|  |  |
| --- | --- |
| DEP Logo | **Department of Environmental Protection** |
| ***STD-14121501.2.0*** | ***Page 1 of 16*** |
| **Database Object Coding Standard** | |

# Purpose

This document specifies the Florida Department of Environmental Protection’s (DEP) Database Object Coding Standard. The purpose of this standard is to ensure that DEP database schema objects not covered by the Physical Data Model Standard (STD-09061804) have a consistent look and feel.

# Scope

This standard applies to all database schema development at DEP.

# Standard

1. All DEP database schemas shall follow Oracle database standards and guidelines found at the ***Oracle Technology Network*** website at <http://www.oracle.com/technetwork/index.html>
2. Developers shall follow the Database Object Coding Specifications which is included as Appendix A to this standard. This specification provides technical guidelines, definitions, and references, including database naming standards, SQL coding standards, and instructions for naming procedures, packages, object types, materialized views, scheduled jobs, and functions.
3. The DEP Logical, Physical and Database Object Coding standards take precedence; if not documented there, the default Oracle data dictionary object standards are considered the DEP Oracle database standard.

# Deviation from Use

Any deviation from this standard shall be documented in associated project and contract documentation. For contracts, deviation from standard shall be documented and approved by the DEP contract manager after consulting with the OTIS technical coordinator. For non-contract work, deviation from use shall be documented in the project plan/scope of work and approved by the project manager.

# Appendices Appendix A: *Database Object Coding Specifications.*

Appendix B: *Oracle Reserved Words*

# Approvals

**Approved By:** Warren Sponholtz, Chief Information Officer   
**Date Approved:** September 2, 2015

**Appendix A: Database Object Coding Specifications**

# Introduction

The development team uses the business model to define the physical implementation of the system. The physical implementation is transformed into a Physical Data Model (PDM) that represents the physical tables and other database objects (views, triggers, procedures, etc.), which will be accessed by the application. During development additional database objects (functions, procedures, packages, database links, materialized views, etc.) are created to meet the business needs of the system. The Database Administration (DBA) staff in the Office of Technology and Information Systems (OTIS) may act as consultants during the coding of these additional objects.

# FUNCTIONS

The function definition allows the developer to implement business rules requiring specialized logic that should not be hard-coded in the application and that only return one value.

# Function Name

The following rules apply to function names:

* A function name is mandatory.
* It cannot exceed 30 characters.
* It should be made up of real words.
* It must end in \_FN.
* Valid examples are CREATE\_BUSINESS\_ID\_FN, TRANSFORM\_GIS\_PROJECTION\_FN.
* The function name includes underscores in place of spaces.
* Function names are not prefixed with an application abbreviation.
* When the function name is only one word, it must not be an [Oracle Reserved Word](#_Oracle_Reserved_Words_1).

# PROCEDURES

The procedure definition allows the developer to implement business rules requiring specialized logic that should not be hard-coded in the application and that may return one or more values.

# Procedure Name

The following rules apply to procedure names:

* A procedure name is mandatory.
* It cannot exceed 30 characters.
* It should be made up of real words.
* It must end in \_PRC.
* It must contain a section at the top with comments on when the procedure was written, who wrote it, and a description of the function that the procedure performs.
* If the procedure is one of several used for a purpose then all procedure names should be declared in the top comment section along with the order in which the current procedure is run in relation to the other procedures.
* Valid examples are LOAD\_RESULT\_TABLE\_PRC, GET\_RESULT\_STATUS\_PRC.
* The procedure name includes underscores in place of spaces.
* Procedure names are not prefixed with an application abbreviation.
* When the procedure name is only one word, it must not be an [Oracle Reserved Word](#_Oracle_Reserved_Words_1).

# PACKAGES

The package definition allows the developer to implement business rules requiring specialized logic that should not be hard-coded in the application. It contains one or more functions and procedures that should be logically grouped and that may protected or available for use to the application.

# Package Name

The following rules apply to package names:

* A package name is mandatory.
* It cannot exceed 30 characters.
* It should be made up of real words.
* It must end in \_PKG.
* It must contain a section at the top with comments on when the package was written, who wrote it, and a description of the function that the script package.
* If the script is one of several used for a purpose then all related package names should be declared in the top comment section along with information on how they are related.
* Valid examples are PROCESS\_INVOICES\_PKG, PERFORM\_BILLING\_PROCESS\_PKG.
* The package name includes underscores in place of spaces.
* Package names are not prefixed with an application abbreviation.
* The package may contain only a specification or a specification and body.
* When the package name is only one word, it must not be an [Oracle Reserved Word](#_Oracle_Reserved_Words_1).

# Package Specification and Body Name

The Package Specification and Package Body names must follow the standards found in the Function and Procedure sections of this standard.

