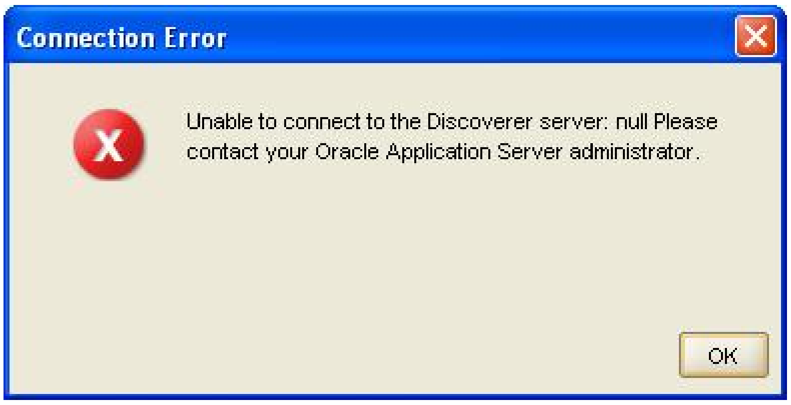
12.5 IE7 and Discoverer connection

One strange feature observed relates to the use of IE7 and connections to discoverer.

Fill in the connection information as follows:

Enter the correct username, password etc and click GO or press the enter button.

This will launch a connection to the Application server. On a successful connection IE7 will show as a new tab within the browser, however the connection tab will show a connection error as follows:



Unable to connect to the Discoverer server: null Please contact your Oracle Application Server administrator.

Figure 3 - Connection error message

This can safely be ignored provided the new tab has opened. Very strange but related to the use of the unsupported browser (IE7). It is almost as if there is a second connection attempt that will fail due to the lack of any parameters being specified.

12.6 Discoverer Query settings

To prevent the discoverer query from informing one that the query will take x minutes to run one can change the settings within the perf. txt (located in the ORACLE_HOME/discoverer/util directory) file as follows:

QueryBehavior = 0

Action to take after opening a workbook (0 = Run Query Automatically, 1 = Don't Run Query, 2 = Ask for Confirmation)

PredictionThresholdSeconds = 60

Warn the user if the predicted query time exceeds the following value. (seconds). Min. value = 1 Max. value = N.A (see description at the begining of file for range of valid values)

PredictionThresholdSecondsEnabled = 1

Query prediction threshold disabled (0) or enabled (1)

Do not forget to run the apply preferences file after the change

Note: It is also possible to turn off the whole of the query predictor. The setting inside pref.txt is called QPPEnable. Setting this to 0 turns off the whole of the query predictor.

This generally seems to improve Discoverer performance but may not be useful to the uninformed end user.

12.6.1 Loss of Drill to icon

It has been observed that occasionally the drill icon disappears. This is probably related to the setting in the perf.txt file named ShowDrillIcon. This setting controls whether the drill icon is displayed. Setting this to false, saving the pref.txt, applying the preferences and restarting the Discoverer server component should take care of it.

There is an associated issue with the loss of the drill icon in Desktop Discoverer, which is suspected to be due to the Desktop version being unaware of the hierarchy. This can be resolved by defining a hierarchy within the folder itself, but is really overkill.

12.6.2 Hiding drill to items

When Drilling to a related item, several folder items are displayed and can be drilled. Use the following to disable or hide an item in the list.

1. Connect in the Discoverer Administrative edition as the eulowner.
    a. Open the Business Area and Folder where the item resides.
    b. Highlight the item, right click and choose 'Properties'.
    c. Change 'Default Position' from 'Top' to 'Data Point'.
    d. Reconnect in Plus and test the Drill to Related Item. The item should no longer be in the list of drillable items.
2. Or Disable Drilling for the user by:
    a. Open the Discoverer Administrative edition.
    b. Choose Tools - Privileges and find the user.
    c. Uncheck Item Drill and the Link on the column for 'Drill to Related' is no longer present.

