

FORMS PERSONALIZATION FEATURE – NEW FOR 11.5.10

Susan Behn, Solution Beacon, LLC

Introduction

The Oracle E-Business Suite is known for its flexibility and configurability. Most Oracle Applications users are familiar with using profile options, setups, key and descriptive flexfields, folder tools, etc... to achieve this flexibility. However, few users know that Oracle also supports changing the functionality of forms. This has historically been available through the use of a special library named CUSTOM.pll. Now it is even easier. Starting with release 11.5.10, Oracle has provided a new tool to declaratively alter form functionality.

Oracle provides two different personalization tools. OA framework personalizations are for self services application forms such as iExpense and iProcurement. Forms 6i personalizations are for the forms built using the Forms Builder 6i tool. These are most of the core application forms such as in General Ledger, Payables, Assets, Order Entry, etc... The purpose of this paper and presentation is to provide a brief history and background of CUSTOM.pll, and to provide examples for the new tool for Forms 6i personalizations.

Definitions

Functionality of Oracle Applications can be altered using a variety of methods. As the Applications mature, some of the terminology has matured as well. For the purposed of this paper, these methods are defined as follows:

Extensions

Extensions are *additional code* or objects added to the E-Business suite that do not alter the existing functionality. Examples include new forms created from TEMPLATE.FMB, new reports, interfaces and other new objects such as views used for custom reporting.

Customizations

Customizations are *changes* to forms, reports or any other objects delivered with the E-Business Suite. Direct modifications of forms, reports, packages or other objects is not supported or protected during patching. Customization of the E-Business Suite is not recommended and should be used only as a last resort.

Personalizations

Personalizations are *changes* to existing forms using tools provided by Oracle to alter the appearance or behavior of the form or function. These changes are generally not impacted by upgrades or patches, and the use of personalizations is supported by Oracle with some limitations to be discussed later. Personalizations for 6i forms are implemented using CUSTOM.pll or the new forms personalization tools provided by Oracle in release 11.5.10.

Pre 11.5.10 – CUSTOM.pll

The CUSTOM.pll library has been available long before Release 11i; however, this brief description is based on feature in Release 11i and later. CUSTOM.pll is a pl/sql library located in \$AU_TOP/resource on the forms server. Using this library, the applications user can modify the look and behavior of application 6i forms to “personalize” the application form. This type of programmatic modification is supported by Oracle with limitations. Oracle will not help you make the changes; you may be required to disable the custom code while working on a Service Request; you must ensure your changes do not alter the data in a way that will conflict with Oracle Applications.

Programmatic personalization is accomplishing by adding code to the CUSTOM.pll package body using Forms Builder 6i specifying the event for which the code should execute. Some examples of typical modifications are in the list below.

- Hide fields or tabs
- Make fields required

- Restrict update or insert
- Change prompts or labels
- Restrict values returned in a List of Values
- Create zooms and tool bar menu selections

After pl/sql coding is complete, CUSTOM.pll must be moved to the forms server and compiled.

Custom.pll Methodology and Example

The following methodology for implementing changes in the custom library allows multiple developers to implement custom library changes concurrently, reduces the potential problem of reaching the maximum size limitation in CUSTOM.pll and results in a modular custom library implementation that is easier to debug and maintain.

The code required to customize Oracle Applications is placed in a separate library for each Oracle form requiring personalizations. The seeded CUSTOM.pll will be modified to attach these customized libraries. Calls to the libraries will be placed in the CUSTOM package body.

Create New Libraries

Complete the following steps to create a new library for an Oracle form requiring personalizations.

