R12.2 Extension and Development Tips and Tricks

Session ID: 10425

Prepared by:
John Peters
JRPJR, Inc.
john.peters@jrpjr.com
Rev 3

@JohnJrPeters

About the Speaker

- 23 years working with E-Business Suite
- Primarily a Technical Focus
  While functional I also do quite a bit of development
- Workflow SIG Chairman
- Founding Board Member of the Northern California (NorCal) OAUG
- I enjoy sharing what I know about the Oracle E-Business Suite, I have many presentations posted on my web site at: http://jrpjr.com

Agenda

- Development Standards (2 tips)
- OnLine Patching, EBR (6 tips)
- Concurrent Programs (4 tips)
- Interfaces (3 tips)
- PL/SQL Code (7 tips)
- Don’t Hard Code Configure Instead (5 tips)
- Use New Technology (3 tips)

- This is revision 3 of this presentation. Always check my web site for the most recent revision http://jrpjr.com, look under Paper Archives

Development Standards

COLLABORATE16
TECHNOLOGY AND APPLICATIONS FORUM FOR THE ORACLE COMMUNITY

1) Oracle EBS Development Standards

- It is extremely important in the R12.2 world to follow the Oracle EBS Development Standards:
  - Deploying Customizations in Oracle E-Business Suite Release 12.2 (MOS Doc ID 1577661.1),
  - Oracle® E-Business Suite Developer's Guide Release 12.2 (Part No. E22961-13), September 2015
  - Oracle® Workflow Developer's Guide Release 12.2 (Part No. E22011-11), September 2015
  - Oracle® E-Business Suite Integrated SOA Gateway Developer's Guide Release 12.2 (Part No. E20927-10), September 2015
  - Oracle® E-Business Suite Desktop Integration Framework Developer's Guide Release 12.2 (Part No. E22005-10), September 2015

2) Your EBS Development Standards

- You need your own organizations development standards that cover:
  - Documentation
    - What was the gap, who requested it, high level design, on going maintenance, how to migrate it, etc
  - Object Naming Conventions
    - Not just DB and File System Objects
    - Also migration and patch conventions
    - Script structure
  - Source Code Control
    - Now with multiple filesystems and code in the database it become even more important that you start making changes from the current version
    - Check out some of the OpenSource options:
      - GIT/GITHub (Merge)
      - Subversion (Merge or Lock)

OnLine Patching, EBR

COLLABORATE16
TECHNOLOGY AND APPLICATIONS FORUM FOR THE ORACLE COMMUNITY

3) Register You Custom Application

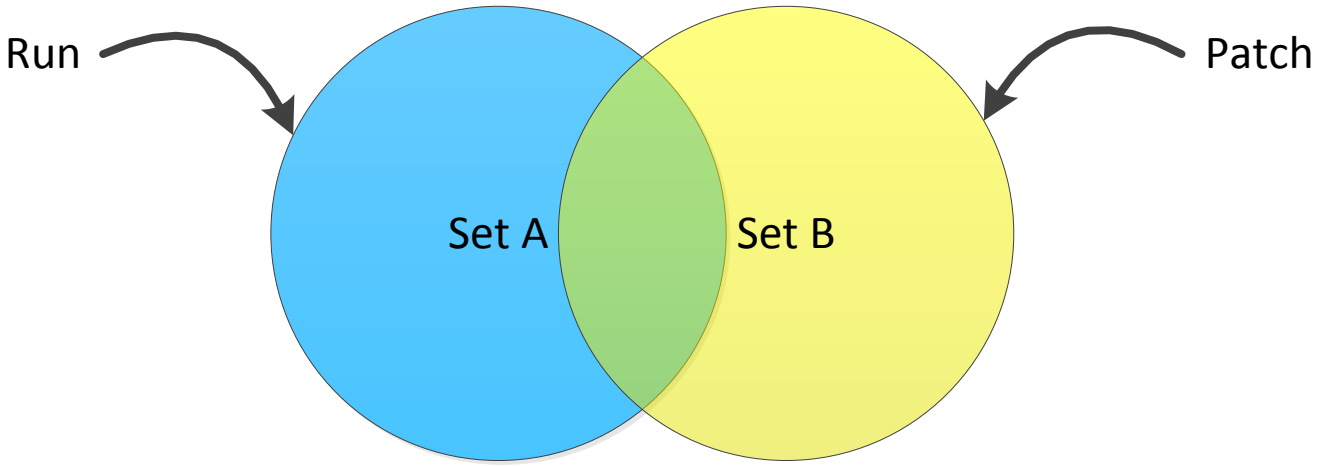
- This is essential in R12.2 for Edition Based Redefinition (EBR) to work.
- This will include your custom top and environment variables to get to it.
- Come up with a short name prefixed by 'XX' and use this throughout your custom work.
  - DB Objects, Filesystem Objects, Concurrent Programs, etc
  - A good option is a stock trading symbol (1-4 characters) or a company abbreviation
- Keep Short Name and Product Code the same.
- One Custom Application vs Module Centric Custom Applications
- Possibly create additional applications if you have some really large customizations

```
select t.APPLICATION_NAME,
       t.DESCRIPTION,
       a.APPLICATION_SHORT_NAME,
       a.PRODUCT_CODE,
       a.BASEPATH
  from FND_APPLICATION_TL t,
       FND_APPLICATION a
 where a.APPLICATION_ID = t.APPLICATION_ID
   and a.APPLICATION_SHORT_NAME like 'XX%';
```

Application Name	Description	Short Name	Product Code	Base Path
O'Reilly Media Custom	O'Reilly Media Custom Application	XXORM	XXORM	XXORM_TOP

4) Know Edition Based Redefinition

- Become very familiar with Edition Based Redefinition (EBR) in the DB and understand how it works especially within the EBS OnLine Patching paradigm



4) Know Edition Based Redefinition

- Editionable DB Objects are:
  - Synonyms
  - Views
  - All PL/SQL Object Types
    Function, Package Spec/Body, Procedure, Trigger, Type and Type Body
- Everything else in the DB is non-Editionable
- Simple Rules
  - A Non-Editioned Object cannot depend on an Editioned Object
  - An Editioned Object (View) cannot be involved in a Foreign Key Constraint
  - An Abstract Data Type cannot be both editioned and evolved

4) Know Edition Based Redefinition

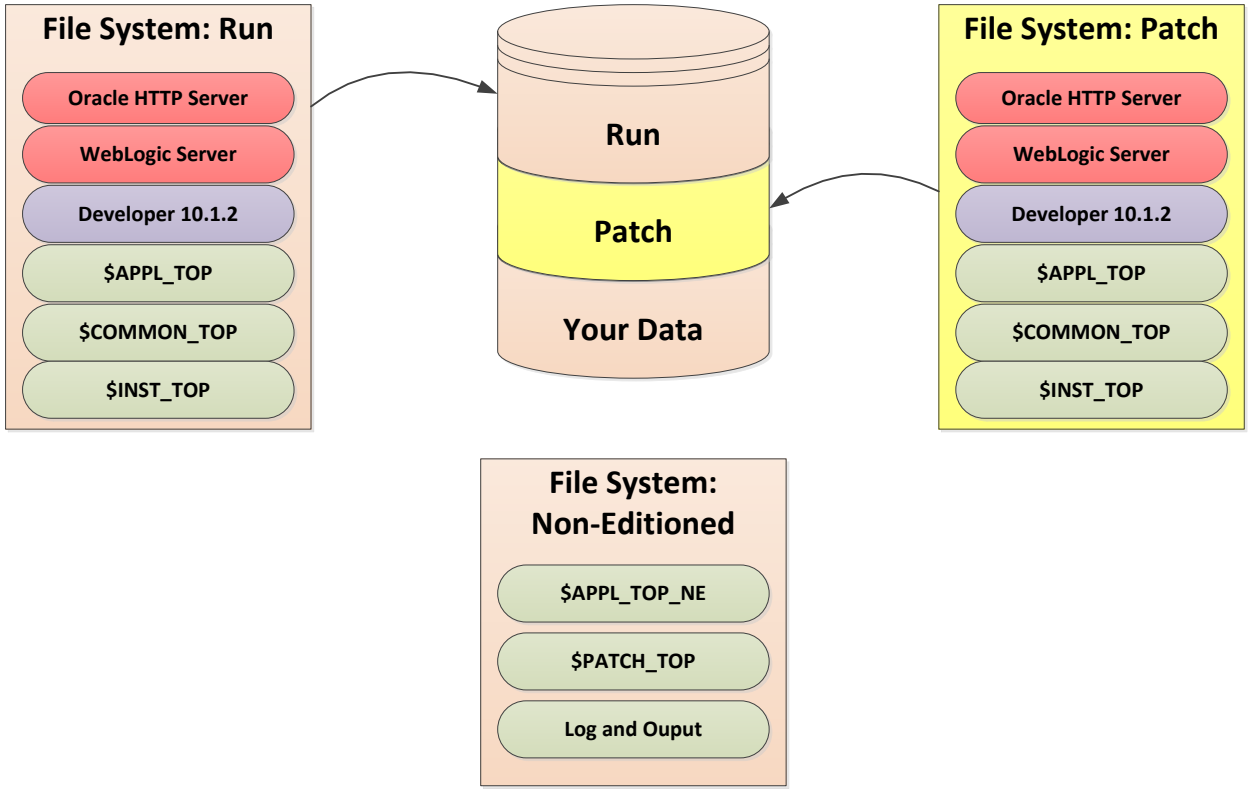
- Edition
  - Basically this is an attribute assigned to the DB session
  - You have two: Run and Patch
  - Only affect: Views, Synonyms, PL/SQL

