ORACLE®
FUSION MIDDLEWARE
FORMS

Oracle Forms 12*c*

New Features – Rev. 2

ORACLE®

Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Table of Contents

Introduction

This document is intended to outline some of the many new features found in Oracle Forms 12c (12.2.1). This document alone does not represent a complete collection of all the new features and enhancements introduced into this new release. Features that are included herein represent a cumulative catalog of features from all minor versions in the 12c (12.2.1) family. It is assumed that readers will be using the latest product version and therefore can take advantage of any feature discussed.

This high level document is only for reference and an introduction to some of the new features. Complete details are available in the product documentation, including the Form Builder Help.

For a more information and usage details on these and other new features, refer to the Oracle Forms Documentation Library and Oracle Form Builder Help.

Forms Runtime and Administration Features

Oracle BI-Publisher Integration

In previous versions, Oracle Forms offered integration with Oracle Reports. In this release, Oracle Forms includes integration with Oracle BI-Publisher. Developers can choose between using one or both reporting tools.

The integration with Oracle BI-Publisher has been designed to closely resemble the integration provided previously for Oracle Reports. A new property named REPORT_OBJECT_TYPE will be used to identify if the referenced REPORT_OBJECT is intended for Oracle Reports (OraReports) or Oracle BI-Publisher (OraBIP). This can only be set at design-time. Most other related built-ins will remain the same as they were for Oracle Reports. New properties have been added to accommodate BI-Publisher. Refer to the Form Builder Help topic "SET_REPORT_OBJECT_PROPERTY" for a complete list of new properties and their descriptions.

The following assumes Oracle BI-Publisher has already been installed, configured, and is accessible from the installation hosting Oracle Forms. Refer to the BI-Publisher documentation for information on its installation and configuration.

To integrate with BI-Publisher, the following basic steps are necessary.

1. In the Form Builder Object Navigator, create a Report object.
2. Access the properties for this new object in the Property Palette.
3. Set the Report Type property to OraBIP.
4. In the Property Palette, set the required and desired optional properties. (Refer to the Builder Help for details)
5. Create the desired trigger or program unit that will be used to make the call to BI-Publisher.

The BI-Publisher username and password must be passed programmatically in order for the request to succeed.

```
DECLARE
        bi_username       varchar2(20);
        bi_password       varchar2(20);
        repid             REPORT_OBJECT;
        v_rep             varchar2(256);
        rep_status        varchar2(256);

        -- Because Forms calls BIP with Java, we want to trap any Java errors that may occur.
        ex          ora_java.jobject;
        ex_msg   varchar2(255);
BEGIN
        -- Identify the Report Object.
        repid := find_report_object ('BI_EMP');
        -- Before reaching this point be sure to obtain the username and password for BI-Pub.
```

```
         -- Pass in the username and password. This is required.
         SET_REPORT_OBJECT_PROPERTY (repid, BIP_USER, bi_username);
         SET_REPORT_OBJECT_PROPERTY (repid, BIP_PASSWORD, bi_password);

         -- Ask BIP to run the report.
          v_rep := RUN_REPORT_OBJECT(repid);

         /* DO SOMETHING ABOUT CHECKING STATUS HERE

          The call to BIP is asynchronous.  Therefore, it will be necessary to check on its status periodically if
         notifying the user is desired. The report_object_status built-in can be used for that check.
          Likely a timer would be used to periodically check the status.  Using a loop is not recommended.

         Example:  rep_status := report_object_status(v_rep);

         */
EXCEPTION
         WHEN ora_java.exception_thrown THEN
           ex := Exception_.new(ora_java.last_exception);
         BEGIN
             ex_msg := Exception_.toString(ex);
             message('Java Exception occurred: ' || ex_msg);
             ora_java.clear_exception;
         EXCEPTION
             WHEN ora_java.java_error THEN
                        message('Unable to call out to Java, ' || ora_java.last_error);
         END;
 END;
```

Information about using BI-Publisher can be found in the BI-Publisher documentation.
https://docs.oracle.com/middleware/12213/bip

Forms Websocket Java Script Integration (WJSI)

This feature requires the use of Eclipse/Jetty version 9.4.5, a third party Java jar file. This Java jar file must signed with a trusted and known certificate. Refer to the Java documentation for more information about signing Java jar files. The needed Jetty file can be downloaded from here:

http://central.maven.org/maven2/org/eclipse/jetty/aggregate/jetty-all/9.4.5.v20170502/jetty-all-9.4.5.v20170502-uber.jar

Using the Eclipse/Jetty server, a running Forms application can communicate to an HTML page through the Jetty server. The Jetty server is an extremely lightweight application server that has built-in support for Websocket technology and hosts a special intermediate application that assists in the communication between browser and Forms application. This lightweight server is delivered to the user's machine during application startup.

The implementation of this feature allows for the reuse of most existing Forms Java Script Integration code. To convert an existing application's Java Script integration do the following:

- » Locate websocketJSI.pll which is provided in the \forms directory and generate it into a PLX.
- » In the Form Builder, open an existing module and open websocketJSI.olb (found in the \forms directory).
- » Copy or subclass the WEBSOCKETJSI group found in the Object Library into the application.
- » Attach websocketJSI.pll to the module.
- » Perform a Find/Replace PL/SQL in the application. Search for *web.javascript* and replace it with *websocketJSI.javascript* Here is an example:

    Before:

        web.javascript_eval_expr(:block1.JSEXPR);

    After:

        websocketJSI.javascript_eval_expr(:block1.JSEXPR);

- » Generate a new FMX.

In order for the HTML page to send and receive data, it must include a reference to frmwebsocketjsi.js. This file is provided in the \forms\java directory. This file should not be relocated or altered. In the <HEAD> of the HTML page add the following:

        <script type="text/javascript" src="/forms/java/frmwebsocketjsi.js"></script>

Applet parameters associated with this feature are listed in the Applet parameter and Environment Variable tables later in this document.

More information on how to use this feature will be provided in a separate paper and in the Forms Deployment Guide.

System Events

System Events allow developers to develop applications that can react to actions that occur relative to the running application. In order to use System Events, you must create the desired event object in your form. Only one event per type can be included in any one form.
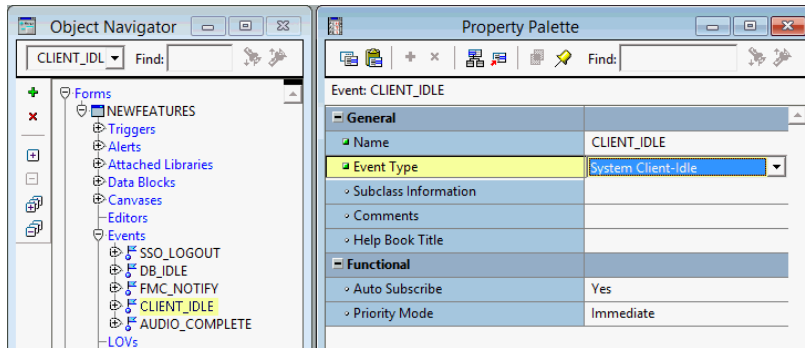

Figure 1 : Creating a System Event in the Form Builder

The new System Events include the following.

Client-Idle (SYSTEM_CLIENT_IDLE)

This event will be raised when the end-user has not performed any actions in the associated application session for the amount of time configured. The maximum idle time will be configurable using either a new applet parameter idleTimeout or using a new SET_APPLICATION_PROPERTY argument CLIENT_IDLE_TIME. Setting the value to 0 will disable idle time monitoring. The configured values will be in seconds. The values set for this event may not exactly represent the time specified. For example, an idle time setting of 60 seconds may not immediately raise the event until 61 seconds have elapsed. Therefore, this event should not be used for critical time conditions. Also, idle time will be ignored while the running form is displaying modal content (e.g. Alert or error dialogs, etc). Here is an example of how to use the SYSTEM_CLIENT_IDLE event.