1. Create a new library using forms developer using the naming convention XXXXXX{existing form name}. For example, the name for the custom library for the Enter Supplier form is XXXXXAPXVDMVD.pll.
2. If it is not already attached, attach the APPCORE2.pll library.
3. Create a package spec and package body for the new library. Place custom code in this package. Figure 1 contains an example of a custom package spec. Figure 2 contains an example of a custom package body.

```
PACKAGE XXXXXAPXVDMVX IS
    Procedure event(event_name VARCHAR2);
END;
```

Figure 1 Package Spec

```
PACKAGE BODY XXXXXAPXVDMVX
IS
    PROCEDURE event (event_name VARCHAR2) IS
        BEGIN
            IF event_name = 'WHEN-NEW-FORM-INSTANCE' THEN
                \*Force Upper Case for Supplier Name*\
                APP_ITEM_PROPERTY2.SET_PROPERTY('VNDR.VENDOR_NAME_MIR',CASE_RESTRICTION,UPPERCASE);
                \* Hide the tax payer id*\
                APP_ITEM_PROPERTY2.SET_PROPERTY('VNDR.NUM_1099_MIR',DISPLAYED,PROPERTY_OFF)
                \*Change the prompt*\
                APP_ITEM_PROPERTY2.SET_PROPERTY('VNDR. END_DATE_ACTIVE_MIR',PROMPT_TEXT,'Inactive Date')
            END IF;
            END event;
        END XXXXXAPXVDMVX;
```

Figure 2 Package Body

Modify CUSTOM.pll

The only additions to CUSTOM.pll are calls to the custom libraries in the package body and attachments to the custom libraries. An example of the objects in CUSTOM.pll is shown in Figure 3. An example of the modified portion of the CUSTOM package body is shown in Figure 4.

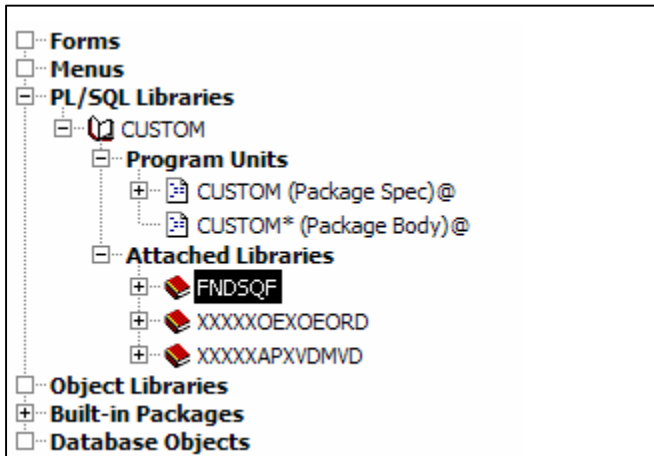


Figure 3 CUSTOM.pll Object Navigator View

```

Form_name      varchar2(30) :=
                name_in('system.current_form');
Begin
  If form_name = 'APXVDMVD' THEN
    xxxxxapxvdmvd.event(event_name);
  Elsif form_name = 'OEXOEORD' THEN
    xxxxxoexoeord.event(event_name);
  end if;
end event;

```

Figure 4 CUSTOM Package Body

NOTE: Future modifications to existing custom libraries for individual forms only require compilation of the modified library for the form. CUSTOM.pll requires compilation only when a new library is created and attached using the methodology described in this document.

Forms Personalization

Forms personalizations declaratively alter the behavior of 6i forms delivered with the E-Business Suite. Although the declarative nature of this tool does not require the use of pl/sql, it is generally recommended that the user have some understanding of pl/sql and forms in order to understand the impact of the changes. Forms personalizations are effective immediately as opposed to CUSTOM.pll which must be compiled.

Most changes traditionally done using CUSTOM.pll can be accomplished using forms personalization; however, users may still choose to utilize CUSTOM.pll for complex modifications. For the same event, the declarative forms personalizations will fire prior to the CUSTOM.pll for the same event.

Profile Options Impacting Forms Personalization

Hide Diagnostics menu entry – This profile options must be set to “No.” Setting this profile options to “Yes” hides the diagnostics menu.

Utilities: Diagnostics – If this profile options is set to “Yes” the apps password is not required to use diagnostic features including forms personalization and examine. These are very dangerous utilities. Set this profile option to “No” in production environments.

Creating Forms Personalizations

The following examples will describe personalizations for the Supplier form.