- Editioning View
  - This provides a logical representation of the data
  - Physical table storage may be different

- Cross Edition Triggers
  - Forward Cross Edition Trigger to update the Patch data
  - Reverse Cross Edition Trigger to update the Run data

5) Understand EBS Editions

- File system structure:
  - Editioned: Run and Patch
  - Non-Editioned
- Database Objects
  - Editioned: Run and Patch
  - Your Data



5) Understand EBS Editions

- How to identify your Edition:

  OS:

```
echo $FILE_EDITION
```

  DB:

```
select ad_zd.GET_EDITION_TYPE,
       ad_zd.GET_EDITION
  from dual;
```

5) Understand EBS Editions

- How to set your Edition:

  OS:

```
source /oracle/ebs122/EBSapps.env run
or
source /oracle/ebs122/EBSapps.env patch
echo $RUN_BASE    - the run filesystem
echo $PATCH_BASE  - the patch filesystem
```

  DB:

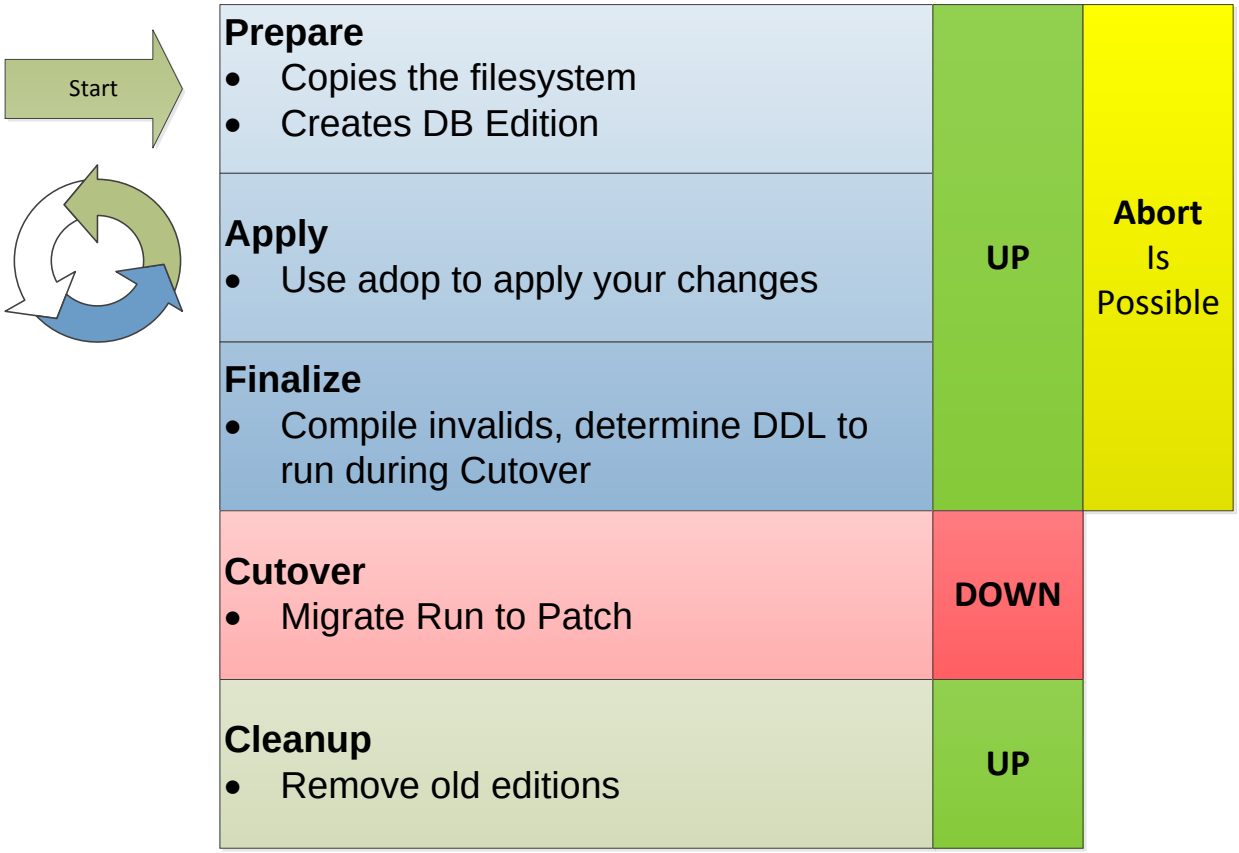
```
ad_zd.SET_EDITION('RUN')
or
ad_zd.SET_EDITION('PATCH')
```

6) Develop in Run Edition

- All development in your DEV instance will happen in that instances Run Edition
- Only Run Editions have a UI
- Be aware of patch cycles in your DEV instance

7) Understand EBS OnLine Patching

- Patch Cycle
- What happens during each step in the Patch Cycle



7) Understand EBS OnLine Patching

- How to check the status of patching, where in the cycle are you?

  OS:

  ```
  adop –status
  ```

  DB:

  ```
  sqlplus apps @ADZDSHOWED.sql
  ```

- This is especially important to verify if a patch cycle is in process in the development instance so you don’t lose you work

8) Use EBS Tools to Deploy Changes

- When deploying your changes in an Online Patching environment you won't have a UI
- So you must use the EBS tools to migrate changes

FNDLOAD

MOS Note: Tips and Examples Using FNDLOAD (Doc ID 735338.1)

XMLImporter/XMLExporter

MOS Note: How to use XMLImporter/XMLExporter to import/export personalization (Doc ID 344204.1)

WFLOAD

MOS Note: How To Use WFLOAD To Download, Upload, Upgrade, Force Upload A Workflow To Database? (Doc ID 1569004.1)

XDOLoader

MOS Note: How To Use XDOLoader to Manage, Download and Upload Files? (Doc ID 469585.1)

Concurrent Programs

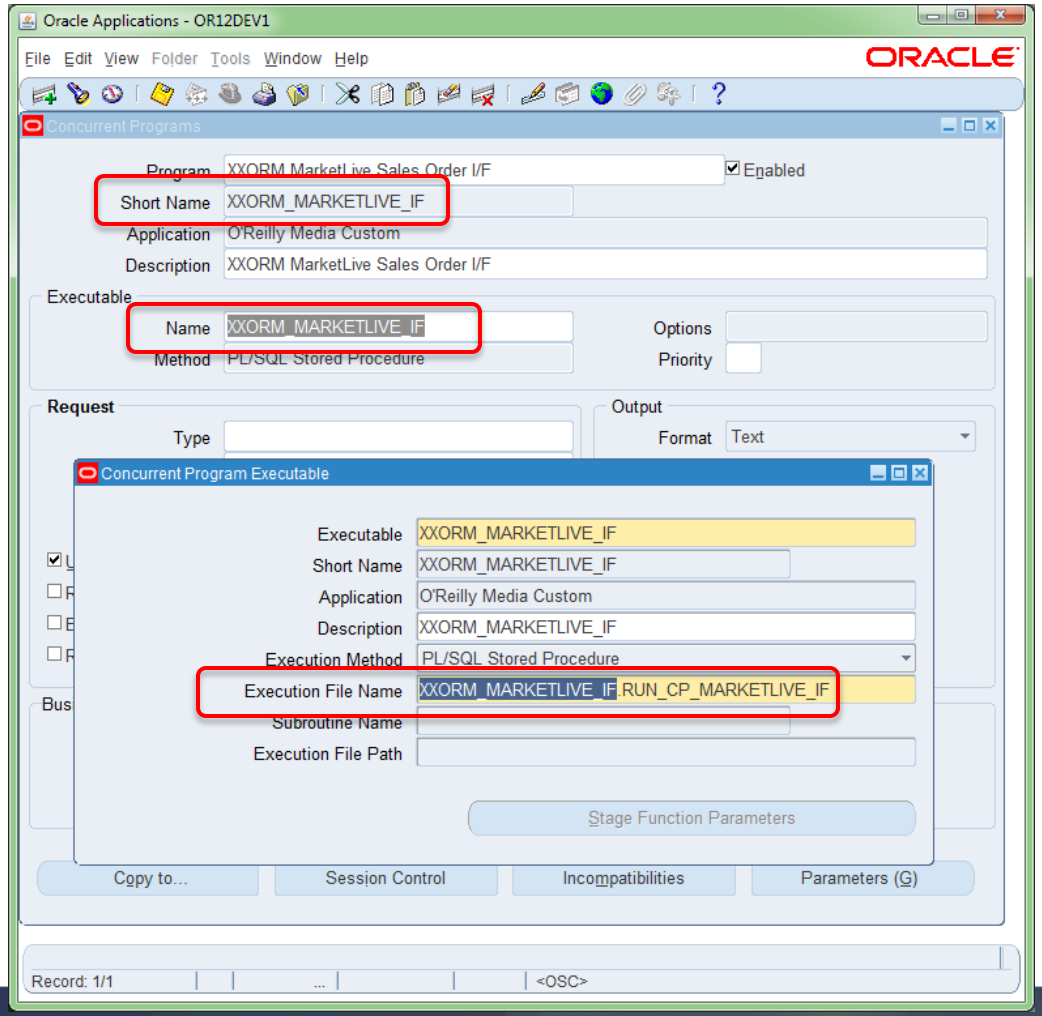
COLLABORATE16
TECHNOLOGY AND APPLICATIONS FORUM FOR THE ORACLE COMMUNITY

9) Names

- Follow a convention for the User Concurrent Program Name
  - Users clearly know what is custom
  - Users know generally what module the custom program is related to, usually a two letter abbreviation after the short name:
    - ‘XXCUST INV%’ – Inventory custom
    - ‘XXCUST GL%’ – General Ledger custom
- Program Short Name
  - Should match your Executable Name