First, either set the Forms applet parameter idleTimeout (in seconds) or set the application property CLIENT_IDLE_TIME (in seconds) to enable the monitoring of client interactivity idle time in the form.

```
-- Set CLIENT_IDLE_TIME to 10 minutes
SET_APPLICATION_PROPERTY (CLIENT_IDLE_TIME, 600);
```

Next, in a WHEN-EVENT-RAISED trigger add something like the following.

```
DECLARE
    al_button Number;
BEGIN
    If :SYSTEM.LAST_EVENT = 'SYSTEM_CLIENT_IDLE' Then
            al_button := SHOW_ALERT ('CLIENT_IDLE_ALERT');
    End if;
END;
```

DB -Idle (SYSTEM_DB_IDLE)

This event will be raised when database interactions have not occurred in the associated application session for the amount of time configured. The maximum idle time will be configurable using either a new environment variable FORMS_DB_IDLE_TIME or using a new SET_APPLICATION_PROPERTY argument DB_IDLE_TIME. Setting the value to 0 will disable idle time monitoring. Once this event is raised, it will not reset/restart until another database action occurs. To enable continued monitoring, set DB_IDLE_REPEAT to TRUE (SET_APPLICATION_PROPERTY (DB_IDLE_REPEAT, PROPERTY_TRUE). This behavior differs from CLIENT_IDLE, which monitors continually once enabled. The configured values will be in seconds. The values set for this event may not exactly represent the time specified. For example, an idle time setting of 60 seconds may not immediately raise the event until 61 seconds have elapsed. Therefore, this event should not be used for critical time conditions. Idle time monitoring will not begin until a value has been set for DB_IDLE_TIME and the first database action has occurred (e.g. database query, etc). Here is an example of how to use the SYSTEM_DB_IDLE.

First, either set the Forms environment variable FORMS_DB_IDLE_TIME (in seconds) or set the application property DB_IDLE_TIME (in seconds) to enable the monitoring of database interactivity idle time in the form.

Next, in a WHEN-EVENT-RAISED trigger add something like the following:

```
BEGIN
    If :SYSTEM.LAST_EVENT = 'SYSTEM_DB_IDLE' Then
            -- Turn off DB Idle monitor so the logout action doesn't
            -- cause the idle monitor to fire again. Then  logout.
            SET_APPLICATION_PROPERTY (DB_IDLE_TIME, 0);
            Logout;
END;
```

Single-Sign-Off (SYSTEM_SINGLE_SIGN_OFF)

This event will be raised when single sign-off has occurred. The application developer can decide how the application should respond to a logout condition. The Forms Server will not be notified of this event until the next exchange from client to server (e.g. heartBeat, maxEventWait, user action, etc). Single Sign-out features are not available when using Java Web Start or the Forms Standalone Launcher (FSAL). Here is an example of how to use SYSTEM_SINGLE_SIGN_OFF.

```
If :SYSTEM.LAST_EVENT = 'SYSTEM_SINGLE_SIGN_OFF' Then
    EXIT_FORM (NO_VALIDATE);
End if;
```

EM Notification (SYSTEM_NOTIFICATION)

This event will be raised when a notification has been received from Fusion Middleware Control (FMC). FMC will be able to send one of five notification levels to any or all of the running Forms sessions from the Forms User Sessions page. The application developer will code the desired action for each notification level. Notification levels can be determined using the new System variable :SYSTEM.NOTIFICATION_VALUE. Here is an example of how to use the SYSTEM_NOTIFICATION event.

```
DECLARE
    notification_level    varchar2(1);
BEGIN
    -- NOTIFICATION_VALUE is the value/number sent from Fusion Middleware Control
    notification_level := :SYSTEM.NOTIFICATION_VALUE;
    If :SYSTEM.LAST_EVENT = 'SYSTEM_NOTIFICATION' Then
     CASE notification_level
            -- Each message should be customized as needed.
            -- Customize these actions as needed.
            WHEN '1' THEN MESSAGE ('Received Notification #1 from Fusion Middleware Control.');
            WHEN '2' THEN MESSAGE ('Received Notification #2 from Fusion Middleware Control.');
            WHEN '3' THEN MESSAGE ('Received Notification #3 from Fusion Middleware Control.');
            WHEN '4' THEN MESSAGE ('Received Notification #4 from Fusion Middleware Control.');
            WHEN '5' THEN MESSAGE ('Received Notification #5 from Fusion Middleware Control.');
      END CASE;
    End if;
END;
```

Media Completion (SYSTEM_MEDIA_COMPLETION)

This event will be raised when a media (audio) file, started with the PLAY_AUDIO built-in, has reached the end of its play time. This will allow developers to programmatically react to this completion. Here is an example of how to use the SYSTEM_MEDIA_COMPLETION event.

```
If :SYSTEM.LAST_EVENT = 'SYSTEM_MEDIA_COMPLETION' Then
    MESSAGE ('Thank you for playing this audio file');
End if;
```

Audio Playback

Oracle Forms application will now be able to play audio files. This feature is not fully supported when using early versions of Java 7 on the client tier. Therefore, using Java 8 is recommended. The feature provides a new built-in called PLAY_AUDIO and will offer various arguments. Valid arguments include the following:

- » <audio file name>
- » *ACTION_PAUSE*
- » *ACTION_STOP*
- » *ACTION_RESUME*
- » *ACTION_TOGGLE_MUTE*

Audio files will be accessible via a properly signed jar file (recommended) or URL. Using a URL may result in Java security warnings being presented to the end-user. Audio files will be stored on the server's local file system (e.g. middle tier); similar to how images for iconic buttons might be stored. Supported formats will include the following:

» MP3
» AIFF containing uncompressed PCM
» WAV containing uncompressed PCM
» MPEG-4 multimedia container with Advanced Audio Coding (AAC) audio

-- Begin playing the welcome audio file.
PLAY_AUDIO ('welcome.mp3');

Single Sign-out

Support for single sign-out and its detection are included with these variations; on demand (programmatic), on-exit, and a new System Event (for detection of logout occurrences). The System Event will allow applications to react to a single sign-out occurrence or explicit sign-out when commanded. Single Sign-out features are not supported when using Java Web Start or the Forms Standalone Launcher (FSAL).

On Demand

On demand single sign-out provides a new built-in, LOGOUT_SSO, which allows the developer to programmatically cause single sign-out to occur. Executing this built-in will cause a new web browser window/tab to open briefly and call the SSO logout API, similar to what would occur in a full SSO partner application.

On-Exit

On-exit single sign-out is activated using a new applet parameter, ssoLogout. Setting this parameter to TRUE in the Forms Web Configuration (formsweb.cfg) will cause cleanly exiting forms to terminate the current SSO session, if one exists. The applet parameter ssoLogoutRedirect can be used to redirect the browser to a target URL upon successful SSO logout. The ssoLogoutRedirect parameter only works when used in conjunction with ssoLogout and when an SSO session exists.

Single Sign-out Event

The Single Sign-out event, described in detail above, allows developers to design applications that can react to a single sign-out action occurring.

Client Deployment

Traditional Oracle Forms web applications ran embedded in a parent web browser, using the Java Plug-in. HTML sent to the client from the Forms Servlet included the appropriate applet tags, which caused the Java Plug-in to load and launch the Forms applet. Unfortunately, this method of deployment came with problems and limitations. Some of those limitations included insufficient display space, accidental navigation away from the running Forms application, extra overhead of supporting the browser, limited support for the Java Plug-in, and compatibility issues between browser version and Plug-in version.

New client deployment options include Java Web Start, embedded JNLP, and Standalone. Traditional HTML will continue to be supported and configured as the default. Configuration examples are provided in the Forms Web Configuration (e.g. formsweb.cfg).

Embedded JNLP

Embedded JNLP is a hybrid of traditional HTML and Java Web Start. As with traditional HTML, the Forms Servlet delivers the needed HTML file for starting the applet. However, in the case of embedded JNLP, parameters specific to the Oracle Forms applet are contained within a base64 encoded entry (the JNLP contents). This entry is then surrounded by the needed HTML code to make up a surrounding web page. Using this option offers several advantages. Those include familiar look and feel to traditional HTML and ability to use some JNLP features in addition to HTML features. Embedded JNLP also supports the use of Java Script integration and integration with single sign-on.