First access the form you wish to personalize. After opening the form, go to Help → Diagnostics → Custom Code → Personalize to navigate to the forms personalization tool. The forms personalization form for the suppliers form is shown below in Figure 5.

Seq	Description	Level	Enabled
10	Susan's Test	Form	<input checked="" type="checkbox"/>
			<input type="checkbox"/>
			<input type="checkbox"/>
			<input type="checkbox"/>

Figure 5 Form Personalizations (Suppliers)

Header

The function name and form name will default to the current function and form. These values may not be changed. The debug mode can be set to off, Step-by-Step or Show Debug Messages. *Step-by-Step* will display a pop-up window showing events impacted by the rule. *Show Debug Messages* will show messages where the type = debug.

Sequence numbers are set between 1 and 100 and can be reused. Rules will run in sequence.

Description is free form entry.

Level can be set to form or function beginning with the CU1 patch. Prior to the CU1 patch, personalizations could only be set at the form level. For example, you may want to create a personalization that applies only to the query-only function for the Supplier form.

Personalizations are enabled by checking the *Enabled* check box.

Condition

TRIGGER EVENT

The following provides some guidelines on which standard trigger event to use for specific types of personalizations. The events included in this list are the events Oracle generally provides in all forms. However, you must test to make sure the trigger event is actually firing. To find events that are firing, go to Help → Diagnostics → Custom Code → Show Custom Events. As you navigate through the form, a pop-up window will display the trigger events as they fire. You may also choose

to use any other event fired in the form, however, Oracle does not guarantee that other events will be retained in patches, so these personalizations may not be protected.

- WHEN-NEW-FORM-INSTANCE
 - Security rules
 - Navigation rules
 - Visual attributes
- WHEN-NEW-BLOCK-INSTANCE
 - Same as WHEN-NEW-FORM-INSTANCE
 - Message rules
- WHEN-NEW-RECORD-INSTANCE
 - Default values
- WHEN-NEW-ITEM-INSTANCE
 - Message rules
 - Default values dependent on entry of another item
- WHEN-VALIDATE-RECORD
 - Populate hidden fields
 - Additional validations
- MENU_n
 - Populate tools menu (MENU1-15) (CU1 patch)
- SPECIAL_n
 - Populate tools menu (SPECIAL 1-15)
 - Populate reports menu (SPECIAL 16-30)
 - Populate actions menu (SPECIAL 31-45)
- ZOOM – recommend using MENU_n or SPECIAL_n rather than zoom
- KEY-Fn

The list of values contains the standard trigger events above, however, since Oracle allows you to use other non-standard events, the trigger event field does not validate from the list of values. You can choose a value from the list, but if you are trying to use a standard event and type it incorrectly, your personalization will not work.

Note: You should use MENU1-15 before SPECIAL1-45 to avoid conflict with any pre-existing menu items seeded by Oracle.

TRIGGER OBJECT

The trigger object may be required depending on the trigger event. If the list of values is available when the cursor is in the trigger object field, then the trigger object is required. For example, the following events will require a block name.

- WHEN-NEW-BLOCK-INSTANCE
- WHEN-NEW-RECORD-INSTANCE
- WHEN-VALIDATE-RECORD

The WHEN-NEW-ITEM-INSTANCE trigger event will require a block and field name separated by a period. Other non-standard events may also require a trigger object depending on the event. In this case, the absence of a list of values does not necessarily mean the trigger object is not required.

CONDITION

The condition is an optional sql code fragment used to limit the scope of the personalization. For example, you can limit a message to appear only certain times of the year based on the system date. You may reference the value of a field in the current record in the format *:blockname.fieldname*.

PROCESSING MODE

The processing mode determines when this personalization is applicable. Options are *Only in Enter-Query Mode*, *Not in Enter-Query Mode* or *Both*.

Context

The context determines who the personalization applies to. Multiple rows are allowed and will be processed as “and” statements. Applicable levels are site, responsibility, and user. Industry exists in the list of values, but is reserved for future use. If the context is null, the personalization will apply to all.

TIP: When testing new personalizations, set the context so the personalization only applies to your own user id.