9) Names

- Generally keep the Program Short Name same as Executable Name same as Execution File Name



10) Output File

- The output file is for the end user
- Make sure it is as clear as possible
- Always include the:
  - Parameters user submitted with
  - Request ID (fnd_global.CONC_REQUEST_ID)
  - Database Name (v$database.NAME)
  - Start Time, End Time, Calculated Run Time (min)
  - Summarized Counts at end

  Output files can be saved and then brought out later and questioned. You won't always know where the user ran it, or how.

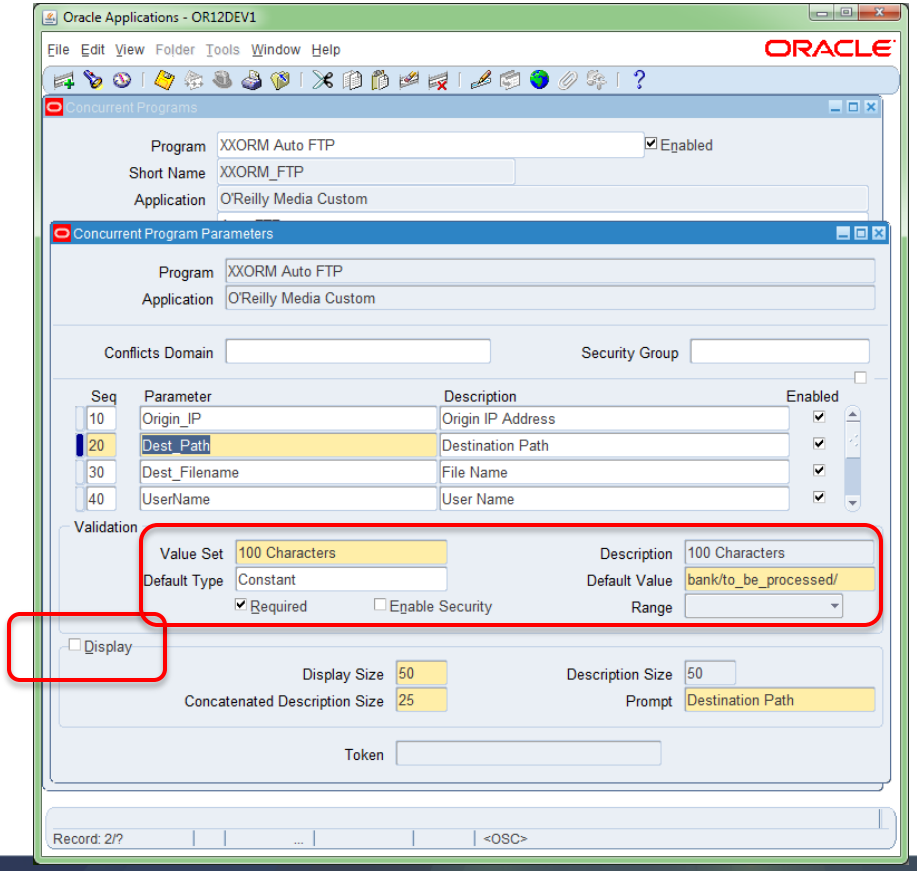
11) Log File

- The log file is for IT to debug
- Have a debug parameter to keep log files small unless in the middle of debugging
- Consider implementing a log table in the DB to hold messages, it is faster to search
- Always include the:
  - Parameters User Submitted With
  - Request ID (fnd_global.CONC_REQUEST_ID)
  - Database Name (v$database.NAME)
  - Start Time, End Time, Calculated Run Time (min)

  This is often all you will have access to since Output Files are no longer visible to all users of a responsbility

12) Parameters

- Don’t hard code values in the program code, include them in hidden Concurrent Program Parameters with default constant values.
- Better yet store them in a flexible data structure (more on that later)



Interfaces

COLLABORATE16
TECHNOLOGY AND APPLICATIONS FORUM FOR THE ORACLE COMMUNITY

13) Have DEV, TEST, PROD Support

- Always consider support for a DEV, TEST, PROD instance of EBS and corresponding external system
- Automatically know the instance in your code, don't ask the user:

  OS

  $TWO_TASK, or query from DB

  DB

  v$database.NAME

- Based on the instance you are running in have a user editable, flexible data structure that holds the required details (more on that later)

14) Interface Top Directory

- If this is a file based interface, use a common Interface Top Directory structure with directories:
  - To_Be_Processed
  - Done
  - Error
- Put this under the instance path, possibly under $APPLCSF.

  `$APPLCSF/interfaces/<if_name>/<processing>`

  Where:

  Interfaces = constant string

  <if_name> = the interface name or abbreviation

  <processing> = the subdirectories mentioned above

15) Consider Parsing File in PL/SQL

- Consider parsing your file in PL/SQL rather than using SQL*Loader
  - Much simpler and more robust error/exception handling
- Use utl_file to read the file
- Mount the interface directory structure on the DB tier
- Use DB Directories, not utl_file_dir to reference physical locations
  - These can be quickly updated during clones by the DBA using a script
- I am not a huge fan of Oracle External Tables in the database. These are a quick way to get data into the DB for testing purposes but are not robust enough for production level interfaces in my opinon.

PL/SQL Code

COLLABORATE16
TECHNOLOGY AND APPLICATIONS FORUM FOR THE ORACLE COMMUNITY

16) PL/SQL Package Version

- In all of your PL/SQL packages have a Package Version function.

```
FUNCTION PACKAGE_VERSION
  RETURN VARCHAR2
IS
BEGIN
  RETURN '2016/01/13';
END PACKAGE_VERSION;
```

- First thing in the Package Body
- This is manually maintained, usually right after the header revision comments
- Now you can query code versions quickly and compare across instances

```
select d.NAME sid,
       v.PROFILE_OPTION_VALUE site,
       xxcust_sla_interface.PACKAGE_VERSION,
       xxcust_sub_interface.PACKAGE_VERSION,
       xxcust_archiving_source.PACKAGE_VERSION,
       xxcust_archiving_dest.PACKAGE_VERSION@XXCUST_DEST_ARCH
  from FND_PROFILE_OPTION_VALUES v,
       FND_PROFILE_OPTIONS o,
       v$database d
 where o.PROFILE_OPTION_ID = v.PROFILE_OPTION_ID
   and o.PROFILE_OPTION_NAME = 'SITENAME';
```

17) Structure Your Code

- Use case and indentation to make the code quickly readable
- Either do this manually or using a tool like TOAD

```
if (x_return_status != fnd_api.g_ret_sts_success) then for i in 1 .. x_msg_count
loop fnd_msg_pub.get(p_msg_index => i,p_encoded => fnd_api.g_false, p_data => v_message,
p_msg_index_out => v_msg_index_out);
process_error_message (p_order_source_id => p_order_source_id, p_orig_sys_document_ref =>
p_orig_sys_document_ref, p_orig_sys_line_ref => p_orig_sys_line_ref, p_error_message => 'error: ' ||
v_msg_index_out); end loop; end if;
```

```
if (X_RETURN_STATUS != FND_API.G_RET_STS_SUCCESS)
then
  FOR i IN 1 .. x_msg_count
  LOOP
    fnd_msg_pub.GET(p_msg_index => i,
                    p_encoded => Fnd_Api.G_FALSE,
                    p_data => v_message,
                    p_msg_index_out => v_msg_index_out);

    PROCESS_ERROR_MESSAGE (p_order_source_id => p_order_source_id,
                           p_orig_sys_document_ref => p_orig_sys_document_ref,
                           p_orig_sys_line_ref => p_orig_sys_line_ref,
                           p_error_message => 'ERROR: ' || v_msg_index_out);
  END LOOP;
end if;
```

18) Reference Parameters in Calls

This

```
fnd_msg_pub.GET(i,
                Fnd_Api.G_FALSE,
                v_message,
                v_msg_index_out);
```

