# Table of Contents

**CODING STANDARDS** .......................................................................................................................... 2

CODING STANDARDS OVERVIEW ........................................................................................................ 2

OBJECTIVES ................................................................................................................................................ 2

ROLES AND RESPONSIBILITIES ........................................................................................................... 2

BUSINESS ANALYST ............................................................................................................................... 2

DEVELOPMENT TEAM ............................................................................................................................ 2

DEVELOPMENT TEAM LEAD ................................................................................................................ 2

DIRECTORIES – LOCATION OF DOCUMENTS, FORMS AND OTHER CODE ................................. 2

**DOCUMENTATION** ............................................................................................................................... 3

PROJECT DOCUMENTATION ................................................................................................................. 3

DOCUMENTATION STANDARDS ........................................................................................................... 3

HEADER .................................................................................................................................................... 3

FOOTER ..................................................................................................................................................... 3

TITLE PAGE ............................................................................................................................................... 4

TABLE OF CONTENTS ............................................................................................................................ 5

DOCUMENT BODY .................................................................................................................................. 6

APPENDIXES ............................................................................................................................................ 7

**SQL AND PL/SQL CODING STANDARDS** ......................................................................................... 7

HEADER .................................................................................................................................................... 7

INDENTATION ........................................................................................................................................... 8

NESTED BLOCKS ...................................................................................................................................... 8

STATEMENTS ........................................................................................................................................... 8

DECLARATIONS ....................................................................................................................................... 9

VARIABLES ............................................................................................................................................... 9

MULTILINE STATEMENTS ....................................................................................................................... 10

SQL ......................................................................................................................................................... 10

IF STATEMENTS ...................................................................................................................................... 10

LOOPS .................................................................................................................................................... 11

SIMPLE LOOP ....................................................................................................................................... 11

WHILE LOOP ....................................................................................................................................... 11

FOR LOOPS .......................................................................................................................................... 11

EXCEPTION HANDLERS .......................................................................................................................... 11

PACKAGES ............................................................................................................................................. 12

PROCEDURES AND FUNCTIONS ......................................................................................................... 12

COMMENTS ............................................................................................................................................ 13

DATABASE INDEXES .............................................................................................................................. 13

**3 & 4 GL CODING STANDARDS** ........................................................................................................... 13

**FORMS** ............................................................................................................................................... 13

SCREEN FORMATS .................................................................................................................................. 13

SCREEN TITLE ........................................................................................................................................ 13

WINDOW & CANVAS PROPERTIES ........................................................................................................ 14

BLOCK PROPERTIES ............................................................................................................................ 14

ITEM PROPERTIES .................................................................................................................................. 14

RECORD GROUPS ................................................................................................................................... 14

LOVs ......................................................................................................................................................... 14

BUTTON NAMING STANDARDS .............................................................................................................. 14

PUSH BUTTONS ...................................................................................................................................... 15

CHECK BOXES ....................................................................................................................................... 15

POP LISTS .............................................................................................................................................. 15

RADIO GROUPS ...................................................................................................................................... 15
CODING STANDARDS

CODING STANDARDS OVERVIEW

The purpose of a coding standard is to make application source code more readable and easier to debug and maintain. Whenever code is developed these standards should be utilized.

OBJECTIVES

The main objectives of Coding Standards are:
- Provide consistency in all aspects of the application process, i.e., Documentation, Coding Conventions, Functionality, and Usability
- Implement application code which is both effective and efficient
- Minimize time spent researching code functionality and rationale
- Provide documentation of the Software Project Plan (including the Project Objective Statement and Project Framework); User, System, and Software Requirements Specifications; Software and Test Design Specifications; Technical and User Guides
- Ensure all documentation is complete

ROLES AND RESPONSIBILITIES

An overview of the roles and responsibilities associated the Coding Standards is presented below.

BUSINESS ANALYST

Business Analyst is a user assigned to a project team. Their responsibilities include creating the User Guide, assisting the development team in the creation of the Unit Test Plan, and assisting developers by clarifying User Requirements.

DEVELOPMENT TEAM

Development teams participate in the modification/enhancement process by making the Technical Guide, adhering to Coding Standards procedures, and working to meet the various requirements specifications. They are key players in the development process who actually build the Unit Test Plan.