Actions

The Actions tab shows the actions on the left and details about the action on the right. There are 4 action types – property, message, builtin, and menu which will be described in detail in the following pages. The details on the right side of the tab page will change depending on the type of action. Figure 6 shows the tab page for an action type of *property*.

Seq	Type	Description	Language	Enabled
10	Property	Vendor Type Requir	All	<input checked="" type="checkbox"/>
				<input type="checkbox"/>
				<input type="checkbox"/>
				<input type="checkbox"/>
				<input type="checkbox"/>
				<input type="checkbox"/>
				<input type="checkbox"/>
				<input type="checkbox"/>
				<input type="checkbox"/>

Select By Text...
 Object Type: **Item**
 Target Object:
 Property Name: **PROMPT_TEXT**
 Value:
 Get Value

Figure 6 Actions

Actions have a non-unique sequence number from 1 to 100. The description is free form entry. You may choose a language if your site is enabled for multiple languages, and you may choose to enable or disable an action using the check box.

PROPERTY ACTION TYPE

Property action types require an object type, target object, property name and value. The properties for object types listed below are available for personalization.

- Item – size, position, visibility, insert, update, delete, value, case restriction, LOV, format mask and more
- Window – size, position, title, visibility
- Block – size, position, default where clause and order by, navigation, insert, update, delete
- Tab Page – enabled, displayed, label
- Canvas – size, position, visibility
- Radio Button – size, position, label, prompt, visibility
- View – size, position
- Global Variable – initial value, value
- Local Variable – value
- LOV – size, position, group name

A list of values is available for the object type, target object and property name. Use the *Select by Text* button to select the target object by prompt name. Use the *Get Value* button to retrieve the current value. Enter the value for the property selected.

Values starting with an = sign will be evaluated at run time. Values that do not start with an = sign are taken as a literal.

MESSAGE ACTION TYPE

Message action types require a message type and message. Messages can be displayed at trigger events other than WHEN-NEW-FORM-INSTANCE. The following list describes the different message types.

- Show – Informational Message
- Hint – Appear on status bar
- Error – Requires user response
- Debug – Only displays if debug mode is set to Show Debug Messages
- Warn – Informational message with caution symbol

BUILTIN ACTION TYPE

The following builtins types are available:

- Launch SRS Form – launches a concurrent program
- Launch a Function – launches another form function
- Launch a URL – launches any URL
- DO_KEY – execute a form builtin
- Execute a Procedure – execute any procedure using syntax exactly as you would in pl/sql code
- GO_ITEM – navigate to a specific item
- GO_BLOCK – navigate to a specific block
- FORMS_DDL – issue dynamic sql statements
- RAISE FORM_TRIGGER_FAILURE
- EXECUTE_TRIGGER – call a trigger.
- SYNCHRONIZE – synchronize form on client and middle tiers
- Call Custom Library – call a specific event you have coded directly in CUSTOM.pll
- Create Record Group from Query – creates a record group dynamically. Assign this new record group to a list of values to limit data returned in the list.

Parameters for builtins vary based on the builtin type. Several examples are provided later.

MENU ACTION TYPE

Using the menu action type *only displays the menu* in the appropriate location on the tool bar. (The builtin to execute the functionality behind this menu item is covered in a later section.) Menus are typically established during the WHEN-NEW-FORM-INSTANCE or WHEN-NEW-BLOCK-INSTANCE event. See Figure 7 for the fields applicable to menu actions. Select a menu entry and give it a label. You may also choose an icon and choose which blocks should include this menu item. The *Add Block* button will provide a list of blocks. To select multiple blocks separate block names with a comma or use the *Add Block* button multiple times to select multiple blocks. Leaving the *Enabled in Block(s)* field null will enable the menu item in all blocks.