Or This

```
fnd_msg_pub.GET(p_msg_index => i,
                p_encoded => Fnd_Api.G_FALSE,
                p_data => v_message,
                p_msg_index_out => v_msg_index_out);
```

19) Create an Output Framework

- Constant Output Header format that can be used on multiple programs

```
fnd file.PUT LINE (fnd file.OUTPUT,'Running XXORM MarketLive I/F at '
                   || to_char(sysdate,'DD-MON-YYYY HH24:MI:SS')
                   || ' Request ID ' || FND_GLOBAL.CONC_REQUEST_ID);
fnd_file.PUT_LINE (fnd_file.OUTPUT,'Parameters:');
fnd file.PUT LINE (fnd file.OUTPUT,'  P ORDER_SOURCE_ID = '
                   || p_order_source_id);
fnd file.PUT LINE (fnd file.OUTPUT,'  P ORIG SYS_DOCUMENT_REF = '
                   || p_orig_sys_document_ref);
fnd_file.PUT_LINE (fnd_file.OUTPUT,'  P_INTERFACE_ORDERS = '
                   || p_interface_orders);
```

- Write to DBMS_OUTPUT or FND_FILE based on FND_GLOBAL.CONC_REQUEST_ID, if -1 then send output to DBMS_OUTPUT
- Constant Output Footer format that can be used on multiple programs

20) Create a Logging Framework

- Constant Log Header format that can be used on multiple programs

```
fnd_file.PUT_LINE (fnd_file.LOG,'Running XXORM_MARKETLIVE_IF.RUN_CP_MARKETLIVE_IF at '
                   || to_char(sysdate,'DD-MON-YYYY HH24:MI:SS')
                   || ' Request ID ' || fnd_file.CONC_REQUEST_ID);
fnd_file.PUT_LINE (fnd_file.LOG,'Parameters:');
fnd_file.PUT_LINE (fnd_file.LOG,'  P_ORDER_SOURCE_ID = '
                   || p_order_source_id);
fnd file.PUT LINE (fnd file.LOG,'  P ORIG SYS DOCUMENT_REF = '
                   || p_orig_sys_document_ref);
fnd file.PUT LINE (fnd file.LOG,'  P INTERFACE_ORDERS = '
                   || p_interface_orders);
```

- Write to DBMS_OUTPUT or FND_FILE based on FND_GLOBAL.CONC_REQUEST_ID, if -1 then send output to DBMS_OUTPUT
- Handle a global variable that holds Debug Level, this can be set from a concurrent program parameter
- Also write log messages to a custom table, use PRAGMA AUTONOMOUS_TRANSACTION so you can commit log table entries independently

21) Always Set Request Status

Ensure you program completes with:

- Normal: no issues encountered, viewing output file is optional
- Warning: issues the user can research and possibly fix, need to view output file
- Error: fatal issues, SQL or PL/SQL, unhandled exceptions, need IT assistance to resolve

Define global constants.

```
  g_retcode_normal  VARCHAR2(2000) := '0';
  g_retcode_warning VARCHAR2(2000) := '1';
  g_retcode_error   VARCHAR2(2000) := '2';

PROCEDURE SUBMIT_TEST(p_errbuf               OUT VARCHAR2,
                      p_retcode              OUT VARCHAR2,
                      p_source_name          IN  VARCHAR2,
                      p_reset_all            in varchar2
                     )
IS
  v_source_name_code      VARCHAR2(100);
BEGIN
  g_retcode := g_retcode_normal;
  g_errbuf := NULL;
  g_error_cnt := 0;
```

Put you actual code here

```
  IF (g_error_cnt > g_retcode_normal)
  THEN
    p_retcode      := g_retcode_warning;
    p_errbuf       := 'See Output File for Error Messages';
    OUT ('Please review out and log files for errors');
  ELSE
    OUT ('Test Scenarios Data Generation successful, no errors encountered');
  END IF;
END SUBMIT_TEST;
```

22) PL/SQL Result Cache

- As of Oracle 11G DB you can have your PL/SQL Function cache results based on a set of Input Parameters.
- Oracle handles this caching for you automatically in memory.
- You specify what causes the cache to be invalidated.
- Make sure your input parameters are unique enough to correctly define the values, you don't want to cache an AMOUNT column.
- Cache the highest level of data with commonality you can.
- Multiple processes all trying to add values to cache will cause Latch Waits, so be careful how you use this.

22) PL/SQL Result Cache

- Package Spec

```
FUNCTION MY_LOOKUP (p_condition1    VARCHAR2,
                    p_condition2    VARCHAR2,
                    p_condition3    VARCHAR2)
RETURN VARCHAR2
      RESULT_CACHE;
```

- Package Body

```
FUNCTION MY_LOOKUP (p_condition1    VARCHAR2,
                    p_condition2    VARCHAR2,
                    p_condition3    VARCHAR2)
  RETURN VARCHAR2
  RESULT_CACHE RELIES_ON (MTL_CATEGORIES_B)
IS
BEGIN
.....
END MY_LOOKUP;
```

Don’t Hard Code
Configure Instead

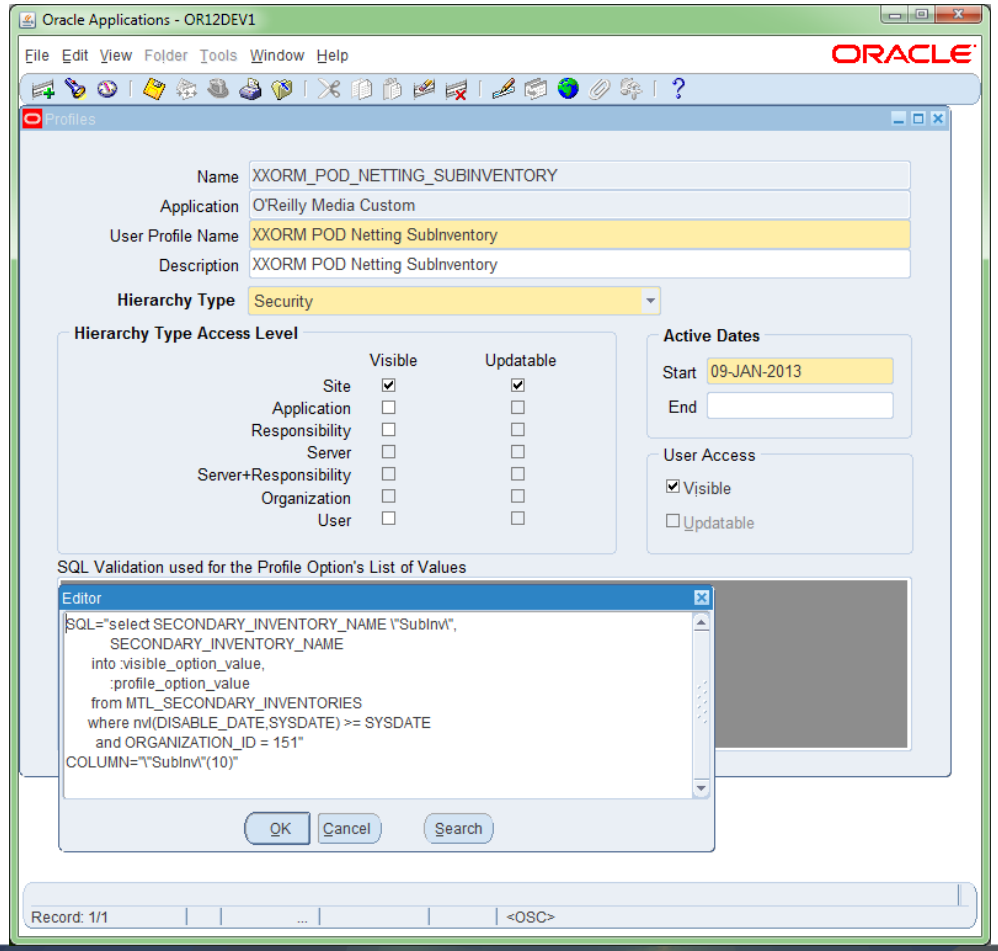
COLLABORATE16
TECHNOLOGY AND APPLICATIONS FORUM FOR THE ORACLE COMMUNITY

23) Don’t Hard Code

- If you ever find yourself assigning values in your code, either strings or numeric, ask yourself is there some place else I can get this data.
- If not seeded then consider:
  - Custom Profile Option
  - Descriptive Flex Fields, DFF
  - Key Flex Fields, Inventory : Item Categories