DEVELOPMENT TEAM LEAD

The development team lead has ultimate responsibility to ensure that their team provides adequate documentation (specifically the Technical Guide and Unit Test Plan). It is also the responsibility of the team lead to define User, System, and Software Requirements Specifications. The team lead produces the Software Design Specification based on the Requirements Specifications. The team lead also participates in the approval and scheduling of the Software Project Plan. Another key responsibility of the team lead is to ensure their team understands and adheres to Coding Standards.

DIRECTORIES – LOCATION OF DOCUMENTS, FORMS AND OTHER CODE

All documentation supporting the Project Management Process for application development is located on the server, under M:\application development group\%

\Project Documents Any informational documents such as instructions and procedures. The directory includes Project Plan; User, System, and Software Requirements Specifications; Technical and User Guides; and Unit Test Plans.

\Standards DOC files containing coding, process, and procedure standards.
Master Forms

Templates for all forms. They include design specifications, requirements specifications, technical and user guides.

Test

Test cases identified for unit testing.

DOCUMENTATION

Documentation is maintained during the entire process from the time an issue is identified or requested to the point of installation.

PROJECT DOCUMENTATION

In addition to the documentation used in the Change Management process, Project Documentation consists of the following:

<table>
<thead>
<tr>
<th>Document Name</th>
<th>Document Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Project Plan</td>
<td>Must: communicate scope and resources; define risks and offer contingency and mitigation plans; include Project Objective Statement, Flexibility Matrix, Project Definition Document, and Project Framework. Addressed to a diverse audience.</td>
</tr>
<tr>
<td>System Requirements Specification</td>
<td>Outlines the entire system requirements (hardware, software, resources, etc.).</td>
</tr>
<tr>
<td>Software Requirements Specification</td>
<td>Contains detailed software application requirements. This defines WHAT the software application is required to do, not how it is to be done.</td>
</tr>
<tr>
<td>Software Design Specification</td>
<td>Contains the architectural and data design of the software application.</td>
</tr>
<tr>
<td>Technical Guide</td>
<td>Contains information for the coverage practice as well as information for the help desk covering basic technical and troubleshooting aspects of the project.</td>
</tr>
<tr>
<td>User Guide</td>
<td>Contains information on how to use the product from an end-user perspective.</td>
</tr>
</tbody>
</table>

DOCUMENTATION STANDARDS

All documentation will conform to the UVSC Style and Format Standards. This requires that all documentation include the following:

HEADER

Each page of the document must provide a header.

The header is a 2x2 table and contains the following information in the order specified. Also, the header format alternates between even and odd pages.
The width of column 1 is 2". The width of column 2 is 4.5". The spacing between columns is 0.15". Column 1, Row 1 contains the UVSC logo, left justified. Column 2, Row 1 is formatted as follows:

<table>
<thead>
<tr>
<th>Line #</th>
<th>Description</th>
<th>Format</th>
<th>Justification</th>
<th>Font</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line 1</td>
<td>UTAH VALLEY STATE COLLEGE</td>
<td>Upper case text</td>
<td>Right alignment</td>
<td>9 pt, Times Roman</td>
<td></td>
</tr>
<tr>
<td>Line 2</td>
<td>OREM, UTAH</td>
<td>Upper case text</td>
<td>Right alignment</td>
<td>9 pt, Times Roman</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Column 1, Row 2 and Column 2, Row 2 both contain only a ¾ pt top line border.

Odd Number Pages

UTAH VALLEY UNIVERSITY
OREM, UTAH

The width of column 1 is 2". The width of column 2 is 4.5". The spacing between columns is 0.15". Column 2, Row 1 contains the UVSC logo, right justified. Column 1, Row 1 is formatted as follows:

<table>
<thead>
<tr>
<th>Line #</th>
<th>Description</th>
<th>Format</th>
<th>Justification</th>
<th>Font</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line 1</td>
<td>UTAH VALLEY STATE COLLEGE</td>
<td>Upper case text</td>
<td>Left alignment</td>
<td>9 pt, Times New Roman</td>
<td></td>
</tr>
<tr>
<td>Line 2</td>
<td>OREM, UTAH</td>
<td>Upper case text</td>
<td>Left alignment</td>
<td>9 pt, Times New Roman</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Column 1, Row 2 and Column 2, Row 2 both contain only a ¾ pt top line border.

**FOOTER**

Each page of the document must provide a footer.

The footer is left justified, consists of a ¾ pt top line border and is indented 0.5" from both left and right margins.