Java Web Start

Java Web Start allows users to run Oracle Forms applications without having a parent web browser. Although a browser may be used to initially obtain the application's Java Web Start launcher file (.jnlp), the browser is not responsible for hosting the application and can be closed after the application has been started. When a browser is used to request the Java Web Start configured application, Single Sign-on can be used to protect this download and login to the application. The user can choose to launch the application subsequently from the browser or use the downloaded Java Web Start launcher stored in the Java Cache Viewer (only if not SSO protected). If the application is SSO protected, all requests to run this application must be made from the browser. A desktop shortcut can also be created, however the same restriction will apply regarding SSO. Regardless of whether or not SSO is used, by always calling the application from the browser, users will be assured of having the latest configuration for the application. So, calling from a browser is recommended.

Because Java Web Start runs the application without a browser, features that are browser dependent are not supported in this mode. These will include Java Script, and Single sign-off. Single sign-on is supported when calling from a browser, as mentioned above.

Forms Standalone Launcher (FSAL)

The Forms Standalone Launcher, also known as FSAL offers a modern style of client/server. FSAL allows users to run Oracle Forms 12c applications similarly to how a typical Java application might be run from their local machine. With FSAL, no browser is used to launch the application. All that is needed is a certified Java SE installation on the end-user machine. The Java installation needs to be either the Java Development Kit (JDK) or Java Runtime Environment (JRE) or Server JRE.

Information on how to use FSAL can be accessed from a running server using this URL

http://server:port/forms/html/fsal.htm

Because FSAL runs the application without a browser, features that are browser dependent are not supported in this mode. These will include Java Script, Single sign-off, and Single sign-on. FSAL should only be used for internal applications where the environment is fully trusted.

To run SSL protected applications, it may be necessary to import the appropriate SSL certificate into the key store of the Java installation being used to run FSAL. Refer to the Java documentation regarding the "keytool" utility for details on how to import an SSL certificate into the key store.

https://docs.oracle.com/javase/8/docs/technotes/tools

If the user access the network through a proxy, it may be necessary to use various proxy settings to allow proper access to the server. Refer to the Java documentation for information on how use proxy settings on the command line.

https://docs.oracle.com/javase/8/docs/technotes/guides/net/proxies.html

Custom Text on Logon and Password Change Dialogs

Oracle Forms provides a default logon feature, as well as a default password change feature. Both present the user with a dialog box that contains no informative text other than the field labels. This feature allows for a custom message to be included in the dialog. To customize these dialogs set the desired environment variable and include the text to be displayed. The environment variables FORMS_LOGON_HINT and FORMS_CHANGE_PASSWORD_HINT are set in the Forms Environment Configuration (default.env or your custom .env). The text or "hint" is limited to 255 single-byte characters. For example:

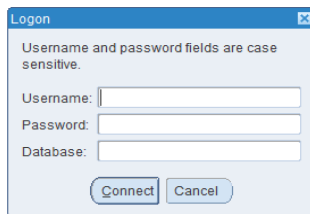*FORMS_LOGON_HINT=Username and password fields are case sensitive.*

Figure 2 : FORMS_LOGON_HINT shown.

Tab Labels with Images

Tab canvas labels can now display images. Images must be of a common web format (e.g. gif, jpg, png). Images can be added declaratively in the Form Builder or at runtime in the application pl/sql. Choosing images programmatically is easy and can give the application a more dynamic appearance since the displayed image can be changed at nearly any time.

SET_TAB_PAGE_PROPERTY (<tab page name>, ICON_NAME, <icon name>);

The image can be include in a signed jar file or accessed via a URL (virtual path), however using a signed jar file is recommended and will avoid the displaying of Java security warnings. Image size should be limited to 16x16 pixels. Larger size images can be used; however this will cause the tab label size to increase to match the size of the image. This may not be desirable.

Figure 3 : With tab label images.

Record Counter

The new SET_APPLICATION_PROPERTY argument, RECORD_COUNT_DISPLAYED, allows developers to programmatically control when the message bar's record counter is displayed or not. This can often be helpful when the current form is not showing content that contains an obvious record, like a custom logon screen. By hiding the record counter, users are less likely to be confused by what is presented.

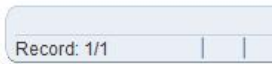*SET_APPLICATION_PROPERTY (RECORD_COUNT_DISPLAYED, PROPERTY_FALSE);*
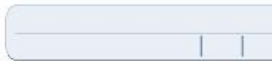

Figure 4 : Record counter displayed.

Figure 5 : Record counter hidden.

Row Banding

Row banding allows multi-record blocks to be striped with alternating colors. This can help to make each row more visible to the user. It can also help to improve the overall appearance of the application. The stripe colors are derived from the runtime colorscheme's pinstripe colors. Custom colorschemes can be created if the provided schemes are not desired. Refer to "Customer Color Scheme" also found in this document.

Enabling or disabling row banding can be accomplished at design-time on the Property Palette for the desired block or item, but will not be visible until runtime. Refer to the Row Banding Frequency property. The value of Row Banding Frequency represents how often the color should alternate. For example a setting of 2 would result in changing the color of ever other row beginning with the second. Setting to 0 or 1 at the block level would result in no rows being striped. In cases where you do not want an item in the block included in the striping, simply set the Row Banding Frequency at the item level to 1. Setting the item level value to 0 implies that it should inherit from the block setting. Row Banding cannot be programmatically set or unset at runtime. The banding is only visible at runtime.

Figure 6 : Row banding shown with colorscheme SWAN.

Gradient Canvas Colors

Gradient canvas colors are set at design-time, but are not visible until runtime. The starting color is based on the background color set for the canvas. If no color is set, the color implemented by the runtime colorscheme will be used. At design-time, the developer can choose the starting position from where the gradient flows (None, Left, Top, Right, or Bottom). The difference in color from its starting point to its end point is set in the runtime configuration Registry.dat (aka "Fonts and Icon Mapping") using these settings:

- » default.gradient.redDelta
- » default.gradient.greenDelta
- » default.gradient.blueDelta

These settings only apply when the canvas background color has a specified value. Each specifies an integer value representing how much to add (or subtract) from the start value to reach the end color.   The color range assumes a value between 0 and 255. Thus, if the start color had an RGB value of 150,150,150 and each of the Delta properties was set to 50, then the end color would be 200,200,200. Similarly, the properties can have negative values. As an example, if all were set to -50, the end color would be 100,100,100. Setting the red, green, blue Delta values to 0 or unset would result in no change to the color. However, if left unset and the canvas background color is unset (<Unspecified>) the colorscheme colors are used and the gradient will go from OLAFDark to OLAFVeryLight (as seen below left).
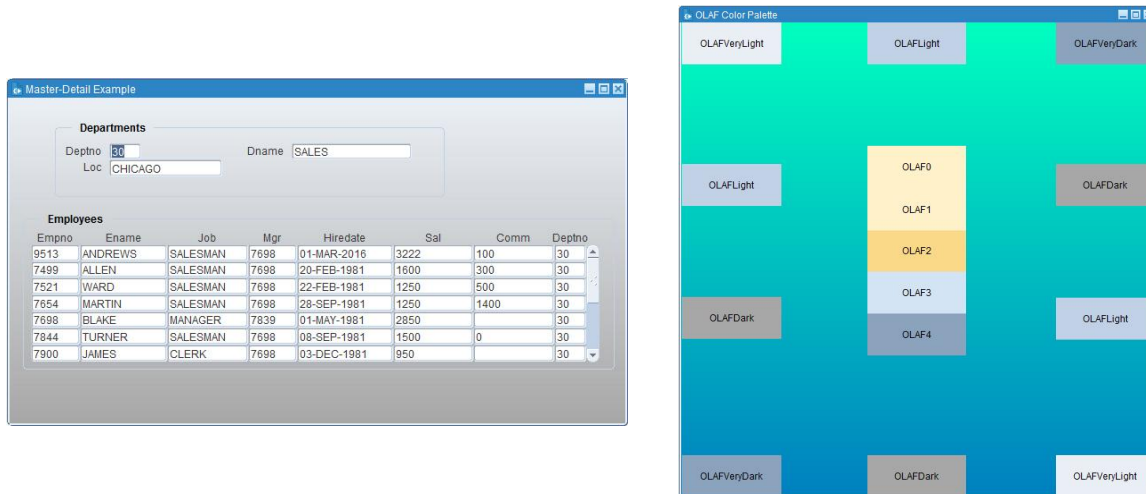

Figure 7 : Gradient colored content canvas.