# OBJECT AND COLLECTION TYPES

Oracle Object and Collection Types should be used sparingly. The developer is responsible for consulting with the DBA section and obtaining permission to use these special types before any coding begins. All code must be reviewed and approved by the OTIS DBA section.

# Object Type

The following rules apply to Object Type names:

* An object type name is mandatory.
* It cannot exceed 30 characters.
* It should be made up of real words.
* It must end in \_OT.
* Valid examples are EMPLOYEE\_OT, DEPARTMENT\_OT.
* The object type name includes underscores in place of spaces.
* Object Type names are not prefixed with an application abbreviation.
* When the object type name is only one word, it must not be an [Oracle Reserved Word](#_Oracle_Reserved_Words_1).

# Collection Type

The following rules apply to Collection Type names:

* A collection type name is mandatory.
* It cannot exceed 30 characters.
* It should be made up of real words.
* It must end in \_CT.
* Valid examples are EMPLOYEE\_ADDRESS\_CT, DEPARTMENT\_ADDRESS\_CT.
* The collection type name includes underscores in place of spaces.
* Collection Type names are not prefixed with an application abbreviation.
* When the collection type name is only one word, it must not be an [Oracle Reserved Word](#_Oracle_Reserved_Words_1).

# DATABASE LINKS

Database Links should be used sparingly. The developer is responsible for consulting with the DBA section and obtaining permission to use database links before any coding begins. All code must be reviewed and approved by the OTIS DBA section.

The following rules apply to Database Links:

* Database Link names are mandatory.
* Each database link name must match the name of the instance to which it is connecting.
* Database Links may be used to access the GIS (Geographic Information Systems) database instances or to connect to another instance only when there is no other source for the required data.
* If the Database Link is accessing a schema maintained by another developer then permission must be obtained from that developer before establishing the link.
* Database Links may only be used for read-only operations.

# MATERIALIZED VIEWS

Materialized Views are created and maintained by the OTIS DBA section when simple views are required. If complex views are required, the developer is responsible for creating the SQL to produce the complex views, consulting with the DBA section on syntax, and obtaining permission to create these views before coding begins. All code must be reviewed and approved by the OTIS DBA section.

The following rules apply when creating Materialized Views:

* Materialized View names are mandatory.
* Materialized View names should match the base table name unless a business reason prevents the developer from meeting this rule.
* The primary key of the Materialized View must be the primary key of the base table.
* Materialized Views should use the same indexes as found on the base table and additional indexes if required for performance.
* Materialized Views of code tables should be set as Complete Refresh unless the data is expected to be updated or added often.
* Materialized Views of data tables should be set as Fast Refresh.
* Materialized Views must be added to a refresh group as directed by the OTIS DBA section.
* Materialized Views may only be built off of primary tables or another materialized view that is built off a primary table. These objects cannot consist of more than two layers of dependent objects.

# JAVA OBJECTS

Java Objects should be used sparingly. If Java Objects are required, the developer is responsible for consulting with the DBA section and obtaining permission to create these objects before coding begins. All code must be reviewed and approved by the OTIS DBA section.

The following rules apply to Java Objects:

* Java Object names are mandatory.
* Java Object names must meet standard rules for naming Java classes.
* Java Objects may be used in conjunction with database procedures.
* Java Object source code must be provided along with the compiled class when submitted to the OTIS DBA section.
* Java Objects must contain standard Javadoc comments and meet standard Java coding rules.

# SCHEDULED JOBS

Scheduled Jobs are maintained by the OTIS DBA section; the primary mechanism for scheduling database jobs is by UNIX crontab. The developer is responsible for creating the SQL to be run by the crontab, creating the UNIX shell scripts for executing the SQL, and consulting with the DBA section to ensure the UNIX code will function in the DEP environment. All code must be reviewed and approved by the OTIS DBA section.

The following rules apply to Scheduled Jobs:

* Scheduled Job names are mandatory.
* Scheduled Job names must begin with the schema name for which the job is being run.
* It cannot exceed 30 characters.
* It should be made up of real words.
* It must contain a section at the top with comments on when the script was written, who wrote it, and a description of the function that the script performs.
* If the script is one of several used for a purpose then all script names should be declared in the top comment section along with the order in which the current script is run in relation to the other scripts.
* Valid examples are PA\_LOAD\_PERMIT\_DATA, CRA\_CALCULATE\_FEES.
* The Scheduled Job includes underscores in place of spaces.
* When the Scheduled Job name is only one word, it must not be an [Oracle Reserved Word](#_Oracle_Reserved_Words_1).
* The executed SQL code must be stored in a database procedure or database package.
* The Scheduled Job must execute under the schema account which owns the database procedure or package.
* The Scheduled Job may contain UNIX commands to create files, ftp files, and/or remove files only from the directory location where the scripts are stored. Exceptions may be made if a business case is presented and approved by the OTIS DBA section.
* Developers are responsible for testing the UNIX shell scripts from their own UNIX account on the DEP development database server before submitting them to be added to the crontab.