© 2004 UTAH VALLEY STATE UNIVERSITY VERSION A May 10, 2004 11:12 AM 1 of 4

The footer will contain the following information in the order specified and is formatted as follows:

<table>
<thead>
<tr>
<th>Line #</th>
<th>Description</th>
<th>Format</th>
<th>Justification</th>
<th>Font</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line 1</td>
<td>Copyright symbol ©</td>
<td>Symbol</td>
<td>Left alignment</td>
<td>12 pt, Times New Roman</td>
<td></td>
</tr>
<tr>
<td>Line 1</td>
<td>Year</td>
<td>Numeric</td>
<td>Left alignment</td>
<td>12 pt, Times New Roman</td>
<td>Use YYYY format for year (e.g., 2004)</td>
</tr>
<tr>
<td>Line 1</td>
<td>UTAH VALLEY STATE COLLEGE</td>
<td>Upper case text</td>
<td>Left alignment</td>
<td>12 pt, Times New Roman</td>
<td></td>
</tr>
<tr>
<td>Line 1</td>
<td>VERSION (X) (followed by 3 spaces)</td>
<td>Upper case text</td>
<td>Left alignment</td>
<td>12 pt, Times New Roman</td>
<td>Let (X) represent the document version with an alpha character (A-Z)</td>
</tr>
<tr>
<td>Line 1</td>
<td>Date/Time Stamp (followed by 3 spaces)</td>
<td>Date/Time field</td>
<td>Left alignment</td>
<td>12 pt, Times New Roman</td>
<td></td>
</tr>
<tr>
<td>Line 1</td>
<td>Page # of Total #</td>
<td># of #</td>
<td>Left alignment</td>
<td>12 pt, Times New Roman</td>
<td>Page numbers begin on the first page subsequent to the Table of Contents page.</td>
</tr>
</tbody>
</table>

**Note:** Footer is indented 0" from the left margin and 0.25" from the right margin.

**TITLE PAGE**
Page Setup

The top margin is defined at 2”, while the bottom, left, and right margins are defined at 1”. The margin standard for header and footer is defined as 0.67” from edge. Also, “Different Odd and Even” and “Different First Page” options are defined for the header and footer in Page Setup-Layout. The format for Paper Size is “Letter 8.5x11 inches.” Also as a standard the paper orientation is “Portrait” and the vertical alignment is set to “Center.”

Document Title

Name of Project

The Title Page of all documentation is formatted as follows:

<table>
<thead>
<tr>
<th>Line #</th>
<th>Description</th>
<th>Format</th>
<th>Justification</th>
<th>Font</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line 1</td>
<td>Document Title</td>
<td>Mixed case text</td>
<td>Centered</td>
<td>32 pt, bold Times New Roman</td>
<td></td>
</tr>
<tr>
<td>Line 2</td>
<td>Blank</td>
<td></td>
<td></td>
<td>32 pt, Times New Roman</td>
<td></td>
</tr>
<tr>
<td>Line 3</td>
<td>Name of Project</td>
<td>Mixed case text</td>
<td>Centered</td>
<td>24 pt, Times New Roman</td>
<td></td>
</tr>
<tr>
<td>Line 4</td>
<td>Blank</td>
<td></td>
<td></td>
<td>24 pt, Times New Roman</td>
<td></td>
</tr>
<tr>
<td>Line 5</td>
<td>Blank</td>
<td></td>
<td></td>
<td>10 pt, Times New Roman</td>
<td></td>
</tr>
<tr>
<td>Line 6</td>
<td>Creation Date:</td>
<td>Mixed case text</td>
<td>Centered</td>
<td>10 pt, Times New Roman</td>
<td>Date/time field after text with Month DD, YYYY HH:MI AM format.</td>
</tr>
<tr>
<td>Line 7</td>
<td>Last Updated:</td>
<td>Mixed case text</td>
<td>Centered</td>
<td>10 pt, Times New Roman</td>
<td>Date/time field after text with Month DD, YYYY HH:MI AM format.</td>
</tr>
</tbody>
</table>

TABLE OF CONTENTS

Page Setup

The top margin is defined at 2”, while the bottom, left, and right margins are defined at 1”. The margin standard for header and footer is defined as 0.67” from edge. Also, “Different Odd and Even” and “Different First Page” options
are defined for the header and footer in Page Setup-Layout. The format for Paper Size is “Letter 8.5x11 inches.”
Also as a standard the paper orientation is “Portrait” and the vertical alignment is set to “Top.”