The screenshot shows the 'Actions' tab in the Oracle Forms Personalization tool. A table lists the actions, with the first row selected:

Seq	Type	Description	Language	Enabled
10	Menu	Google	All	<input checked="" type="checkbox"/>
				<input type="checkbox"/>
				<input type="checkbox"/>
				<input type="checkbox"/>
				<input type="checkbox"/>
				<input type="checkbox"/>
				<input type="checkbox"/>

Configuration fields for the selected action:

- Menu Entry: MENU5: Google
- Menu Label: Google
- Render line before menu
- Icon Name: (empty)
- Enabled in Block(s): VNDR , SITE
- Add Block... button

Figure 7 Menu Actions

Example 1 – Force upper case for Vendor name – Shown in Figure 8

Trigger Event = WHEN-NEW-FORM-INSTANCE

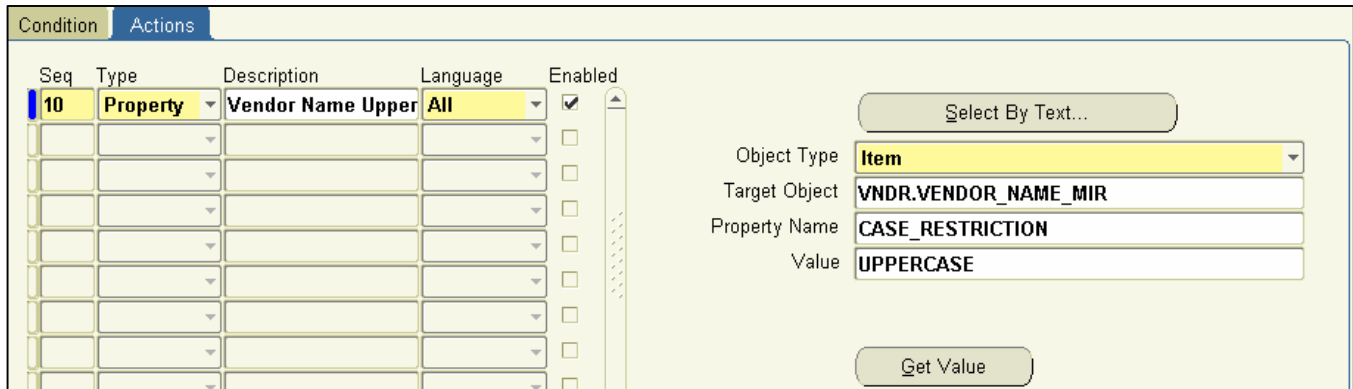


Figure 8 Example 1

Example 2 – Set a global variable to the value of the email address in FND_USERS and display this value in a message

Trigger Event = WHEN-NEW-FORM-INSTANCE

First set the global variable as shown in Figure 9.

