

SELECT[®] Online

The Journal of the International Oracle Users Group

Select Journal
Home

Regular Columns

Past Issues

General Information

IOUG Home



Select Magazine - January 2001

Volume 7, No. 1

Documentation What's that? Statements

By James Shiff

Documentation is an often neglected or forgotten part of the system development life cycle. Find out how one developer solved this problem using a documentation map, directly accessible from the forms and reports being documented.

Documentation is an often neglected or forgotten part of the system development life cycle. Find out how one developer solved this problem using a documentation map, directly accessible from the forms and reports being documented. Documentation standards

When others come and ask What does this code do or Can you help me understand this part of your program?, we all know the forbidden word in ISE..Documentation. We all have been there and we all say, Go read the documentation. Sometimes the reason that these questions are asked is because no updated documentation exists. If it were written well and in a logical order to follow, we could continue to write or modify programs because we could understand what others did or thought they were doing. Also we would get inspired to copy their format and expand on their knowledge to write our own documentation as well or better. The problem with most documentation is that it hardly ever gets written or updated when it should because we just never have enough time to do it.

In Developer Forms and Reports the most common documentation is in-line, namely following or preceding the logic written. That's perfect! But we need to go one step further. How do we know what other program units or database objects are being modified or what they are doing if we only see one-liners (or a group) in the last object we open in a form or report? This problem is made more complex by having logic changed or added in many other objects in a form [e.g. a trigger added, attributes changed, library added or a modified program unit]. How do you follow the logic when it is scattered in many different places? It would be nice if we had a central repository (a map) to start from to show us where to go for more detailed information.

If we had such a map that was easy to read so that anyone could follow it, we could create more forms and reports that have great code and documentation. So, how do we get this documentation to work so that everyone who creates great code can also follow other developers logic by simply using this map?

The answer is to continue to use in-line documentation everywhere and use the current date (from the central repository) within the header documentation as a pointer from the main documentation. An example follows: In the POST-CHANGE trigger:

```

Begin
--*****
--10/25/00
--Call the Oracle function to calculate the retail amount
--*****
:ti_calc_retail := DAKSF001_FNC(:tdaalw.net_cst_amt,
                               :tdaalw.mkup_pct);
END;
```

You can also create a procedure in the program unit (central repository) called Form_Documentation (in Developer Forms) or Report_Documentation (in Developer Reports) that holds all of the general knowledge about what is being modified or added in the module.

```
PROCEDURE FORM_DOCUMENTATION IS
BEGIN
    NULL;
```

```
*****
-- Documentation for Debit Allowance Entry/Maintenance, DA_ENTRY_MAINT.
-- Author: Paul Keber
-- Date: 09/19/2000

    Purpose: To allow the user to enter and change debit allowances.

--      This form can be entered in either NEW or CHG mode. NEW mode is
used to enter a

-- new debit allowance. CHG mode is used to change an existing debit allowance.

-- All required fields must be entered. When the OK or the
OK+Repeat button is

-- pressed, validation checks are performed on all enterable fields. If there are

-- any errors, an error window will be displayed
detailing all errors. If there

-- are no errors, logic to save the debit allowance will begin. If
the window is

-- closed, an alert box is displayed asking if changes are to be
saved. If the

-- user replies Yes, the same logic to save the debit allowance is
applied.

--

-- If no changes had been made on the window when the OK or the OK+Repeat
button

-- is pressed, the form will be closed without any more actions being performed.

--

-- When changes are being saved in NEW mode, the Notes form is displayed requiring
-- the user to enter a vendor note. Once the vendor note is entered, the Notes
-- form is closed and a message window displays the new debit allowance number
-- and asks the user if the debit allowance is to be printed. After responding to
-- the question, the form is closed if the user selected the OK button. If
-- the user selected the OK+Repeat button, all Debit Allowance Value and
-- Proration Level fields are cleared out, and the user may enter another debit
```

- allowance.
-
- When changes are being saved in CHG mode, fields changed or entered are saved
- after they are validated. The user may cancel changes by clicking the Cancel button.
- If the user presses the Notes button, Vendor and Kohl's notes may be changed on
- the form. Changes made on the Notes form are only posted, not committed, by the
- Notes form. They are committed by this form when the user clicks the OK
- button, or when the user clicks the Cancel button and then responds Yes when asked
- if he/she wants to save the changes.
-
- The Refresh button is used to reset the fields on the Entry/Maintenance window
- to the values that existed at the time the form was entered or how it was
- displayed after the OK+Repeat button was pressed.
-
- The Delete button is used to delete an existing debit allowance which is
- brought up in CHG mode.
- Modification History Log
-
-

```

Date          Modified By      Unit      Remarks
-----
*****
-- 10/25/2000 PJK          TDAALW      Modified POST-CHANGE on net_cst_amt and
created a function to return the data
--                required.

END;

```

This type of documentation describes any multiple location or other important issues. Since we are in Forms or Reports, it is quick and easy to follow step by step. Compare this type of documentation to other ways in which documentation is often hidden in directories on our PC, LAN, or version control packages that are often forgotten. Much documentation that does exist is outside of the Form or Report with no way of cross referencing what and where modifications were made.

Example of a complicated form:
 Example of Form documentation:
 Example of in-line documentation
 Documentation for Forms and Reports can be created by placing a general history modification log in the

Forms_documentation or Report_documentation in the program unit section and also in the specific location (if possible) of the modified code Example:

```

PROCEDURE FORM_DOCUMENTATION IS
BEGIN
  NULL;
--      *****
--Documentation for Logon Security. SE_LOGON
--Author:      Jim Shiff
--Date:   09/15/2000
--
--Purpose:     To determine if the username and Password is valid in the
--RACF tables. Also verify the environment the SE_LOGON is running in.
--A JavaBean will help support the call to the RACF tables using a Java Wrapper.
--
--Modification History Log
--
--Date   Modified      Unit           Remarks
--   -----
*****
END;
```

Where can this documentation be placed?

Document for Developer Forms Form_documentation (Program unit Procedure)

Document for Developer Reports Report_documentation (Program unit Procedure)

How can you use this documentation?

Wherever the actual code resides (program unit, object group, LOVE.) it will use the date in this modification history log as a reference. The general and/or detail notes can be placed here in the Remarks section, with a more detailed explanation placed where the actual code resides.

Conclusion

Documentation should be a way of life for most IT shops. It should be completed before going to production. How it is completed and the timing will be based on your standards and your developers own rules. There's an old saying Pay me now or pay me later which can be applied to documentation; Document me now so others can understand or Make up documentation later and have bewildered developers always at your back door asking, What does this code do? Good luck in your future documentation standards.

About the Author

Jim Shiff is a Senior Technical Analyst for Kohls Department Stores.He has been in the IT field for 13 years, the last 8 in Oracle as a Developer in PL/SQL and Reports, and as a DBA.Jim would like to extend his thanks to the Oracle developers at Kohls for their support.You can reach him at jim.shiff@kohls.com

[Download Acrobat Reader](#)

Copyright 2003 by the International Oracle Users Group