# SQL CODING STANDARDS and GUIDELINES

Database object SQL and PL SQL code is maintained by the developer and is expected to adhere to general best-practice coding standards. There are multiple references online for best-practices, those consulted for the standards/guidelines found below are:

Burleson Consulting:

<http://www.dba-oracle.com>

Oracle: <https://docs.oracle.com/cd/B28359_01/server.111/b28286/intro002.htm#SQLRF50928>

Database access software such as Dell TOAD, Dell SQLNavigator, and Oracle SQLDeveloper contain built-in code hints and formatting options that meet international standards, DEP standards, and DEP guidelines. Developers are expected to utilize these utilities as needed to adhere to best-practices.

# STANDARDS

The following standards apply to SQL coding:

* PL SQL scripts, packages, and procedures must contain a comment header as defined below.
* PL SQL scripts, packages, and procedures must contain comments throughout the code to document business rules and special information used to derive the logic related to the SQL.
* Double-quote marks should not be used in any scripts that create database objects such as CREATE FUNCTION, CREATE PACKAGE, CREATE TABLE, CREATE VIEW, etc.
* Default database parameters such as EDITIONABLE, BEQUETH DEFINER, and FORCE should not be used in any scripts that create database objects such as CREATE FUNCTION, CREATE PACKAGE, CREATE TABLE, CREATE VIEW, etc.
* Variable names:
  + Must not exceed 30 characters.
  + Should be made up of real words.
  + Should clearly identify the data element being stored.
* For long PL SQL, a variable should be defined and set throughout the script with a value to track the execution for debugging purposes. For example, the variable ERROR\_STAGE VARCHAR2(50) may be declared and used before each critical section of code as ERROR\_STAGE := ‘1.0’; or ERROR\_STAGE := ‘Entering Loop for inserting Employee Data’;
* PL SQL scripts, packages, and procedures must contain EXCEPTION handling.
* EXCEPTIONS may be handled by storing records in a table (SCHEMANAME.SCHEMA\_ERRORS) or by returning the error to the application.
* PL SQL scripts, packages, and procedures must be formatted for readability and clarity.
* PL SQL within one schema may execute PL SQL within another schema or database instance; coordination with developers responsible for the “external” schemas is required.

# GUIDELINES

The following PL SQL Formatting, Readability, and Best-Practice guidelines also apply to SQL coding:

# PL/SQL Headers

A header should appear at the start of any script, procedure, function, package body, or package spec. The template header should contain information on when the code was written, who wrote it, and the function that it performs. An example header is:

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Description: Describe the purpose of the object. If necessary,

-- describe the design of the object at a very high level.

--

-- Input Parameters:

--

-- Output Parameters:

--

-- Error Conditions Raised:

--

-- Author: <your name>

--

-- Revision History

-- Date Author Reason for Change

-- ----------------------------------------------------------------

-- 03 DEC 2014 J.Schmoe Created.

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Alignment of PL/SQL Operators

These guidelines enhance the readability of code by adding white space and clarifying complex expressions. Arrange series of statements containing similar operators into columns whenever it will not cause excessive white space and you have sufficient room to do so.

Correct:

vFirstName := 'Roger';

  vLastName  := 'Smith';

  vSSN := 999999999;

Incorrect:

vFirstName := 'Roger';

  vLastName := 'Smith';

  vSSN := 999999999;

# PL/SQL Indentation Best Practices Standards

Code should always be indented consistently, using three spaces for each level of indentation. Variable, type, and constant declarations should all be indented to the first level of indentation; do not use tab characters.

IF (some expression) THEN

IF (some expression) THEN

IF (some expression) THEN

<statements>

ELSIF (some expression) THEN

<statements>

END IF;

END IF;

  END IF;

<statements>

Statements following the WHEN clause of an exception handler should be indented five spaces, in order to create a column-like effect within the exception handler.

Correct:

EXCEPTION

WHEN OTHERS THEN

DBMS\_Output.Put\_Line (SQLERRM);

Incorrect:

EXCEPTION

WHEN OTHERS THEN

DBMS\_Output.Put\_Line (SQLERRM);

# PL/SQL Horizontal Spacing Best Practices Standards

One of the most important elements in creating readable code is the spacing placed around operators. Below shows common operators and keywords that need to be preceded and followed by a space when they are used in expressions.

|  |  |  |  |
| --- | --- | --- | --- |
| Operators and keywords to be preceded and followed by a space in expressions. | | | |
| + | - | \* | / |
| & | < | > | = |
| != | <= | >= | := |
| => | || | .. | : |
| <> | IN | OUT | AND |
| OR | NOT | NULL |  |

Often more than one of the operators and keywords shown in the table above will be adjacent to each other inside an expression. In this instance, it is recommended that only one space lie between the two operators/identifiers. For example:

IF (vMajor IS NOT NULL) THEN

Spaces should precede and follow character (') literals.