### Table of Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
<th>Format</th>
<th>Justification</th>
<th>Font</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Topic</td>
<td>Upper case text</td>
<td>Left alignment</td>
<td>10 pt, Times New Roman</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sub-Topic</td>
<td>Upper case text</td>
<td>Left alignment</td>
<td>10 pt, Times New Roman</td>
<td>0.5” left indentation.</td>
</tr>
<tr>
<td></td>
<td>Sub-Topic Detail</td>
<td>Upper case text</td>
<td>Left alignment</td>
<td>10 pt, Times New Roman</td>
<td>1” left indentation.</td>
</tr>
</tbody>
</table>

**Note:** A new page must be created each time a new topic is introduced.

### DOCUMENT BODY

**Page Setup**

All margins are defined at 1”. The margin standard for header and footer is defined as 0.67” from edge. Also, “Different Odd and Even” and “Different First Page” options are defined for the header and footer in Page Setup-Layout. The format for Paper Size is “Letter 8.5x11 inches.” Also as a standard the paper orientation is “Portrait” and the vertical alignment is set to “Top.”

### TOPIC

Topic description.

### SUB-TOPI

Paragraph text.

### SUB-TOPI DETAIL

Paragraph text.

The Document Body for all documentation is formatted as follows:

<table>
<thead>
<tr>
<th>Line #</th>
<th>Description</th>
<th>Format</th>
<th>Justification</th>
<th>Font</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line 1</td>
<td>TOPIC</td>
<td>Upper case text</td>
<td>Left alignment</td>
<td>14 pt, bold Times New Roman</td>
<td></td>
</tr>
<tr>
<td>Line 2</td>
<td>Blank</td>
<td></td>
<td></td>
<td>10 pt, Times New Roman</td>
<td></td>
</tr>
</tbody>
</table>
APPENDIXES

Page Setup

All margins are defined at 1”. The margin standard for header and footer is defined as 0.67” from edge. Also, “Different Odd and Even” and “Different First Page” options are defined for the header and footer in Page Setup-Layout. The format for Paper Size is “Letter 8.5x11 inches.” Also as a standard the paper orientation is “Portrait” and the vertical alignment is set to “Top.”

APPENDIX

Paragraph Text

The Appendixes for all documentation is formatted as follows:

<table>
<thead>
<tr>
<th>Line #</th>
<th>Description</th>
<th>Format</th>
<th>Justification</th>
<th>Font</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line 1</td>
<td>APPENDIX</td>
<td>Upper case text</td>
<td>Left alignment</td>
<td>14 pt, bold</td>
<td>10 pt, Times New Roman</td>
</tr>
<tr>
<td>Line 2</td>
<td>Blank</td>
<td></td>
<td></td>
<td>10 pt, Times New Roman</td>
<td></td>
</tr>
</tbody>
</table>

SQL AND PL/SQL CODING STANDARDS

Oracle’s PL/SQL has been chosen as the tool of choice for developing SQL applications. The following details the standards to be followed when developing an application using PL/SQL. Code efficiency and code simplicity go hand in hand. Don’t sacrifice clarity, readability, or correctness for nonessential improvements. Anyone caught using a “GOTO” will be severely punished.

HEADER

Each module must contain a header. The header will contain the following information in the order specified:

/*************************************************************************************
Project Name:
Purpose:
Interface Description:
   Calling Sequence:
   Arguments:
   Subordinate modules:
Development History

<table>
<thead>
<tr>
<th>DATE</th>
<th>DEVELOPER INITIALS</th>
<th>MODIFICATION DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

********************************************************************************************/

Always make sure that purposes and descriptions follow the three C’s, i.e., be Clear, Concise, and Complete.

**INDENTATION**

Indentation will be 2 spaces, except for Nested Blocks which will be 3 spaces. Here is a comparison of an IF statement without indentation and one with indentation:

**IF without Indentation**

```plaintext
IF TO_NUMBER(v_Value) > 22 AND num1 BETWEEN lval AND hval THEN
  v_NewVal := 100;
ELSIF TO_NUMBER(v_Value) < 1 THEN
  calc_tots(TO_DATE('12-jan-95'));
ELSE
  clear_vals
END IF;
```

**IF with Indentation**

```plaintext
IF TO_NUMBER(v_Value) > 22 AND num1 BETWEEN lval AND hval THEN
  v_NewVal := 100;
ELSIF TO_NUMBER(v_Value) < 1 THEN
  calc_tots(TO_DATE('12-jan-95'));
ELSE
  clear_vals
END IF;
```