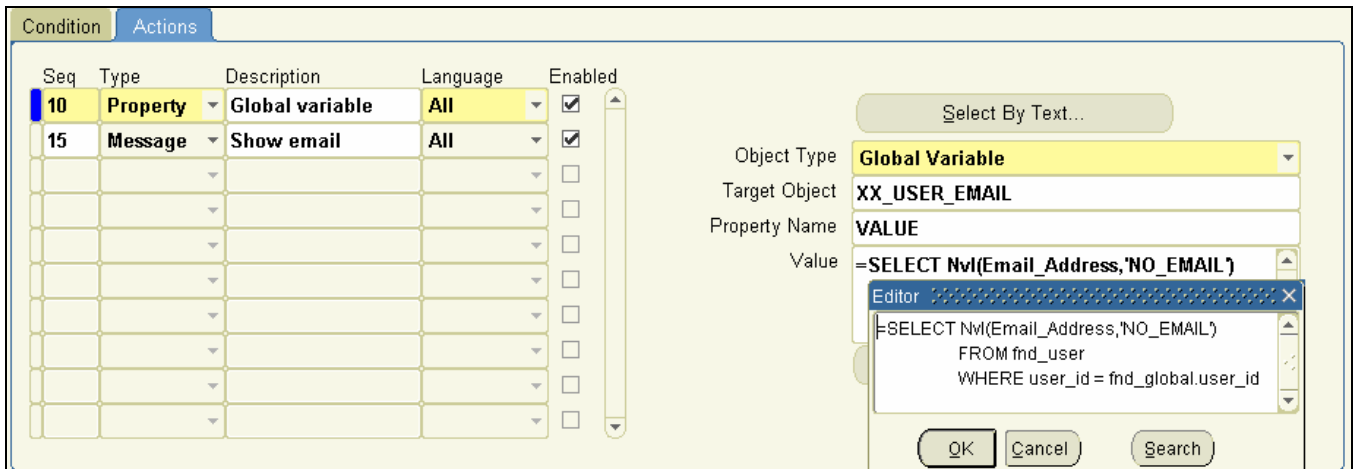


Figure 9 Set Global Variable

Next, set a message to display the global variable as shown in Figure 10.

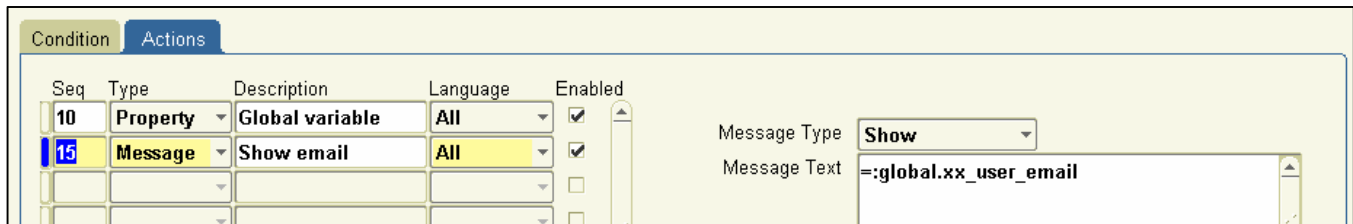


Figure 10 – Display a Message

Example 3 – Launch a function (CU1 patch) to view payment history form

Trigger Event = WHEN-NEW-FORM-INSTANCE

Step 1 – Establish the menu entry as shown in Figure 11.

Seq	Type	Description	Language	Enabled
10	Menu	Payment History	All	<input checked="" type="checkbox"/>
				<input type="checkbox"/>
				<input type="checkbox"/>
				<input type="checkbox"/>
				<input type="checkbox"/>

Menu Entry: MENU3: Payment History

Menu Label: Payment History

Render line before menu

Icon Name:

Enabled in Block(s):

Figure 11 – Define menu

Step 2 – Set a global variable to pass the vendor name to the payment history function as shown in Figure 12.

Seq	Type	Description	Language	Enabled
10	Property	Global - vendor id	All	<input checked="" type="checkbox"/>
20	Builtin	Payment inquiry	All	<input checked="" type="checkbox"/>
				<input type="checkbox"/>
				<input type="checkbox"/>
				<input type="checkbox"/>

Select By Text...

Object Type: :GLOBAL Variable

Target Object: XX_VENDOR_NAME

Property Name: VALUE

Value: =:VNDR.VENDOR_NAME_MIR

Figure 12 – Set a global variable

Step 3 – Use a Builtin action to launch a function to view payment history form as shown in Figure 13.

Seq	Type	Description	Language	Enabled
10	Property	Global - vendor id	All	<input checked="" type="checkbox"/>
20	Builtin	Payment inquiry	All	<input checked="" type="checkbox"/>
				<input type="checkbox"/>
				<input type="checkbox"/>
				<input type="checkbox"/>

Builtin Type: Launch a Function

Function Code: AP_APXPWALL_VIEW

Function Name: View Payment Overview

Parameters:

Figure 13 – Launch a Function

Step 4 – Access forms personalization for the payment history form via Payments → Inquiry → Payments Overview. Set the vendor name in the query find form to the global variable established in the supplier form only if access to this form was not from the personal home page as shown in Figure 14. :SYSTEM.LAST_FORM = 1 when this form is accessed from the personal home page. When forms are accessed directly from the home page, the navigator is bypassed and global variables cannot be initialized. Failure to set this condition will result in an error when the action is executed to set the vendor name in the query form to the global variable that does not exist.