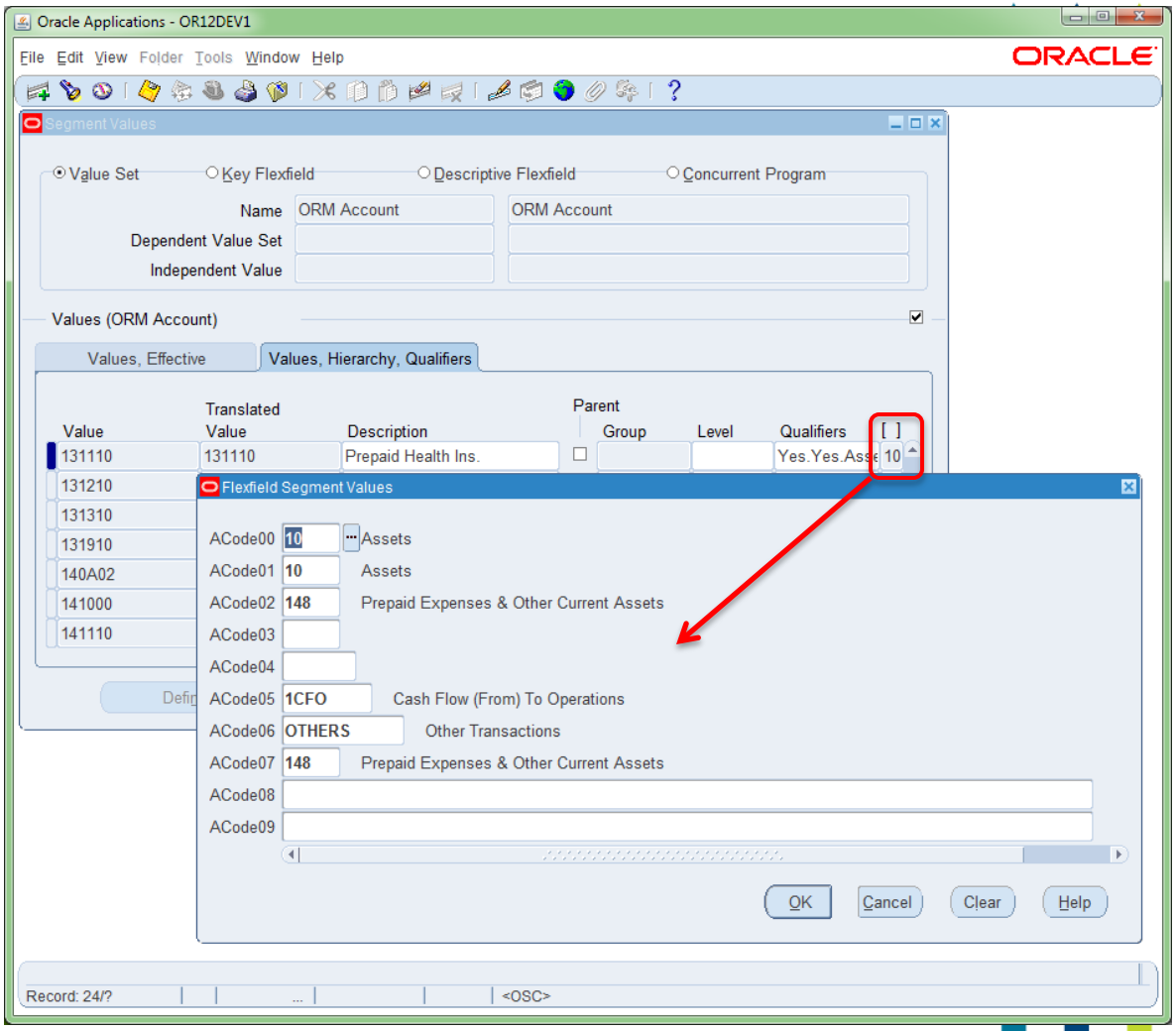
24) Custom Profile Options

- Application Developer => Profile Option
- Use the SQL Validation to validate input.



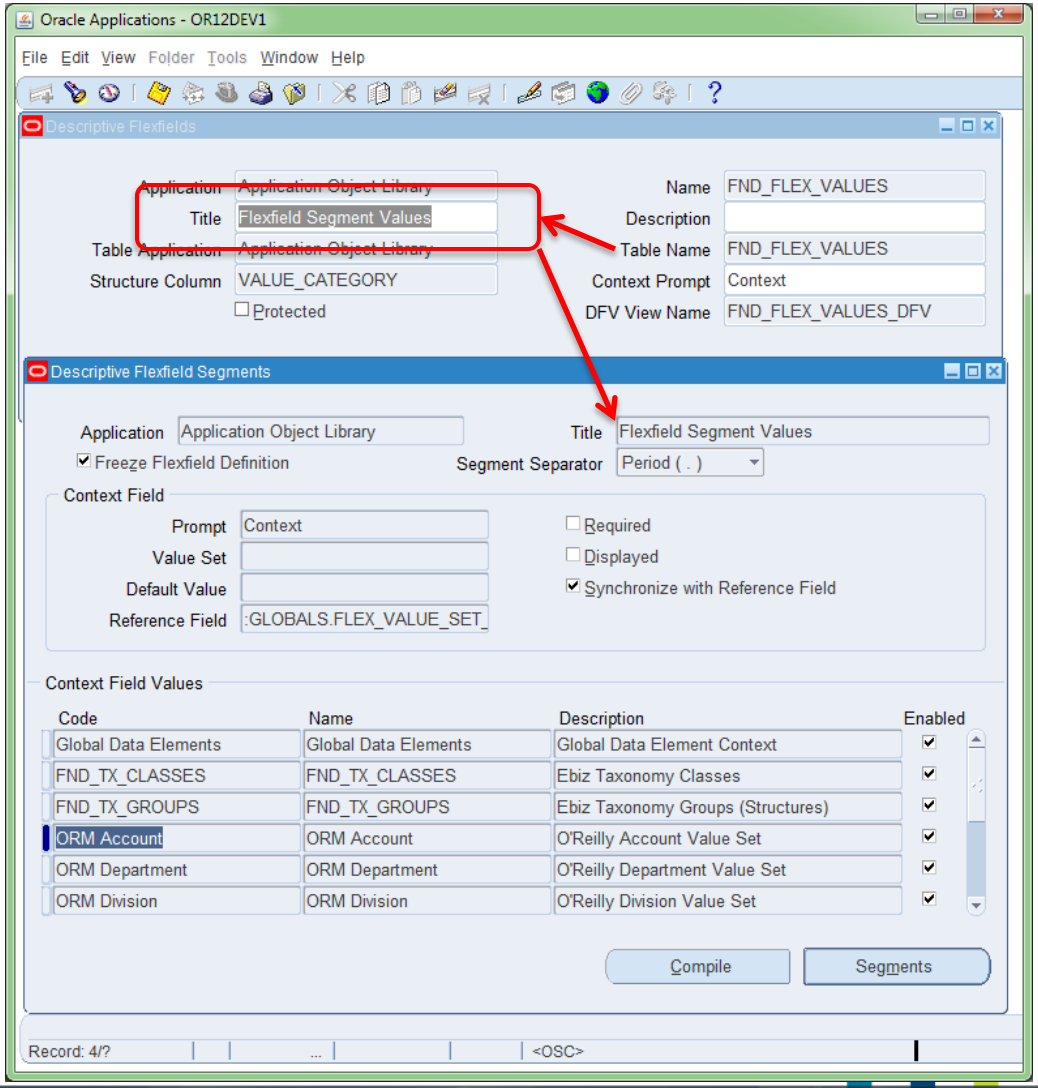
25) DFF on Value Set Values

- Obviously you can define DFF's on existing EBS data tables
- What if you want to add a DFF to variable user entered data, like a Value Set Value?



25) DFF on Value Set Values

- Value Set Values DFF is called: 'Flexfield Segment Values'
- Make this context sensitive
- Reference Field: :GLOBALS.FLEX_VALUE_SET_NAME
- Each context sensitive column can be optionally required
- Values come from Value Sets as well



25) DFF on Value Set Values

- This gives you one extra dimension to a given Value Sets Values, up to 50 additional columns of data per value
- DFF is context sensitive so different meanings for each value set
- Now when a user assigns a value to the GL Account they must fill in the DFF columns providing additional details for programs and reports to use
- This places the users in control of their data
- Removes dependency on programs or IT staff

26) Item Categories Structure

- What if you have a requirement for more than one condition columns to provide multiple result values?
- Sort of a multi-dimensional array of config data
- Then use an Inventory Item Category Structure KFF
- Don’t assign the Category Structure to an Item Category Set so it will not confuse or impact the Inventory users
- Use the Item Category Codes screen to set values
- This provides 20 segments of data that can be validated using Value Sets
- Segments are limited to character data, 40 bytes in length