**NESTED BLOCKS**

Nested blocks will be indented a minimum of 3 spaces. For example:

```plaintext
BEGIN
  expression;
BEGIN
  expression;
EXCEPTION
  WHEN NO DATA_FOUND THEN
  expression;
END;
END;
```

**STATEMENTS**

Use one statement per line. Consider the following statement:

```plaintext
new_id := 15;  max_dollars := 105;  sales_tax := .0625;
```

Would look better as:

```plaintext
new_id := 15;
max_dollars := 105;
sales_tax := .0625;
```

Use whitespace inside a statement. Consider the following statement:

```plaintext
WHILE(total_sales<maximum_sales AND company_type = 'NEW')LOOP
```

Would look better as:

```plaintext
WHILE total_sales<maximum_sales AND company_type = 'NEW' LOOP
```
WHILE (total_sales < maximum_sales AND company_type = 'NEW') LOOP

Use spaces to make module calls and their parameter lists more understandable. Consider the following statement:

calc_totals(company_id,LAST_DAY(end_of_year_date),total_type);

Would look better as:

calc_totals (company_id, LAST_DAY(end_of_year_date), total_type);

Parenthesize all expressions that evaluate to a single value up to the outermost expression. For example:

monthly_total := (daily_val * 365.25) / 12;

IF (expression AND (expression OR expression)) THEN

DECLARATIONS

Place one declaration per line. Align declarations and initial values. Here is a comparison of a Declaration without alignment and one with alignment:

<table>
<thead>
<tr>
<th>Without Alignment</th>
<th>With Alignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECLARE</td>
<td>DECLARE</td>
</tr>
<tr>
<td>company VARCHAR2(30) := 'CDSI';</td>
<td>company VARCHAR2(30) := 'CDSI';</td>
</tr>
<tr>
<td>id VARCHAR2(30) := 'ES';</td>
<td>id VARCHAR2(30) := 'ES';</td>
</tr>
<tr>
<td>hiredate DATE;</td>
<td>hiredate DATE;</td>
</tr>
</tbody>
</table>

VARIABLES

All variables will begin with v_ followed by a mixed case title, except global variables which will begin with g_. Upper case the first letter of each word in the title. The v_ is used to distinguish between a variable and module call or column name. There should be only one underscore in a variable name. Make variable names meaningful. Alphabetize the variables when you are declaring more than one in a single statement.

Here is a comparison (which is the function, column name, and variable in the first example?):

<table>
<thead>
<tr>
<th>Without Standards</th>
<th>With Standards</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECLARE</td>
<td>DECLARE</td>
</tr>
<tr>
<td>total_number_of_widgets NUMBER(30);</td>
<td>v_AnotherWidget VARCHAR2(30);</td>
</tr>
<tr>
<td>another_widget VARCHAR2(30);</td>
<td>v_MoreWidgets NUMBER(2);</td>
</tr>
<tr>
<td>more_widgets NUMBER(2);</td>
<td>v_TotalNumberOfWidgets NUMBER(30);</td>
</tr>
<tr>
<td>BEGIN</td>
<td>BEGIN</td>
</tr>
<tr>
<td>total_number_of_widgets(total_number_of_widgets,total_number_of_widgets);</td>
<td>total_number_of_widgets (total_number_of_widgets, v_TotalNumberOfWidgets);</td>
</tr>
<tr>
<td>END;</td>
<td>END;</td>
</tr>
</tbody>
</table>
MULTILINE STATEMENTS

Use indentation to offset all continuation lines under the first line by lining up parameters. Consider the following statement:

```java
generate_company_statistics (company_id, last_year_date, rollup_type, total, average, variance, budgeted, next_year_plan);
```

Would look better as:

```java
generate_company_statistics (company_id, last_year_date, rollup_type, total, average, variance, budgeted, next_year_plan);
```

SQL

The compiler takes Case (upper or lower) into consideration when storing statements into the shared pool. For instance, even though the following statements are identical in functionality they are stored as separate statements because they were written using different case:

```sql
select sysdate from dual;
SELECT sysdate FROM dual;
```

Because we want to be as efficient as possible with regards to the database, we will utilize the following standards:

All reserved words should be capitalized.