Using this feature in cases where a hidden stacked canvas that is larger than the main canvas and is shown later in the session's life is not recommended. This is because the gradient fill pattern is drawn only at initial startup. Altering the view area at runtime may result in an incomplete gradient flow. It should be noted that using a gradient in both the content and stacked canvas will result in color transitioning not being in sync with each other. This is because the coloring on one canvas is unrelated to the coloring on another. Additional usage and limitation information can be found in the Form Builder Help.

Smartbar (toolbar) Size

In previous versions, Forms toolbars associated with menus displayed 16x16 pixel icons and were contained in buttons of the approximate same size. Attempts to use larger images would clip the image and the button size

would remain the same. With the new applet parameter, smartBarHeight the smartbar size can be increased to either a fixed, but larger size or changed dynamically based on the value of the applet parameter clientDPI. If the provided images are smaller than the containing button, the image will be scaled up to fill the button. If the image is larger than the button, it will be clipped. Supported sizes will be 16x16, 32x32, and 48 x 48 pixels. Valid values for this parameter will be medium, large, or dynamic. Any other value or unset will be treated as the default behavior, using the small (16x16) buttons.
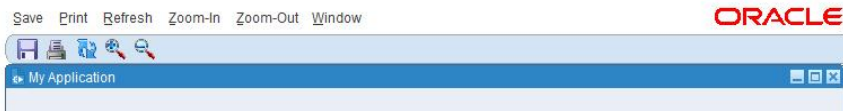
Figure 8 : smartbarHeight un-set and using 16x16 pixel images.

Figure 9 : smartbarHeight set to medium and using 32x32 pixel images.

Figure 10 : smartbarHeight set to large and using 48x48 pixel images.

Improved Canvas Background Image Support

In earlier versions, Forms performed its own image compression in order to reduce an image's content size and reduce network traffic associated with sending it to the user's machine. It is now possible to bypass image processing for images displayed as backgrounds on a canvas. This feature is only supported for GIF, JPEG, and PNG image formats. To use this feature images are loaded programmatically using SET_CANVAS_PROPERTY or SET_TAB_PAGE_PROPERTY. Images are loaded using the new BACKGROUND_IMAGE property. The position of the image cannot be moved nor can more than one image be displayed. The images can be stored on the server's file system or accessed via a virtual path (e.g. URL, jar, etc). Here are some examples:

Relative URL:

SET_CANVAS_PROPERTY ('CANVAS1', BACKGROUND_IMAGE, 'URL:/media/team-usa.jpg');

Fully qualified URL:

SET_CANVAS_PROPERTY ('CANVAS1', BACKGROUND_IMAGE, 'URL:http://example.com/media/team-usa.jpg');

Access from jar file or codebase:

SET_CANVAS_PROPERTY ('CANVAS1', BACKGROUND_IMAGE, 'URL:team-usa.jpg');

Access from server's file system:

SET_CANVAS_PROPERTY ('CANVAS1', BACKGROUND_IMAGE, 'C:\graphics\team-usa.jpg');
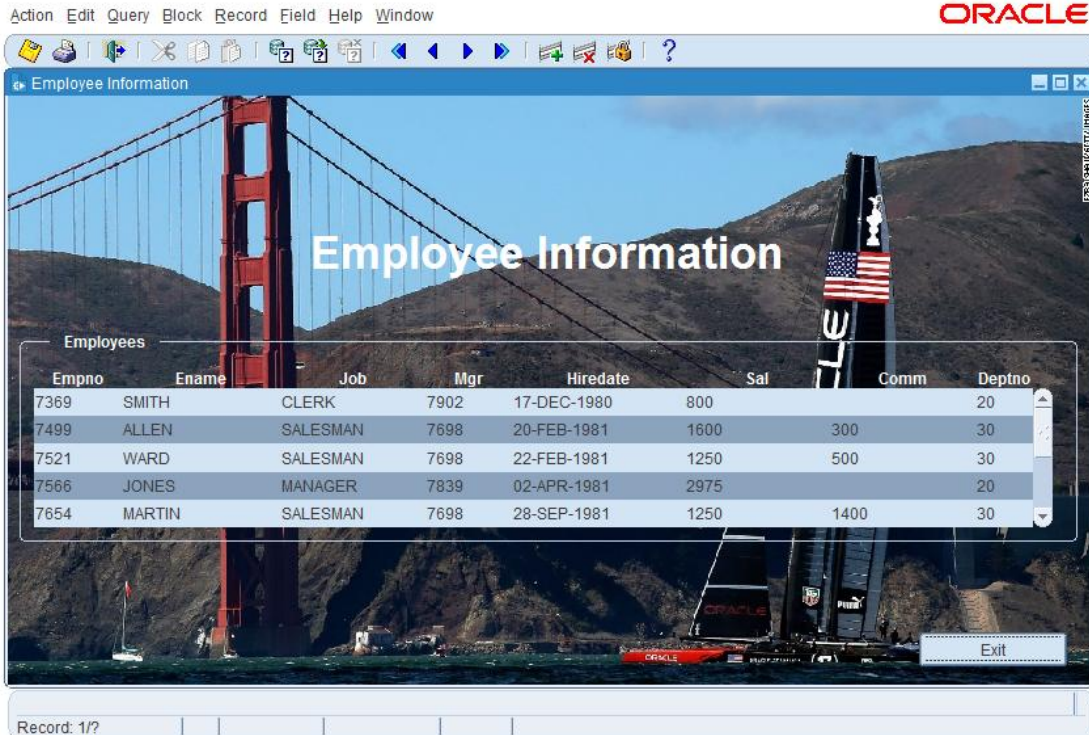
Figure 11 - Canvas background image.

Image Item Sizing Style Property

Image items offer the following properties for Sizing Style:

- » Crop: Displays only the portion of the full image that fits in the display rectangle.
- » Adjust: Scales the image to fit within the display rectangle. Size ratio is maintained.
- » Fill: Image size scaled to fit the image item fully (height and width). Size ratio is not maintained.

Item Level Mouse Cursor (Pointer)

The SET_ITEM_PROPERTY argument, CURSOR_STYLE is used to change the mouse cursor (pointer) while hovering over an item. In earlier versions, it was only possible to set the cursor style at the application level. This resulted in the cursor appearing the same regardless of where the mouse was pointed. The desired cursor can be selected at design-time as an Item Property or at runtime in the application pl/sql. Possible cursors are:

- » ARROW
- » BUSY
- » CROSSHAIR
- » DEFAULT
- » INSERTION
- » HAND
- » MOVE

Resizing pointers:

> » NRESIZE
> » SRESIZE
> » ERESIZE
> » WRESIZE
> » NERESIZE
> » NWRESIZE
> » SERESIZE
> » SWRESIZE

SET_ITEM_PROPERTY ('block1.displayItem1', CURSOR_STYLE, 'HAND');

Jump to Oracle.com

Figure 12 : Mouse HAND cursor.

Setting Max Event Wait Programmatically

The applet parameter maxEventWait was added in a previous release to assist in the Forms integration support with Streams (Advanced Queuing). This parameter creates a HEARTBEAT-like ping to the server that can be set in seconds rather than minutes. It can be helpful when the need to query the status of an asynchronous task is being performed . One issue with using this applet parameter is that it can result in a significant increase in network traffic and degrade application performance. The Application Property MAX_EVENT_WAIT can now be programmatically controlled. This allows developers to enable, disable, and change MAX_EVENT_WAIT as needed. Its value is set in seconds in the application's PL/SQL. Setting the value to 0 will return control back to the HEARTBEAT and it previously configured setting, essentially disabling MAX_EVENT_WAIT.