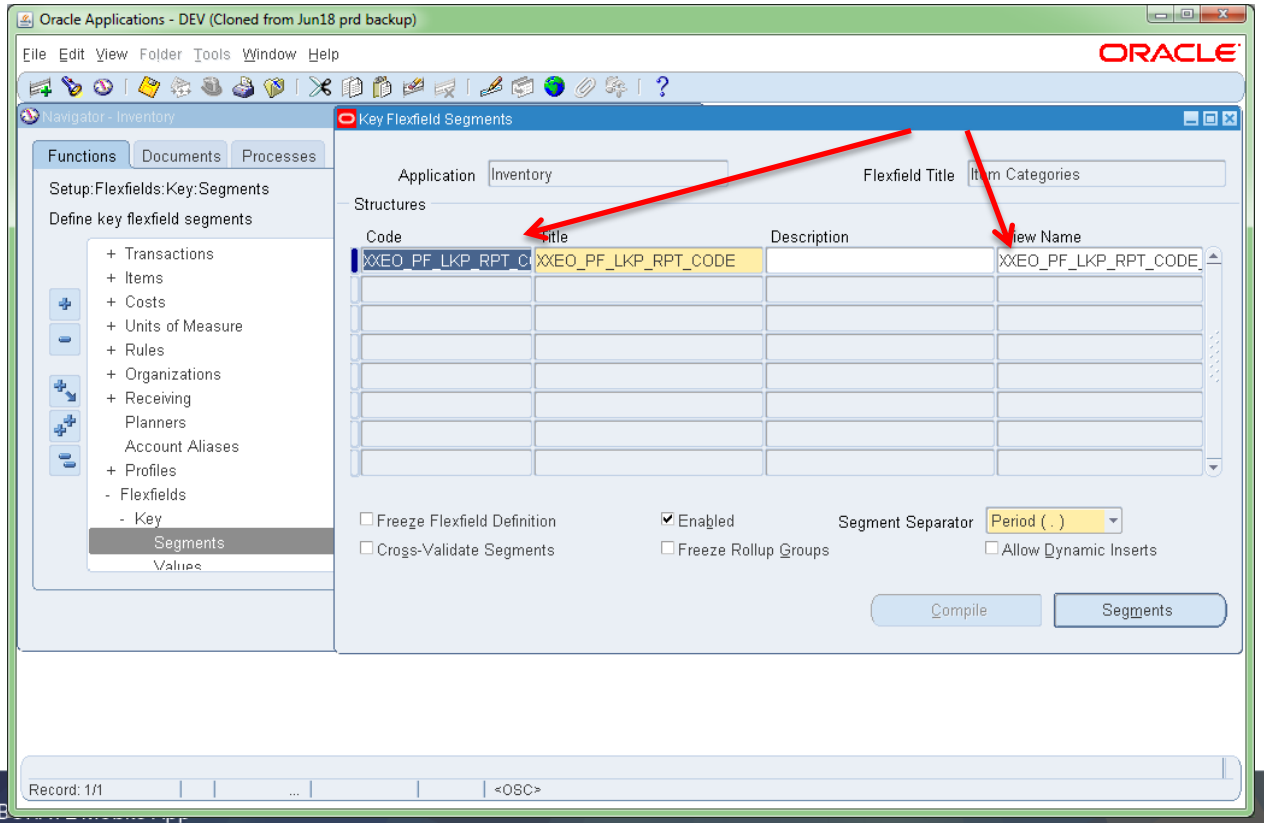
26) Item Categories Structure

- Example we have a need to have 9 condition values uniquely define 2 result values
- Segments 1-15 are being used for conditions
- Segments 16-20 are used for results
- You can vary the condition and result columns anywhere from 1 to 19.

Segment	Prompt/Description	Value Set	Required	C/R
1	TRX_TYPE	XXEO_PF_TRX_TYPE_A	Y	C
2	TRX_SUBTYPE	XXEO_PF_TRX_SUBTYPE_A	Y	C
3	ASSIGNMENT_TYPE	XXEO_PF_ASSIGNMENT_TYPE_A	Y	C
4	FULFILL_VIA	XXEO_PF_FULFILL_VIA_A	Y	C
5	PAYMENT_METHOD_CODE	XXEO_PF_PAYMENT_METHOD_CODE_A	Y	C
6	PAYMENT_METHOD_TYPE	XXEO_PF_PAYMENT_METHOD_TYPE_A	Y	C
7	ACCOUNT	XXEO_PF_ACCOUNT_A	Y	C
8	AMOUNT	XXEO_PF_SPECIAL_STRINGS	Y	C
9	VAT_COUNTRY	XXEO_PF_COUNTRIES_A	Y	C
16	Reporting Code	40 Chars	Y	R
17	Rpt Sequence Prefix	XXEO_PF_RPT_SEQUENCE_PREFIX	N	R

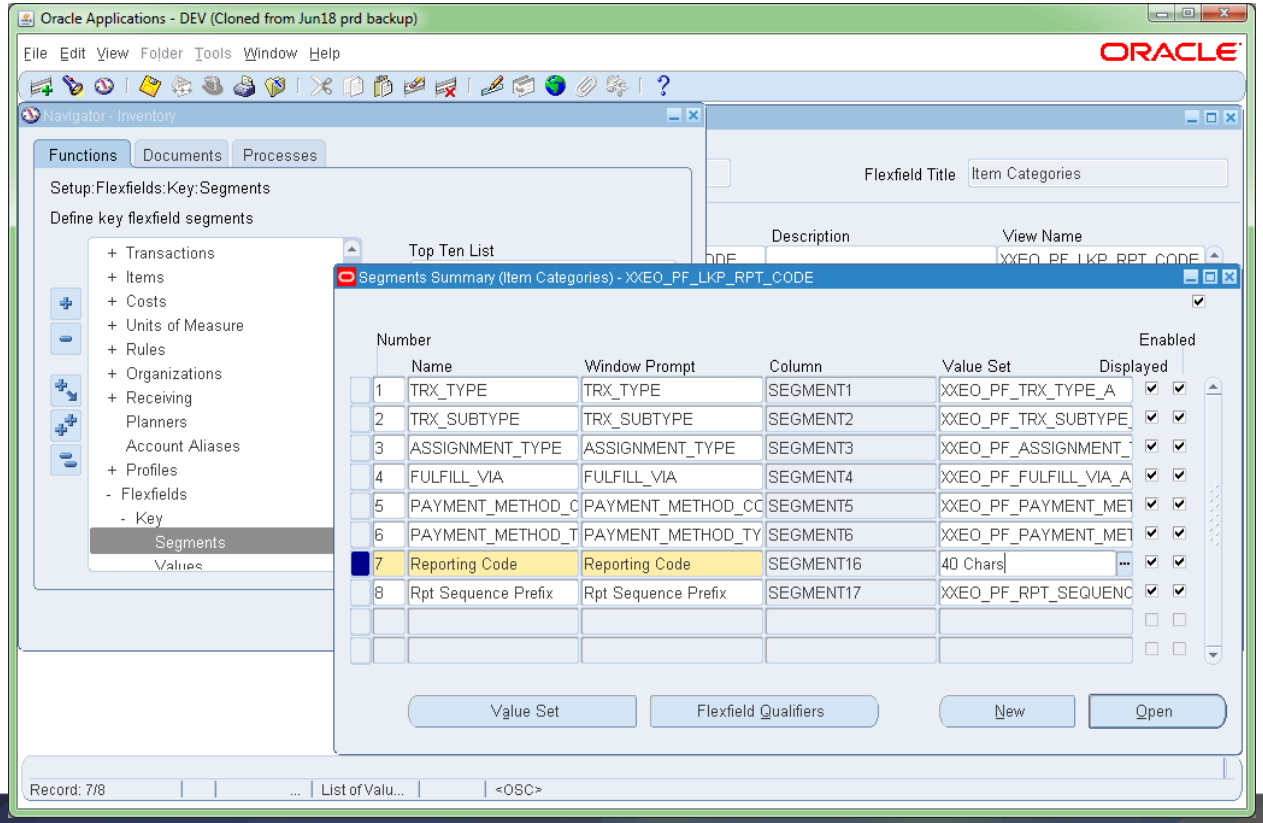
26) Item Categories Structure

- Key Flex Field => ‘Item Categories’
- Enter a View Name, and EBS will automatically build a view with column names that match you category structure
- Use a View Name like => Code || ‘_V’
  Ex: XXEO_PF_LKP_RPT_CODE => XXEO_PF_LKP_RPT_CODE_V