Each line of a WHERE clause should have its own line. For example:

```sql
WHERE AP.part_ref = SA.part_ref
AND AP.cage_code = '81205'
```

All aliases should be meaningful (i.e., not A, B, C). Consider the following statement:

```sql
FROM serial_asset SA, assess_part AP, part_category PC
WHERE SA.part_ref = AP.part_ref;
```

On a subquery, align the keywords under the first select statement as though there were a pseudo whitespace column. For example:

```sql
SELECT serial_no
FROM serial_asset SA
WHERE serial_ref IN
  (SELECT serial_ref
   FROM nha_serial NS
   WHERE NS.serial_ref = SA.serial_ref
   AND removal_date IS NULL);
```

IF STATEMENTS

Expression is the term used for the logic between the IF and THEN keywords. For example: `emp_ref <> '123'

When using a single expression, place the THEN on the same line. For example:
IF <expression> THEN
    executable_statement;
ELSIF <expression> THEN
    executable_statement;
ELSE
    executable_statement;
END IF;

When using multiple expressions, use clear indentation and place THEN on its own line. For example:

IF <expression> AND
    <expression>
THEN
    executable_statement;
ELSE
    executable_statement;
END IF;

**LOOPS**

**SIMPLE LOOP**

LOOP
    executable_statement;
END LOOP;

**WHILE LOOP**

WHILE <condition>
LOOP
    executable_statement;
END LOOP;

**FOR LOOPS**

FOR for_index IN low_value .. high_value
LOOP
    executable_statement;
END LOOP;

FOR record_index IN my_cursor
LOOP
    executable_statement;
END LOOP;

**EXCEPTION HANDLERS**

The format of the EXCEPTION section is like the IF statement. For example:

BEGIN
    <expression>
EXCEPTION
    WHEN NO_DATA_FOUND THEN
    executable_statement;
WHEN DUP_VAL_ON_INDEX THEN
   executable_statement;
WHEN OTHERS THEN
   executable_statement;
END;

PACKAGES

Packages should always be named in lower case and end with the suffix _pkg. The names should be meaningful and each word is separated by an underscore (_).

The order of elements is as follows:
1. Variables
2. Complex datatypes (e.g., records and tables)
3. Database-related declarations (e.g., cursors)
4. Named exceptions
5. Modules (procedures and functions)

For example:

PACKAGE package_name_pkg IS
   /****************************************************************************
   Project Name:
   Purpose:
   Development History
   DATE       DEVELOPER INITIALS  MODIFICATION DESCRIPTION
   15-JUL-04   LBR             Initial package creation
   ****************************************************************************/
   v_var1 VARCHAR2(60);
   v_var2 VARCHAR2(60);
   PROCEDURE procedure_name_p (v_var1 IN VARCHAR2,
                     v_var2 OUT VARCHAR2);
END package_name_pkg;

Always include a label to indicate that the package is ending.

PROCEDURES AND FUNCTIONS

Procedures should always be named in lower case and end with the suffix _p. Functions should always be named in lower case and end with the suffix _f. The names should be meaningful and each word is separated by an underscore (_).

When calling a procedure (or function), always indicate with a comment in the line before where the procedure is coming from. For example:

-- This procedure is stored in the database under the SATURN schema
item_p.strip_dash(part_number);
If a SELECT statement can possibly return more than one row and is not in a cursor, an exception handler should be in place. For example:

```
BEGIN
    SELECT serial_no
    INTO s_num
    FROM serial_asset
    WHERE part_ref = 1234;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        s_num := 'MAS0001';
    WHEN TOO_MANY_ROWS THEN
        NULL;
END;
```

**COMMENTS**

The main priority is **COMMENT AS YOU CODE**!

Comments should be placed in any place the code may be difficult to understand. Always answer the question of those who may follow you of: “Why did you do this?”

**DATABASE INDEXES**

Indexes should use the following naming conventions (x represents a sequential number starting with 1):

- `tablename_ux` for unique indexes.
- `tablename_nx` for non-unique indexes.

**3 & 4 GL CODING STANDARDS**

When writing a non-user interface application (C++, COBOL, Java, Pro C, etc.) the standards should follow as closely as possible, those outlined for SQL and PL/SQL. Code efficiency and code simplicity go hand in hand. Don’t sacrifice clarity, readability, or correctness for nonessential improvements. Anyone caught using a “GOTO” will be severely punished.

**FORMS**

Oracle Developer has been selected as the user interface tool for forms and reports development. The following details the standards to be followed when developing user interface screens. The Forms Standards Checklist (found in Appendix A) should be used as an aid to assure that the application conforms to these standards.