SET_APPLICATION_PROPERTY (MAX_EVENT_WAIT, 10);

Client IP Address

In previous versions, the only way to obtain the client IP address was either with a custom Java Bean or by WebUtil enabling the application. A new application property USER_IP_ADDRESS can be used to get the user IP address without the need for Java Beans or WebUtil.

v_clientIP := GET_APPLICATION_PROPERTY (USER_IP_ADDRESS) ;

Form Query-Only Mode

The new GET_FORM_PROPERTY, FORM_QUERY_ONLY returns TRUE/FALSE depending on how a form was opened/started.

```
Declare
        x  varchar2(6);
Begin
        x := GET_FORM_PROPERTY('myForm',FORM_QUERY_STATUS);
        message (x);
End;
```

Oracle Diagnostic Logging and Forms MESSAGE Built-in

Oracle Diagnostic Logging (ODL) was introduced in Fusion Middleware 11g. This diagnostic logging was a significant improvement over traditional logging methods. In Forms 12c, it is now possible to programmatically send entries to ODL. This can be helpful for both troubleshoot and auditing. One of four new MESSAGE built-in arguments can be included. When included, output will be sent to ODL rather than the user interface. Possible arguments are as follows:

- ODL_DEBUG
  Generates FRM-91930 in ODL followed by the text provided in the "message_string".

- ODL_NOTIFICATION
  Generates FRM-91934 in ODL followed by the text provided in the "message_string".

- ODL_WARNING
  Generates FRM-91935 in ODL followed by the text provided in the "message_string".

- ODL_ERROR
  Generates FRM-91936 in ODL followed by the text provided in the "message_string".

The FRM error code can be used to easily find the desired entry in the log. Here is an example.

```
Declare
        usr_name  varchar2(20);
Begin
        usr_name := GET_APPLICATION_PROPERTY(USERNAME);
        MESSAGE (usr_name || ' attempted to insert a record.', ODL_NOTIFICATION);
END;
```

JVM Controller

The JVM Controller is used to manage the extra memory used by Forms applications. Specifically, the JVM Controller manages memory used by Java calls made by the Forms Runtime. Java is used by the Forms Runtime when calls are made to Oracle Reports, Oracle BI-Publisher, or Java that has been imported into a Forms

application. In previous versions, the JVM Controller was either on or off. It can now be configured to work in the most efficient configuration appropriate for specific conditions.

Child JVM Management

Children JVMs are spawned as needed. When these extra JVMs are no longer needed, the Controller can be configured to terminate them.

» OFF (default)
  JVMs would not be removed automatically by JVM Controller.
» AGGRESSIVE
  The frequency of removing Child JVMs is at its highest.
» MODERATE
  The frequency of removing Child JVMs is lower than AGGRESSIVE.
» CONSERVATIVE
  The frequency of removing Child JVMs is lower than the previous two options.

Load Balancing

Load balancing options include the following:

» Random (default)
» LeastLoadedFirst
» RoundRobin

Logging

The logging feature of the JVM Controller is helpful when troubleshooting problems, understanding load patterns, determining when a controller is stopped or started, and a variety of other data points. In earlier version it was only possible to enable or disable logging. Beginning in version 12, logging now can be set to one of six values. Each value will record a different level of granularity. Possible values include:

» Off
» Debug
» Warn
» Error
» Crit
» Info

More details about using and configuring the JVM Controller can be found in the Forms Deployment Guide.

Fusion Middleware Control Advanced Configuration

Advanced Configuration allows for management of configuration and template files not previously accessible in Fusion Middleware Control (FMC). File types include Forms client templates, webutil.cfg, and key binding/mapping files.
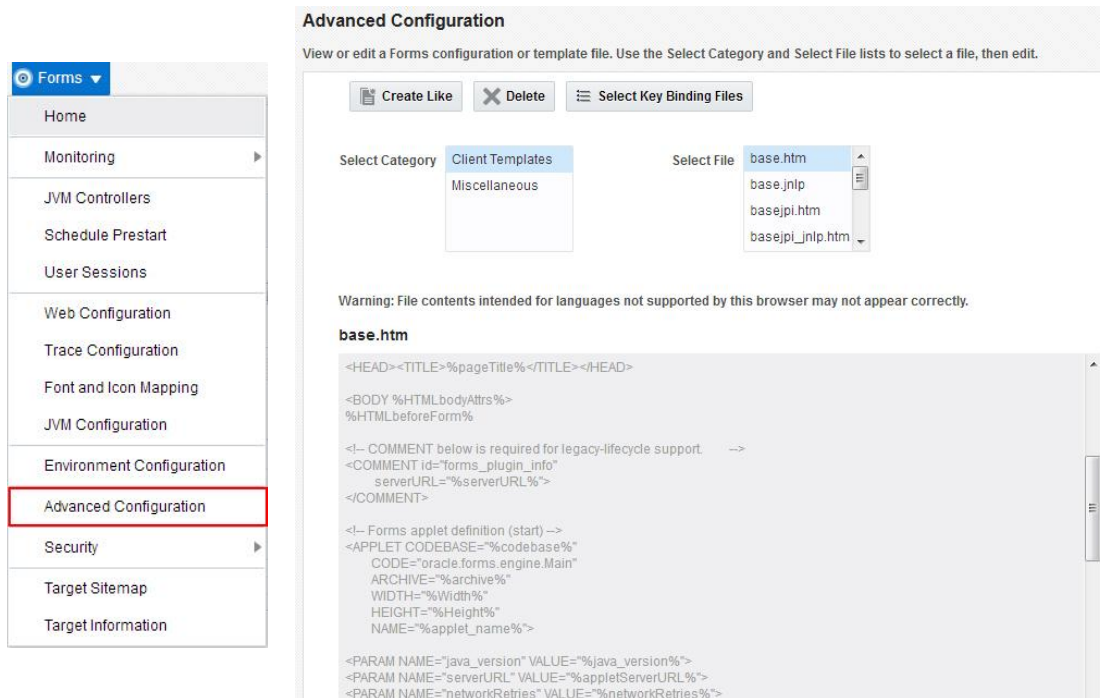

Figure 13 - Forms Advanced FMW Configuration

Custom Color Schemes

Oracle Forms delivers nine (9) runtime color schemes. The desired color scheme is set as a Web Configuration parameter (in formsweb.cfg). The parameter "colorScheme" is set to "SWAN" by default in Forms 12.2.1. It is now possible to create custom color schemes. The desired values of the custom color scheme are set in the Forms Font and Icons Mapping configuration (Registry.dat). This is accessible in Fusion Middleware Control. An example is provided in the installation, but not likely appropriate for a production deployment. The example has been named "sample". To enable this example, simply set the Web Configuration parameter customColorScheme=sample. More information can be found in the Forms Deployment Guide.

New Applet/Servlet Parameters

With new applet/servlet parameters, applications can gain new behaviors and appearances without making risky code changes. More information about applet parameters and how to use them can be found in the Oracle Forms Deployment Guide.