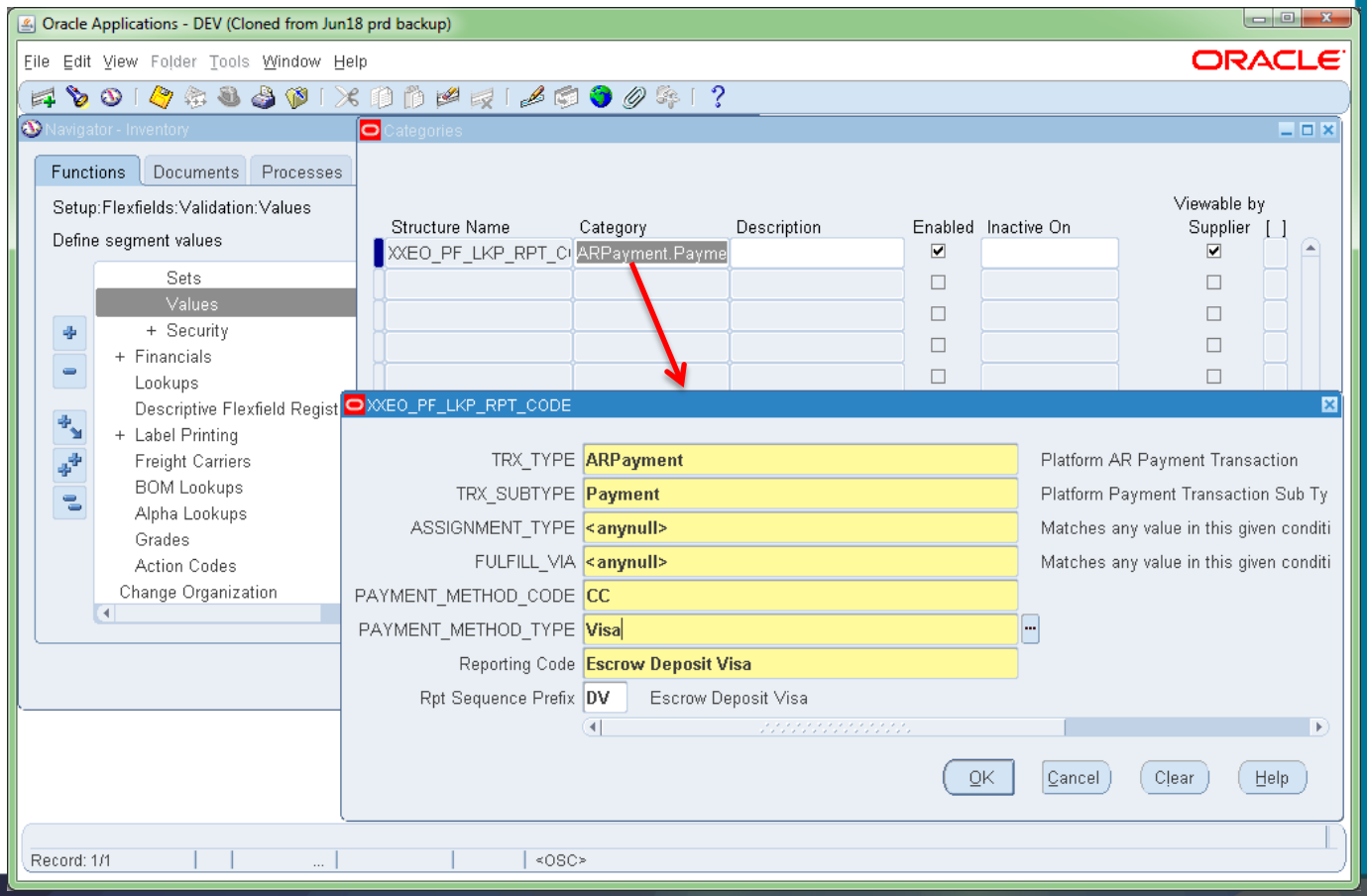
26) Item Categories Structure

- Define your Category Structure segments, 20 max
- Segments are limited to 40 characters
- Follow a convention for naming or organizing columns: conditions on left, results on right. Also convention for prompts.



26) Item Categories Structure

- Here you can see entering values in into the Category Structure Codes
- Inventory => Setup : Categories : Category Codes
- Notice the <anynull> values, more on that later



26) Item Categories Structure

- Write a single PL/SQL utility package that can be used to search a category structure and return the best matching result columns
- This can be generic because you can assume:
  - You have 1 to 19 condition columns
  - You have 1 to 19 result columns
- Mapping of the columns to your data is in the call to the generic function
- Weighted left to right, left segments are the most important conditions, this allows fuzzy searches using special strings <any>, <anynull>, etc
- Use PL/SQL Results Cache to improve performance

26) Item Categories Structure

- You can now configure instead of changing hard coded values
- You don’t need to create a custom table and form to hold a matrix of conditions and results
- Empower your users to configure the programs behavior without involving IT

27) Value Set Values

- How do we take a Value Set's values and add some static constants? I want a value set that contains EBS Currency Codes, and some special strings.
- Create a Value Set that holds the static constants
  - XXEO_PF_SPECIAL_STRINGS

String	Description
<any>	Matches any value in this given condition
<anynull>	Matches any value in this given condition, including NULL values
<null>	Matches only rows with a NULL condition
<gt0>	Greater than zero
<ge0>	Greater than or equal to zero
<eq0>	Equal to zero
<ne0>	Not Equal to zero
<le0>	Less than or equal to zero
<lt0>	Less than zero

- Create a view that unions in the base values and static constants
- Create another Value Set that is table based on the view

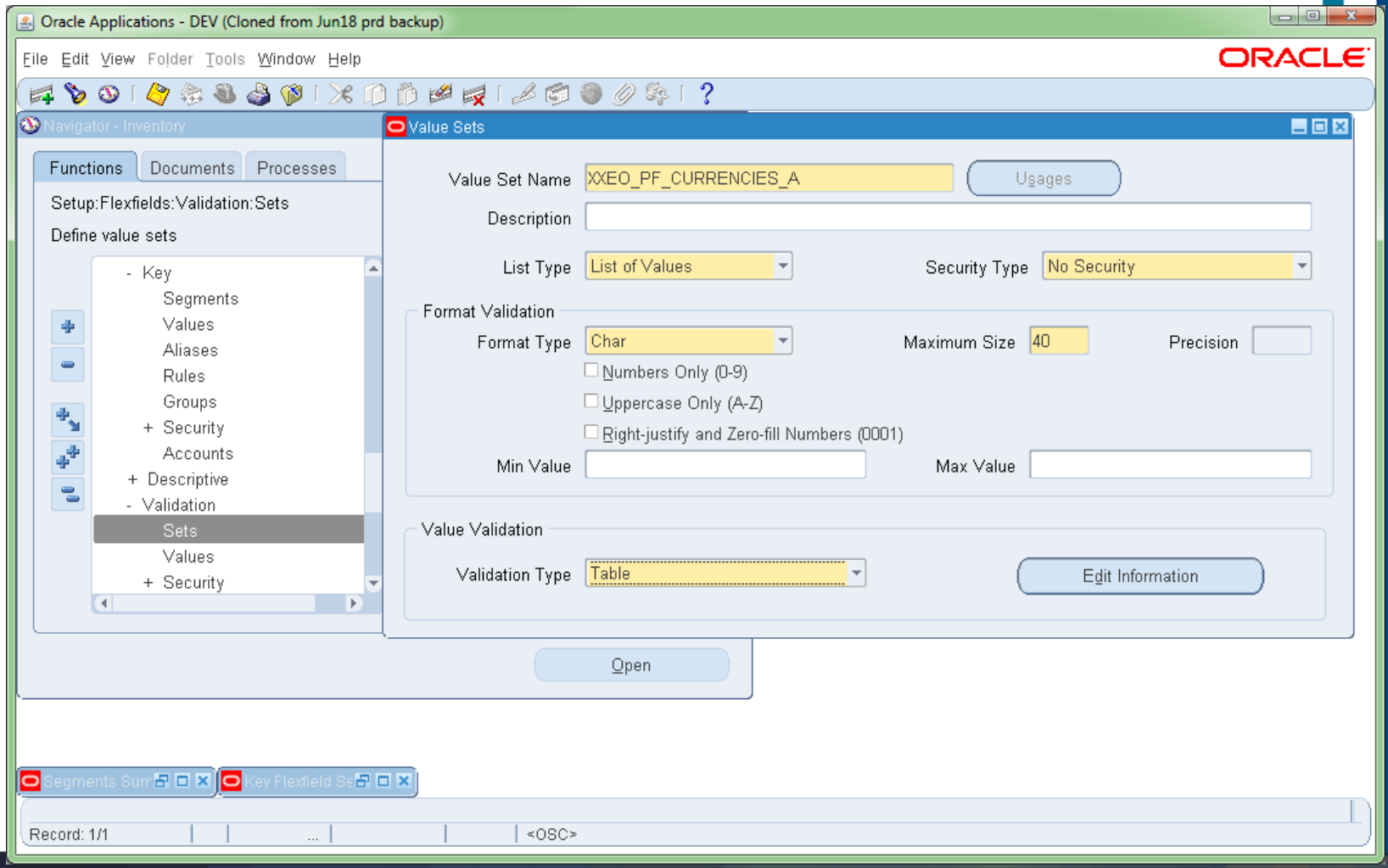
27) Value Set Values

- View XXEO_PF_CURRENCIES_A_V

```
CREATE OR REPLACE FORCE VIEW APPS.XXEO_PF_CURRENCIES_A_V
(
   FLEX_VALUE,
   DESCRIPTION
)
AS
   SELECT CURRENCY_CODE, NAME
     FROM FND_CURRENCIES_VL v
    WHERE CURRENCY_FLAG = 'Y'
      AND SYSDATE BETWEEN NVL (START_DATE_ACTIVE, SYSDATE)
                      AND NVL (END_DATE_ACTIVE, SYSDATE)
   UNION ALL
   SELECT FLEX_VALUE, DESCRIPTION
     FROM XXEO_PF_VALUE_SET_VALUES_V
    WHERE FLEX_VALUE_SET_NAME = 'XXEO_PF_SPECIAL_STRINGS';
```