**SCREEN FORMATS**

**SCREEN TITLE**

- A mixed case descriptive title should be used in the upper left corner of the screen. The screen title should also match the menu item as closely as possible (Try to keep the title to 40 characters or less).
- The module name (all Upper Case) and revision number should follow the title.
- The database name in parentheses should appear following the module name and revision number.
- The date in the format DD-MON-YYYY should follow the database name.
Here is an example:
General Menu Form GUAGMNU 6.0 (STU6) 11-MAY-2004

WINDOW & CANVAS PROPERTIES

- The size for the window and canvas should be W 692 x H 404.

BLOCK PROPERTIES

- Order By clause should be specified, if possible.
- The detail block should not allow queries if the master block is blank.
- A scroll bar is required for multiple display records.
- Scroll bar width is 13.
- When multiple records are displayed the record the cursor resides on should have a white background while the other records have cream colored background.

ITEM PROPERTIES

- Auto-hint should be on.
- Auto-hint’s text should be in mixed case, brief, and use a standard font and font size; no periods. They must be consistent throughout the application.
- Field text should be black with white background. Field font is 9 pt medium. Field height is 15.
- Currency formats should be $9,999.99 or larger.
- Numbers should be right justified, with a specified format if applicable.
- Date formats should be DD-MON-YYYY with a width of 72.
- Text items should be left justified.
- Case Restriction should be Upper, when applicable.
- Field labels should be black; unless the field is required then maroon is to be used.
- Field labels will use a standard font in 11 pt bold; no abbreviations or underscores. They must be consistent throughout the application (Unit name will always be “Unit name”).
- Field labels will either be on the left side of the field and right adjusted OR above the field and centered.
- Block headings will use a standard font in 12 pt bold positioned at the upper left hand corner of the inset box.

RECORD GROUPS

- Select statements should have an Order By clause.
- Selected records should be distinct.

LOV’S

- Each column should have a column title.
- All columns and column titles are visible on the LOV.
- When possible, the LOV should not cover the item tied to the LOV.
- The LOV label will match the field label from which it is called.

BUTTON NAMING STANDARDS

<table>
<thead>
<tr>
<th>Function</th>
<th>Button Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMIT</td>
<td>Save</td>
</tr>
<tr>
<td>PREVIOUS RECORD</td>
<td>&lt;</td>
</tr>
</tbody>
</table>
### Push Buttons

- Every screen will have an Exit button, as a minimum.
- Push buttons should be created for frequently used functions, e.g., COMMIT.
- Use push buttons to invoke one, and only one, function.
- Buttons that perform an action but require some other piece of information from the user should end with a series of three periods (e.g., Save As…).
- Buttons should not be either navigable or mouse navigable.
- Group large numbers of buttons together based on the functionality.
- Buttons which are grouped together by function should be horizontally stacked.
- Buttons will take the expected action or display a message in the status line (e.g., “No LOV for this field”).
- The List button should be used for dynamic Lists of Values.
- Standard buttons will be grouped at the right of the screen, preferably at the bottom. If the page layout does not permit this, the buttons may be placed at the top right of the screen.
- Special buttons may be placed where most applicable (e.g., Category Totals).
- The standard button order from right to left is: Exit, Save, Query, LOV (Other buttons may be included as needed: Details, New, Delete, Clear, Print, Previous, Next, Execute, Search, Undo, etc.)
- Buttons should be spaced a minimum 3 pixels apart.
- Buttons should use standard graphics when possible, if text is used the font should always be black, 11 pt bold.

### Check Boxes

- Use Check Boxes when only one value is applicable (e.g., Yes/No, True/False…).

### Pop Lists

- Use Pop Lists for a static list of values more than 2, but less than 13.

### Radio Groups

- Use Radio Groups for a mutually exclusive, static list of values less than 8.
- Always have a default value.

### Menu Bar
• Menu bars should only contain the functions applicable to the form. When using a standard menu, gray out the functions which do not apply.

GENERAL

• Control items, such as list boxes and radio buttons, should be grouped together.
• Select valid values from a list box, rather than just typing it in.
• Provide a clear method to exit every window (Exit/Close button).
• Avoid forcing the user to scroll.
• Use as much white space as possible.
• Use mixed case text. Capitalize the first letter of each word.
• Draw boxes to logically separate items on the screen.
• Place scroll bars inside the logical boxes, right aligned with the item(s).
• Navigation should be left-to-right, top-to-bottom whenever possible.
• Pop-up and List of Value windows should be positioned to avoid covering the header and error/message areas.
• Titles for boxes are optional. They should only be used for clarification.
• Duplicate record check should be available for each enterable record.
• Delete Alert Box should be used prior to deleting a record.
• “Are You Sure?” Alert should be used prior to executing a process/report.
• Confirm Delete and Save should commit to the database immediately.