Applet Parameter	Description
idleTimeout	Specifies how much idle time (in seconds) can elapse before a SYSTEM_CLIENT_IDLE event will fire.
ssoLogout	Specifies if the session should be SSO logged out when a successful Forms logout occurs. If the application does not exit cleanly, SSO Logout may not occur. Default is FALSE.
ssoLogoutRedirect	Used in conjunction with ssoLogout, sets the URL used for browser redirection after a successful ssoLogout.
logoutTargetURLParamname	Specifies an alternative OAM logout URL other than the default.
java_version	Specified which Java version is to be used on the client environment. Default value is 1.7+
customColorScheme	Specifies the name of the custom colorScheme created in Registry.dat to be used. Setting this will override the colorScheme parameter.
webstart	Specifies whether or not Java Web Start should be enabled. Set webstart=enabled to enable. See example in Web Configuration (formsweb.cfg).
webstart_codebase	Specifies the URL to the application Codebase. This may be required when using Java Web Start and the Forms managed server (WLS) is behind a proxy server. Example: webstart_codebase=http://proxy_server:port/forms/java
alwaysOnTop	Specifies if the Forms MDI window (separateFrame=true, Java Web Start, Standalone Launcher) will remain on top of all other open windows. This is not supported with Java Web Start in version 12.2.1.0. Default is FALSE.
centerOnStartup	Specifies if the Forms MDI window (separateFrame=true, Java Web Start, Standalone Launcher) will start centered on the screen. This is not supported with Java Web Start in version 12.2.1.0. Default is FALSE.
isResizable	Specifies if the Forms MDI window (separateFrame=true, Java Web Start, Standalone Launcher) is resizable by the end-user. Default is TRUE.
dynamicLayout	Specifies whether or not the Forms applet should redraw the parent window (and notify the server) while resizing the that window. Enabling causes an

	increase of network traffic while resizing is in progress. Default is TRUE.
hideActivityBar	Specifies whether or not the Forms activity bar (also known as progress bar) should be hidden. Default is FALSE.
WebUtilVersion	Specifies whether or not to display the WebUtil version in the client side console. Default is FALSE.
hideAppletVersion	Specifies whether or not to display the Forms applet version in the client side console. Default is FALSE.
hideClientExceptions	Specifies if Java exceptions should not be displayed to the end-user in both the Java console and error message dialog. Default is FALSE.
fsalcheck	Specifies whether to perform checksum comparison of Forms stand-alone app launcher or not. When enabled, it triggers a comparison to be made on server. The checksum of FSAL on the client machine will be compared with the checksum of FSAL archived on the server. Default is TRUE
consoleUseRegistryFont	Specifies whether or not the application's console (aka message bar) uses the default font attributes specified in "Font and Icon Mapping" (i.e. Registry.dat).
oam_redirect_root	Used in cases where mixed protocols are used between the end-user, OHS, Forms, and OAM. Valid values include HTTP, HTTPS, or a fully qualified URL to the server hosting Forms (e.g. <protocol>://<client-visible-host-name.domain>:<client-visible-port>)
smartBarHeight	Specifies the desired Smartbar menu size to be used at runtime. Valid values are *medium* (32x32), *large* (48x48), *dynamic* (adapts based on value of clientDPI). Icons larger than the button size will be clipped. Icons smaller than the button size will scale up to fill the button. The default (e.g. unset) or setting to any value other than those listed here will result in buttons fixed to contain icons that are 16x16. These buttons will not scale if the value of clientDPI is changed.
smoothScalingMaxZoom	Smooth scaling often produces a significant improvement in image quality when the scaling factor is significantly lower than 100%. This improvement decreases as scale increases, and seems unnoticeable when the scaling factor is significantly greater than 100%. The downside of smooth scaling is that a large scaling factor may produce an OutOfMemoryError [which will leave the image item blank]; the probability increases as the scaling factor increases. Smooth scaling is by default not enabled. This parameter specifies the tradeoff between image quality and the probability of an OutOfMemoryError. It specifies the maximum zoom_percent for which smooth scaling will be used. For example, set smoothScalingMaxZoom=200. This would specify that smooth scaling should be used for all downscaling, and for upscaling up to a factor of 2 (200%). Note that smoothScalingMaxZoom can be specified in a URL. An end user might want to lower it or disable it

	altogether [by specifying smoothScalingMaxZoom=0] if an OutOfMemoryError occurred. [Even if the java console is not visible, a blank image item would suggest that an OutOfMemoryError had occurred.]
websocketJSIServerTimeout	The time, in minutes that the WJSI server should continue to run while idle. After this time, FORMS_TIMEOUT will be honored.
websocketJSISessionTimeout	The time, in minutes the WJSI session should be permitted to remain idle before being terminated. The default is 5 minutes.
websocketJSILogging	Set to TRUE to enable WJSI logging. Output will be sent to the Java Console. Default is FALSE.
removeCommentLinesFromApplet	Removes comment lines from html/jnlp sent to client. This can improve startup performance and reduce the amount of content being sent to user. Set to TRUE to enable. Default is FALSE.
gzipCompressApplet	Setting to TRUE will cause the Forms Servlet to attempt compressing the html/jnlp content sent to the client. This assumes that the request coming from the client indicates support for compression in its HTTP Accept-Encoding header. Enabling this parameter can help to reduce application startup time. This setting may interfere with some testing tools. If such a problem occurs, simply disable this parameter while testing is being performed. Default is FALSE.

New Environment Variables

Environment variables for Oracle Forms are used to control and customize the Forms Runtime or design-time tools (e.g. Builder, Compiler, etc). For runtime. setting changes are made in the Environment Configuration (default.env or custom env file if used). More information about runtime environment variables and how to use them can be found in the Oracle Forms Deployment Guide.

Environment variables used by the design-time tools are set the operating system environment or on Microsoft Windows, set in the Windows Registry.

Environment Variable	Description	Applies To
FORMS_SELECT_ON_CLICK	Specifies whether or not the text in a field should be selected when using the mouse to enter the field.	Runtime
FORMS_LOGON_HINT	Specifies the text that should be displayed in the default logon dialog box. The total number of characters cannot exceed 255.	Runtime
FORMS_CHANGE_PASSWORD_HINT	Specifies the text that should be displayed in the default	Runtime

	logon dialog box. The total number of characters cannot exceed 255.	
FORMS_HTTP_PROXY_HOST	Specifies the HTTP enabled proxy host. Proxying may be necessary for calls to Oracle BI-Publisher, or when a form uses server-side Java (Imported Java) to make HTTP/HTTPS. If using the JVM Controller, refer to the Java documentation: https://docs.oracle.com/javase/8/docs/api/java/net/doc-files/net-properties.html	Runtime
FORMS_HTTP_PROXY_PORT	Specifies the HTTP enabled proxy port number.	Runtime
FORMS_HTTPS_PROXY_HOST	Specifies the HTTPS enabled proxy host. Proxying may be necessary for calls to Oracle BI-Publisher, or when a form uses server-side Java (Imported Java) to make HTTP/HTTPS requested call. If using the JVM Controller, refer to the Java documentation for information on how to use proxy settings: https://docs.oracle.com/javase/8/docs/api/java/net/doc-files/net-properties.html	Runtime
FORMS_HTTPS_PROXY_PORT	Specifies the HTTPS enabled proxy port.	Runtime
FORMS_PROXY_BYPASS	Specifies a list of hosts that should not be accessed using a proxy. Hosts in the list are separated by '|' character and individual host can include wildcard character '*'.	Runtime
FORMS_HIDE_LOGIN_RESET	Specifies whether or not the logon dialog should be displayed in the event a database logon fails when SSO is enabled.	Runtime
FORMS_LOGIN_DIALOG_BLANK	Specifies whether or not the logon dialog displayed after a failed database logon contains the username and database name.	Runtime
FORMS_DB_IDLE_TIME	Specifies the amount of time, in seconds before the Forms DB Idle Time System Event will fire.	Runtime
FORMS_EXTENDED_STRING	Specifies whether or not to support the DB 12.1+ MAX_STRING_SIZE=EXTENDED option. Valid values are TRUE/FASE	Runtime
FORMS_BI_SERVER	Specifies the host and port of the BI-Publisher server. (e.g. <protocol>://<BI-Publisher-host-name.domain>:<client-visible-port>) Using this variable allows application developers to use a relative path that	Runtime