Condition	Actions
Trigger Event: WHEN-NEW-ITEM-INSTANCE (You can enter additional event names.)	
Trigger Object: PAYMENT_QF.START_CHECK_NUMBER	
Condition: :SYSTEM.LAST_FORM > 1	
Processing Mode: Both	

Figure 14 – Check condition

Step 5 – Assign the vendor name in the query form to the value in the global variable established in the supplier form as shown in Figure 15.

The screenshot shows the 'Actions' tab in the Oracle Forms Personalization tool. A table lists actions with columns for Seq, Type, Description, Language, and Enabled. Action 10 is a 'Property' type, and action 20 is a 'Builtin' type. To the right, the configuration for action 20 is shown: Object Type is 'Item', Target Object is 'PAYMENT_QF.DISPLAYED_VENDOR_NAME', Property Name is 'VALUE', and the Value is set to the global variable expression '\$(global.xx_vendor_name.value)'. Buttons for 'Select By Text...' and 'Get Value' are also visible.

Seq	Type	Description	Language	Enabled
10	Property		All	<input checked="" type="checkbox"/>
20	Builtin		All	<input checked="" type="checkbox"/>

Object Type: Item
 Target Object: PAYMENT_QF.DISPLAYED_VENDOR_NAME
 Property Name: VALUE
 Value: \$(global.xx_vendor_name.value)

Figure 15 – Use global variable

Step 6 – Use a Builtin action to execute DO_KEY → NEXT_BLOCK as shown in Figure 16. This executes the find query.

The screenshot shows the 'Actions' tab in the Oracle Forms Personalization tool. Action 20 is selected as a 'Builtin' type. The configuration for this action is shown: Builtin Type is 'DO_KEY' and the Argument is 'NEXT_BLOCK'.

Seq	Type	Description	Language	Enabled
10	Property		All	<input checked="" type="checkbox"/>
20	Builtin		All	<input checked="" type="checkbox"/>

Builtin Type: DO_KEY
 Argument: NEXT_BLOCK

Figure 16 – DO_KEY Builtin

Personalization Tips

- If you disable a tab page, make sure the user cannot still navigate to the items on the tab page
- You may need to exit and re-open the form to see personalization changes
- After upgrades, go to the personalization for each form and choose Tools → Validate All
- Use debug message before and after events
- Initialize global variables to null in the navigator form using the WHEN-FORM-NAVIGATE trigger event
- Use the *Validate* button to validate strings
- Conditions will return true, false or error
- Values will return the resulting string or an error
- Use the *Apply Now* button to apply the action now and see the results (does not always work if dependant on the results of another action)
- Use the Insert 'Get' Expression button to get any property of an item (CU1 patch)
- Turn custom code off to confirm any form problem is due to custom code Help → Diagnostics → Custom Code → Off
- Use FNDLOAD as shown below to move personalizations to other instances
 - Download for a specific form:


```
FNDLOAD <userid>/<password> 0 Y DOWNLOAD $FND_TOP/patch/115/import/affrmcus.lct
          <filename.ldt> FND_FORM_CUSTOM_RULES form_name=<form name>
```
 - Download all personalizations


```
FNDLOAD <userid>/<password> 0 Y DOWNLOAD $FND_TOP/patch/115/import/affrmcus.lct
          <filename.ldt> FND_FORM_CUSTOM_RULES
```
 - Upload

```
FNDLOAD <userid>/<password> 0 Y UPLOAD $FND_TOP/patch/115/import/affrmcus.lct  
<filename.ldt>
```

Additional Resources

The documentation listed below contains details related to extending and personalizing Oracle Applications.

- Oracle Applications User Interface Standards for Forms-Based Products
- Oracle Applications Developer's Guide
- Oracle Applications System Administrator's Guide
- Oracle Applications User Guide
- MetaLink note 279034.1 – Forms Personalization
- Configuring, Reporting and System Administration in HRMS
- Oracle Self Service Web Applications Implementation Manual
- www.solutionbeacon.com – newsletters, free tools, white papers and presentations, Vision access