VARIABLE NAMING STANDARDS

The following naming conventions will be used. Descriptive names will be mixed case with the first letter of each word capitalized. No spaces or underscores in the descriptive name.

<table>
<thead>
<tr>
<th>Item Name</th>
<th>Item Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>xxx_RG</td>
<td>Radio Group</td>
</tr>
<tr>
<td>xxx_RB</td>
<td>Radio Button</td>
</tr>
<tr>
<td>xxx_CB</td>
<td>Check Box</td>
</tr>
<tr>
<td>xxx_PB</td>
<td>Push Button</td>
</tr>
<tr>
<td>xxx_LOV</td>
<td>List of Values</td>
</tr>
</tbody>
</table>

* xxx = descriptive name of item

VALIDATION

Validation for an item must be performed at the form level. The user should be notified of a validation failure immediately following entry into the item. Do not use database triggers or constraints as the method to validate user input.

SCROLL BARS

• Never use a horizontal scroll bar. Screens should be designed to avoid the need for horizontal scrolling.
• For vertical scroll bars, the top of the up arrow button should be aligned with the top of the first record. The bottom of the down arrow button should be aligned with the bottom of the last record.
APPENDIX A – FORMS CHECKLIST

All forms are located in m:\application development group\master forms\.

### Code Modification Checklist

<table>
<thead>
<tr>
<th>AI/AE #(s)</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module Name:</td>
<td>Rev#</td>
</tr>
<tr>
<td>Team Member:</td>
<td></td>
</tr>
</tbody>
</table>

**Request Type:**
- [ ] Production
- [ ] Maintenance
- [ ] User Support

- [ ] Create Change Request Form.
  (Include Problem Description, Resolution, Install/Back-Out Text, Acceptance Criteria/Unit Test Plan, and Risk.)

**Maintenance AI’s ONLY**

- [ ] Development definitive estimate
- [ ] Include budget estimates (+25% → -10%).
  Add 20% test time to estimate
- [ ] Change Request ownership to Product Manager for **APPROVAL**. Also set Approver Requested Action to **Approval** and set the Date Approval Requested.
- [ ] Approved
- [ ] If definitive estimate is over 60 hours, complete the Estimating Form.
- [ ] If estimates are over 80 hours, obtain Project Manager approval.
- [ ] Re-approval accepted

- [ ] Check module out of Version Tool.
- [ ] Include header comment and AI/AE comments where code is modified in the module.
- [ ] Submit Database Change Request.

**Peer Review Only**

- [ ] Send out agenda
- [ ] Peer Review

- [ ] Check module(s) back into VM, add Change Request #(s).
- [ ] Create/append Unit Test Plan & perform Unit Test
- [ ] Move the module(s) to the appropriate folder in the DEVELOPMENT environment.
Copy module(s) to the Web folder in the DEVELOPMENT environment.

If Test is NOT Required

Move the module(s) to the appropriate PRODUCTION STAGE folder. Include scripts for database changes.

Update Documentation

Test/QA Required Change Requests Only

Promote module(s) to TEST_STAGE.

Transfer Change Request Ownership to the Tester and change request status to Test

Unit Test

☐ Test Passed  ☐ Test Failed

User Acceptance Test

☐ Test Passed  ☐ Test Failed

Promote module(s) to PRODUCTION STAGE if they passed testing. If Test FAILS, demote module(s) to DEVELOPMENT STAGE. Start new check list.

Stage module for release in the appropriate PRODUCTION STAGE folder. Include scripts for database changes.

Change Change Request ownership to Product Manager for ACCEPTANCE. Set Approver Requested Action to Acceptance and set the Date Acceptance Requested. Set request status to Sent for Acceptance.

If ACCEPTED, change request status to Release Management.


Notify release manager of modules to be released.

Date module released to production. ____________

Once released promote module(s) to PRODUCTION

Close the Change Request

Update the Resolution Type, Resolution Date, and State fields and set the Request Status field to Closed.

Place Checklist in folder and file the folder.
GLOSSARY

AI/AE  Application Issue/Application Enhancement