	points to the BI-Publisher WSDL in the application code when integrating with BI-Publisher.	
FORMS_MAP_GIF_IMAGE_TO_NATIVE	Specifies if image items containing GIF images should bypass the Forms Multimedia processing layer. Bypassing the Multimedia processing layer will deliver uncompressed images to the user interface and will typically exhibit superior image quality when Native (a new Builder property for image items) is specified. However, network bandwidth will be increased and performance may be degraded. If the behavior of Native image items is desired for image items that specify an Image Format of GIF, this can be achieved without modifying or recompiling fmb files. Set the value of this environment variable to TRUE (or 1) to enable this feature. Set to FALSE (or 0) to disable. Default is FALSE.	Runtime
FORMS_MAP_JFIF_IMAGE_TO_NATIVE	Specifies if image items containing JFIF/JPEG/JPG images should bypass the Forms Multimedia processing layer. Bypassing the Multimedia processing layer will deliver uncompressed images to the user interface and will typically exhibit superior image quality when Native (a new Builder property for image items) is specified. However, network bandwidth will be increased and performance may be degraded. If the behavior of Native image items is desired for image items that specify an Image Format of JFIF/JPEG/JPG, this can be achieved without modifying or recompiling fmb files. Set the value of this environment variable to TRUE (or 1) to enable this feature. Set to FALSE (or 0) to disable. Default is FALSE.	Runtime
FORMS_WJSI_OVERRIDE_TIMEOUT	Time, in minutes (3-1440) that the Forms runtime process should wait before honoring FORMS_TIMEOUT while awaiting the return from a modal Java Script call using WJSI.	Runtime
FORMS_OBR_REMOVE_PATH	Specifies whether or not the full file system path should be included in the URL when using the Builder's One Button Run feature. This will be helpful when the Application Server URL is set to a remote node. Default is FALSE.	Builder
FORMS_ASSUME_MULTI_MONITOR	Specifies whether or not the Form Builder should retain	Builder

	window position information upon exiting. When set to TRUE, a window positioned on a secondary monitory when exiting will have its position on that monitor retained. If the secondary monitor is expected to not be available in the future, setting this variable to FALSE will cause window positions to not be retained and restarting the Builder will occur on the main display.	
FORMS_SHOW_REP_DIALOG	Specifies whether or not to display the Create Report dialog when creating a new report object in the Object Navigator. Setting to TRUE (1) will cause the dialog to be displayed. Setting to FALSE (0) or unset will prevent the dialog from appearing.	Builder

Record Manager

The Forms record manager has historically used the file system to store "archived" records (i.e. records retained by the Forms session but not currently in an active state). The amount of disk storage used can potentially be quite large. However, large-memory systems (such as Exalogic) often have relatively small local file systems, and performance will suffer if the record manager has to use a remote file system for archived records. Improvements to the Record Manager now allow it to store record data in local memory rather than the file system. The result is less disk activity, faster read/write times, and overall improved performance.

Administrator customizations are available, but intended for advanced users while troubleshooting. These customizations can also be helpful in cases where the available memory is below that which is required.

Refer to the Form Builder Help for more information.

Support for DATA_VALUE_IS_NULL Property

The new DATA_VALUE_IS_NULL property applies to the GET_ITEM_INSTANCE_PROPERTY and GET_ITEM_PROPERTY built-ins. This new argument will return 'TRUE' if the specified item or item instance is null and 'FALSE' otherwise. In the case of GET_ITEM_PROPERTY, the item instance in the block's current record is inspected. This is valuable when working with image items, as these items previously did not offer a method for determining if an image was present or not.

The DATA_VALUE_IS_NULL property can also be used for other item types, except button and Htree.

Oracle Platform Security Services (OPSS) Integration

In Forms versions prior to 12.2.1, integration with single sign-on (SSO) required that the LDAP server used by the authentication server be Oracle Internet Directory (OID). Remote Access Descriptor (RAD) data that was previously stored in OID can now be stored in the Oracle Platform Security Services (OPSS) repository. As a result, Forms no longer has a direct dependency on any LDAP server. Users can now choose any LDAP server that is compatible with Oracle Access Manager (OAM). This feature is not supported with Oracle Reports.
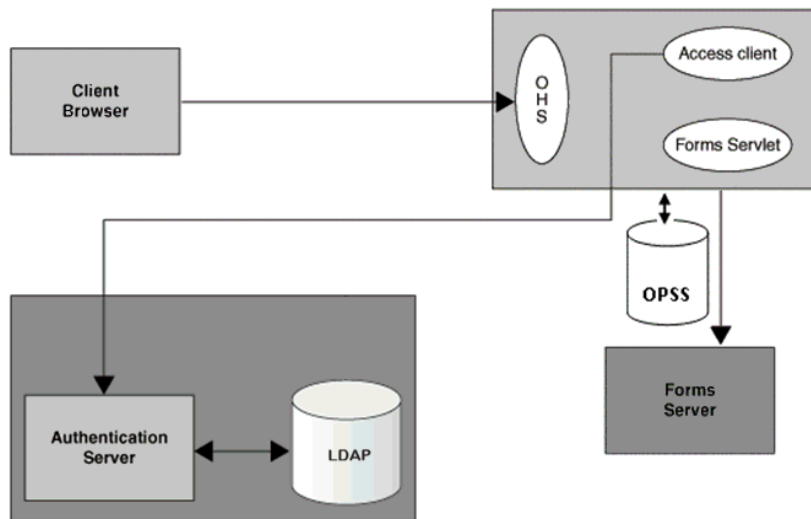
Figure 14 : Forms SSO and OPSS integration flow.

Remote Access Descriptor Administration

Remote Access Descriptor (RAD) data includes the database connection information required by Forms applications to connect to the database when integrated with SSO. These records can now be administered directly in Fusion Middleware Control (FMC). This administration can be accomplished in FMC, regardless of whether configured to use the legacy option that uses OID or using OPSS. Also included is a utility page that allows administrators to easily migrate data from OID to OPSS.
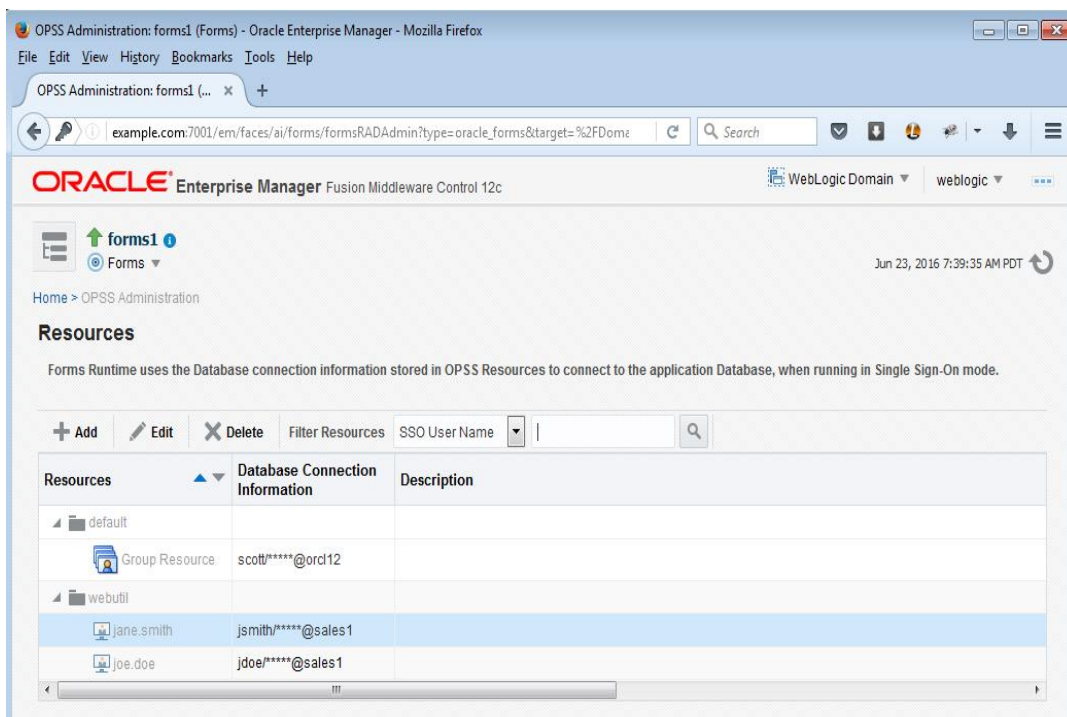
Figure 15 : OPSS RAD administration in FMC.