12.6.3 Drill to detail in another worksheet (Hyperdrill)

The Item Class Wizard in the Admin Edition is used to build a hyperdrill.

Choose Insert-->Item Class from menu to launch the Item Class Wizard or choose Item Class from the Administration Task List.

In the first wizard step, select the Drill to Detail checkbox as well as the List of Values checkbox.

Choose Next and enter the following information:

1. Select the item containing the list of values.
2. Select the items that will use this item class.

NOTE: If you do not select the List of Values option in the Item Class Wizard, you will need to apply the item class to the specific items later.

After selecting the detail folders, select Next to enter a name and description in the final step of the wizard.

12.6.4 LOV Item class setups

Take a look at these two snippets of code:

```
SELECT 'MINI' Prodsize from dual
UNION
SELECT 'SMALL' Prodsize from dual
UNION
SELECT 'MEDIUM' Prodsize from dual
UNION
SELECT 'LARGE' Prodsize 4 from dual;

SELECT DECODE(ROWNUM,1,'MINI',2,'SMALL',3,'MEDIUM',4,'LARGE') "PRODSIZE"
FROM DUAL
CONNECT BY
LEVEL <= 4;
```

Both of these generate a list of four items. One may well be familiar with the first construct but in actual fact the second one has a better explain plan and is the one that it is best to use inside a CUSTOM folder.

If a list of values has a known set of values you can use code like the above to generate a list of values that never touches the database.

12.6.5 Alternative Sorts

Sometimes, when you look at a list of values, you will see that it is not in the order that you would like. To overcome this, you can use what is called an alternative sort. In order to use an alternative sort, you must have another unique item available in the folder that contains the item on which you created the list of values.

This alternative item must be one of the following:

- Another unique and available item, such as month end date
- - An artificially created sequence number

Among the most common data elements that sort in the wrong order are days of the week and months of the year. For both of these you will need to create an alternative sort if you want your users to see the list displayed in the right order.

To add an alternative sort, use the following workflow:

1. Click the Item Classes tab.
2. Expand the list of item classes.
3. Locate the list of values that needs the alternative sort.
4. Right-click on the item class, and from the pop-up menu, select Edit Item Class.
5. In the Edit Item Class dialog box, click the Alternative Sort tab.
6. Select the item that will be used to generate the new alternative sort.
7. Click the OK button.
8. Recheck the list of values.

12.6.6 Changing Oracle logo in web discoverer

This was the solution that worked:

1. Goto Discoverer Plus configuration.

2. Goto Logo but change the option button from File to URL and type the URL like http://servername/corporate.gif (make sure you place the corporate.gif file in \oracle\Discoverer_home\Apache\Apache\htdocs..)

3. Restart the application server

12.6.7 Moving a workbook from one database to another

It is often necessary to move a workbook from one database to another, i.e. from the development system to the live/production system.

There are 2 options:

- use Discoverer Administrator to export/import the workbook
- use Discoverer Desktop to save the workbook to the local drive.

It is not possible to save a workbook to the local drive using Discoverer Plus.

12.6.8 EUL Library Version 10.2.2.54.25 requires EUL tables 5.1.0.0 message

This message is displayed when a connection has not specified the correct EUL layer. The usual cause is that the End User Layer [EUL] parameter has not been specified as part of the connection detail. See **Error! Reference source not found.** for an example of the connection screen and the location where one would specify the EUL.

12.6.9 Discoverer Plus login fails at first attempt

A problem has been seen where each time when starting up Oracle Discoverer Plus (10.1.2) and trying to login, one needs you need to login twice to because the first time login failed. This behaviour is caused by an incorrect conversion of the URL in OC4J.

The quick fix solution (for end users) is to startup Oracle Discoverer Plus using the complete servername and domainname in the URL (instead of using the shortname). Start Discoverer Plus throught: http://servername.domain.com/discoverer/plus

Changing the URL seems to have resolved the problem.

12.7 Observations

12.7.1 Import Functions

To import a function into discoverer the EUL owner has to have a specific execute privilege on the function that it is desired to import.