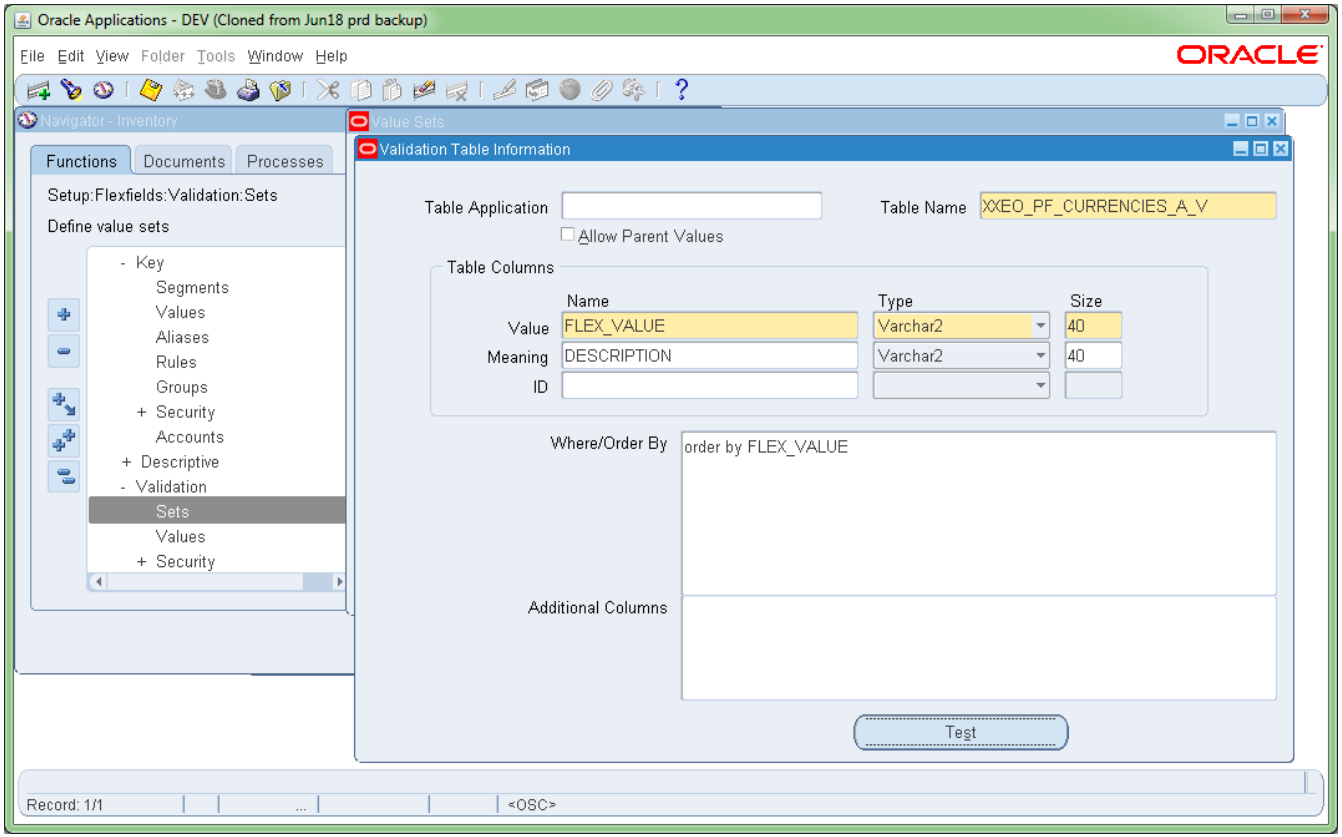
27) Value Set Values

- Value Set XXEO_PF_CURRENCIES_A
- Notice Format Type: Char, Max Size: 40



27) Value Set Values

- View columns:
  - FLEX_VALUE = Actual value code we will store in the category structure
  - DESCRIPTION = Verbose description to help the user select
  - Don't forget the Order By



27) Value Set Values

- LOV will have the constant strings first: <any>, <anynull>, etc
- Then the EBS Currency Codes
- The list will be sorted alphabetically

Use New Technology

COLLABORATE16
TECHNOLOGY AND APPLICATIONS FORUM FOR THE ORACLE COMMUNITY

28) Oracle Reports Replacement

- If you ever find yourself having to open up Oracle Report RDF to make a change ask yourself why?
- Instead:
  - Extract the SQL
  - Use BI Publisher to execute the SQL, generate an XML data stream and format using a template

- How To Migrate Reports To XML Publisher? (Doc ID 1329693.1)

29) Oracle Forms, OA Framework Pages

- If you ever need to create a custom data entry screen don't use Oracle Forms or a custom OA Framework Page, use Applications Express (APEX)
- This tool can be used to create HTML data entry screens or reports
- Deploy to desktops, tablets and smart phones
- Tightly integrates with Oracle EBS and is recommended for light weight EBS extensions by Oracle's ATG team
- Best of all it is Free if you have licensed the DB for development activities

- Extending Oracle E-Business Suite Release 12.1.3 and Above Using Oracle Application Express (APEX) MOS Note: 1306563.1 (Mar-2015)
- Extend EBS Using Applications Express http://jrpjr.com/paper_archive/Collab14_Extend_EBS_Using_APEX.pdf
- Migrate your Discover Reports to Oracle APEX http://jrpjr.com/paper_archive/collab15_disco_to_apex.pdf

30) Custom Web Application Desktop Integrators

- Consider using Custom Web Application Desktop Integrators
- Oracle now allows you to create Custom WEB ADI's
- Can be used for data entry and reporting
- Oracle® E-Business Suite Desktop Integration Framework Developer's Guide Release 12.2 Part No. E22005-10
- MOS Note: Oracle Web Applications Desktop Integrator Documentation Resources, Release 12 (Doc ID 396181.1)
- Custom SubLedger Accounting JE WebADI I/F
  http://jrpjr.com/paper_archive/collab11_custom_webadi_pres.pdf
- Custom Web ADI Integrators
  http://jrpjr.com/paper_archive/5_12_peters.pdf

References

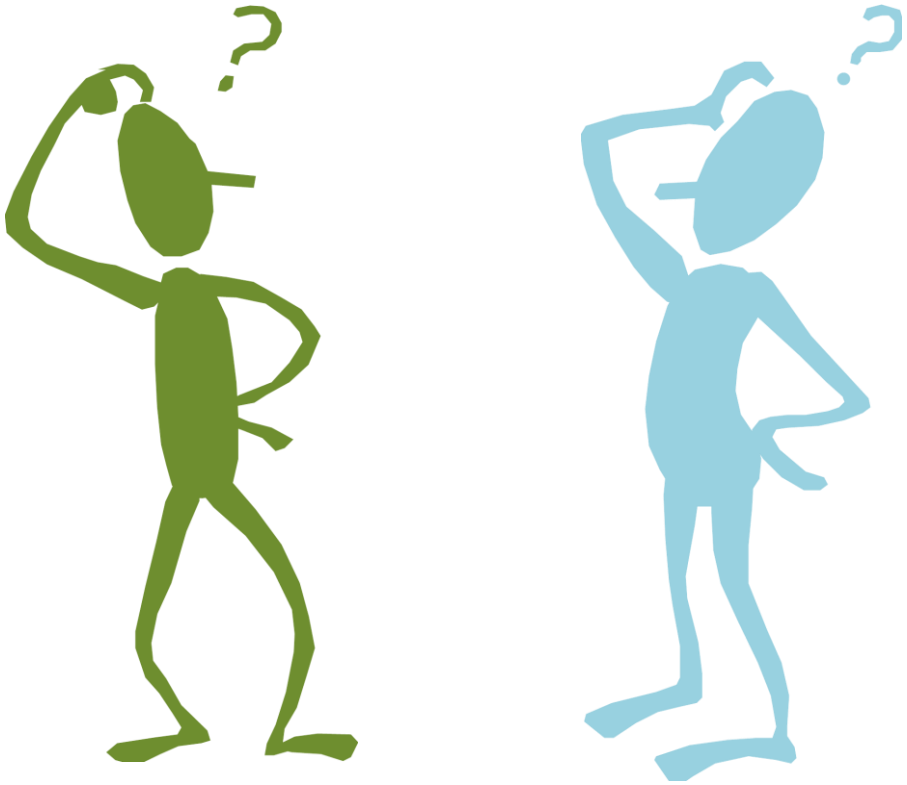

COLLABORATE16
TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY

Additional References

My Web Site and Past Presentations http://jrpjr.com

- **R12.2 Development and Deployment of Customizations**
  http://jrpjr.com/paper_archive/collab15_R12_2_dev.pdf
- **Extend EBS Using Applications Express**
  http://jrpjr.com/paper_archive/Collab14_Extend_EBS_Using_APEX.pdf
- **Migrate your Discover Reports to Oracle APEX**
  http://jrpjr.com/paper_archive/collab15_disco_to_apex.pdf
- **Custom SubLedger Accounting JE WebADI I/F**
  http://jrpjr.com/paper_archive/collab11_custom_webadi_pres.pdf
- **Custom Web ADI Integrators**
  http://jrpjr.com/paper_archive/5_12_peters.pdf

Questions and Comments?



John.Peters@jrpjr.com

Session ID 10670