Forms Application Deployment Services

Forms Application Deployment Services (FADS) allows administrators or developers to package applications, deploy, configure, and store archived copies of the applications with the click of a button. Using the FADS web interface, you can check on the status of your deployments, deploy updated versions of your applications, delete no longer needed applications, and much more. A command line interface is also available, which may be helpful for Forms build integration and scripting automated deployment jobs where using a web interface may not be appropriate. FADS is optionally installed/configured into the WLS Administration Server, therefore it must be running in order to access it. Information on installation/configuration and usage can be found in the Forms Deployment Guide.

Once installed and configured, access the FADS web interface as follows:

    http://example.com:7001/fadsui

Figure 16 - Forms Application Deployment Services (FADS) dashboard.

Forms Servlet Test Mode

The Forms Servlet Test utility has always been helpful for troubleshooting, but was previously difficult and/or risky to enable because it required modifying the Forms J2EE deployment plan. Enabling this utility now simply requires setting allow_debug=testmode in the Forms Web Configuration (aka formsweb.cfg). This setting will continue to behave as it did in the past, however it will now also enable the Servlet Test utility too. Valid values for allow_debug are:

- » FALSE (default)
  Blocks servlet related debug commands and related logging.
- » TRUE
  Enables various debug functions exposed by Forms, as well as enabling Servlet specific commands. Servlet Admin test web page (/forms/frmservlet/admin) is not accessible.
- » TESTMODE
  Same as setting to TRUE, however allows access to the Servlet Admin test page when requested.

By default allow_debug is disabled (FALSE). It is recommended that this parameter only be set to TRUE or TESTMODE during troubleshooting. It should not be set to TRUE or TESTMODE unless explicitly being used by administrators during testing.

Forms Helper Script

The Oracle Forms Configuration Helper script (frmconfighelper) helps administrators perform complex post install, Forms configuration tasks in a much easier manner than if using the traditional tools. This utility includes the following functions.

- » enable_ohs
- » enable_sso
- » enable_sso_ssl
- » enable_webgate
- » create_machine
- » create_managed_server
- » deploy_app
- » update_app

Details about them can be found in the Appendix section of the Forms Deployment Guide and when running the script without passing any arguments.

Forms Builder Features

Builder Preferences

General Tab

» Hide PL/SQL Compile Dialog
When checked, causes the PL/SQL compilation results window to automatically close if no errors are detected. When unchecked, the compilation progress dialog will remain open until the user clicks on OK.

» Web Browser
Previously located on the Runtime tab, this preference has been moved to the General tab. The drop-down list can hold several browser locations. These are used to launch a form or display web content. Storing multiple browsers allows the developer to easily switch between browsers when testing.

» Compiler Output
Enter the full path to the directory where generated modules (fmx, mmx, plx) should be saved. The user must have read/write on the directory set in this field. This feature makes it easier to separate source code from generated modules.

» Auto Backup Time
This represents the number of minutes the Form Builder will wait before creating a backup of any recent changes. The number entered can include fractions of a minute (e.g. 2.5), however the value must be 1 or greater. These backups are intended for crash recovery only. In the unlikely event the Builder is unintentionally terminated, upon its next startup and the user attempting to reopen the previously opened module(s), the user will be prompted to either use the back or use the previously save version of the file. Backups will only be created for the current module and while the Builder is the active window. To disable Auto Backup, set the value to 0 or leave the field blank.

Runtime Tab

» Show URL Parameters
When checked, using the Builder's RunForm Button will display the complete URL in the browser. This includes configuration options and username/password details. This can be helpful when testing or troubleshooting. Because checking this box will show sensitive details in the URL, care should be taken when using this feature. When unchecked, details typically exposed in the URL will not be shown.

Module Converter

In a previous version, an XML conversion utility was provided which allowed for the conversion of Forms binary menu and form modules to an XML format and back to binary. This utility was provided as a command-line only tool. In this release, the utility is accessible directly from within the Builder. The XML conversion utility is now integrated with the previous binary to text converter. In addition to the necessary fields for choosing and converting to and from XML, the dialog includes check boxes to determine if overwriting an existing file should occur during translations and whether or not the dialog should remain open after a conversion completes successfully. This can be accessed by selecting File > Convert… from the Builder menu.
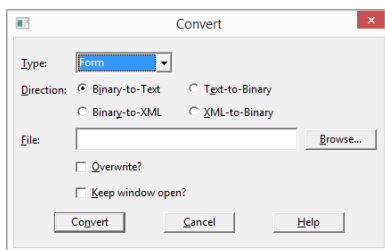

Figure 17 : Module converter dialog.

Product Documentation and Community

The Builder Help menu now includes links to Builder Help, the Documentation Library, Oracle Forms Forum on OTN, and the Oracle Forms product page on OTN. These can be found by selecting Help from the Builder menu.

Java Developer API (JDAPI)

The Forms Java Developer API allows developers to create Java applications that can manipulate Forms modules in mass. This can be extremely helpful if the same change is required in many modules. In previous versions of the JDAPI, it was not possible to generate a PLX file using the JDAPI. This limitation meant that developers using JDAPI needed to make their changes then separately generate new PLX files. In this version, PLX files can now be generated programmatically. It is also now possible to execute the "compile_all" option prior to generating new FMX or MMX files. The compile_all option does not apply to PLL/PLX files.

WebUtil

Never copy webutil.pll/plx from previous versions. Each Forms release comes with a copy that is intended for that version and includes any bug fixes or new features unique to that version. A new webutil.plx should be created as part of any installation's post installation steps.

WebUtil without OLE

WebUtil is a Forms optional add-on that provides client side functionality not natively available. Previously, using this add-on required the inclusion of third party libraries (i.e. JACOB). These third party libraries needed to be signed with a trusted digital certificate. JACOB provides the integration components necessary for Oracle Forms to communicate with OLE functionality, generally for interaction with Microsoft Office products. For some users, although some features of WebUtil are needed, OLE functionality is not needed or used in their applications. WebUtil is now delivered with an updated Object Library that includes an alternative object group which does not contain the OLE components. This allows developers, who do not plan to use OLE, to have a lighter weight WebUtil option with no need for third party libraries.

File Last Modified Date

Knowing the last time a file was modified can be helpful information for example when comparing a file on the user's machine to a file the server is attempting to deliver to the user. If, as an example the file on the user's machine is already current it may be decided to not replace it. The following two new functions have been added to the WEBUTIL_FILE package to accomplish this task.

» FILE_MODIFIED_DATE
  Returns date/time as a string with the default format set to "dd-MMM-yyyy HH:mm:ss". To change the date format, add the applet parameter WebUtilDateFormat and include the desired format. Setting an invalid format will result in the default format being used.

» FILE_MODIFIED_EPOCH
  Returns the amount of time, in seconds since January 1, 1970 until the last modified date of the file. This is also known as POSIX or Unix time. This value can be helpful when attempting to do time/date calculations or comparisons.

Extend WEBUTIL_SEPARATE_FRAME Package

WEBUTIL_SEPARATE_FRAME has been extended to support all non-browser configurations like Java Web Start and Standalone Launcher (FSAL). The following functions are now supported with JWS and FSAL.

- » IsSeparateFrame
- » GetSeparateFrameSize
- » ShowMenuBar
- » ShowStatusBar
- » SetTitle
- » SetIcon
- » AllowResize
- » CenterMDI

Conclusion

Oracle Forms 12c (12.2.1) includes many new features and enhancements. These new features range from the very complex to the very simple. By introducing new features into new and existing applications, those applications take on a more modern presence. More importantly, by leveraging the power of new features and enhancements an end-user's interaction with the application becomes more efficient. Additional information about Oracle Forms 12c and its features can be found in the Oracle Forms 12c Documentation Library.

ORACLE®

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

CONNECT WITH US

blogs.oracle.com/oracle

facebook.com/oracle

twitter.com/oracle

oracle.com

Integrated Cloud Applications & Platform Services

ORACLE FORMS 12C - NEW FEATURES
August 2017
Author: Oracle Forms Product Management
Contributing Authors: Oracle Forms Support Global Technical Lead Team

Oracle is committed to developing practices and products that help protect the environment