SELECT first\_name || ' ' || middle\_name || ' ' || last\_name

'student\_name'

  FROM   STUDENTS

  WHERE  ssn = 999999999;

Do not leave any blank spaces preceding or following the \*\* operator.

nRaisedNum := nNum\*\*nPower;

Do not leave blank spaces before or after the plus (+) and minus (-) signs when used as unary operators.

nNumber := -nSecondNumber;

 nNumber := +nSecondNumber;

Do not use spaces between multiple parentheses or semicolons (;). Always precede the first opening parenthesis of a set with a space.

AND (((x < 5) AND (y < 5))

  OR   ((x > 5) AND (y > 5)));

# PL/SQL Vertical Spacing Best Practices Standards

Vertical spacing helps distance elements in the code from one another, reducing the visual clutter above and below statements. To create appropriate vertical spacing for your code, place a blank line in the locations described in the following list:

Before lines containing the keywords IF, ELSE, ELSIF, and EXCEPTION. If the line is preceded by a comment, place the blank line before the comment instead of before the line of text.

--

  -- If the student's grade point average meets the criteria for

  -- mandatory academic counseling, add the student's name and social

  -- security number to the list.

  --

  IF (nRealGPA < 1.5) THEN

<statements>

--

  -- We also want to consider students who are failing two or more

  -- classes, even if their GPA is above 1.5.

  --

  ELSIF Has\_Two\_Fails (nForSSN => nSSN) THEN

<statements>

ELSE

<statements>

  END IF;

Before any line containing the LOOP keyword. Do not place a blank line before source code containing the END LOOP keyword. (As with lines of code containing the IF keyword, keep the comments for a line of code with the comment by placing a blank line before the comment.)

--

  -- For each student returned by the query, add the student's social

  -- security number to the PL/SQL table.

  --

  FOR Students\_rec IN Students\_cur LOOP

<statements>

  END LOOP;

Before each exception after the first declared within the EXCEPTION section of a PL/SQL block.

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

<statements>

WHEN TOO\_MANY\_ROWS THEN

<statements>

WHEN OTHERS THEN

<statements>

Before and after the variable, constant, and type declarations for a PL/SQL block.

PROCEDURE Update\_Student\_GPA (nSSN IN number)

  IS

<declaration>

<declaration>

BEGIN

<statements>;

  END Update\_Student\_GPA;

Following the declaration of the procedure and its parameters.

PROCEDURE Update\_Student\_GPA (nSSN IN number)

IS

Do not place an empty line before a line containing the END IF keyword. Do place blank lines after the last line of code containing the END IF keyword.

IF (some expression) THEN

IF (some expression) THEN

IF (some expression) THEN

<statements>

END IF;

END IF;

  END IF;

<statements>

**Appendix B: Oracle Reserved Words**

# Oracle Reserved Words – Listed Alphabetically

| **A-G** | **F-N** | **N-S** | **S-Z** |
| --- | --- | --- | --- |
| ACCESS | FILE | NOWAIT | SYNONYM |
| ADD | FLOAT | NULL | SYSDATE |
| ALL | FOR | NUMBER | TABLE |
| ALTER | FROM | OF | THEN |
| AND | GRANT | OFFLINE | TO |
| ANY | GROUP | ON | TRIGGER |
| AS | HAVING | ONLINE | UID |
| ASC | IDENTIFIED | OPTION | UNION |
| AUDIT | IMMEDIATE | OR | UNIQUE |
| BETWEEN | IN | ORDER | UPDATE |
| BY | INCREMENT | PCTFREE | USER |
| CHAR | INDEX | PRIOR | VALIDATE |
| CHECK | INITIAL | PRIVILEGES | VALUES |
| CLUSTER | INSERT | PUBLIC | VARCHAR |
| COLUMN | INTEGER | RAW | VARCHAR2 |
| COMMENT | INTERSECT | RENAME | VIEW |
| COMPRESS | INTO | RESOURCE | WHENEVER |
| CONNECT | IS | REVOKE | WHERE |
| CREATE | LEVEL | ROW | WITH |
| CURRENT | LIKE | ROWID |  |
| DATE | LOCK | ROWNUM |  |
| DECIMAL | LONG | ROWS |  |
| DEFAULT | MAXEXTENTS | SELECT |  |
| DELETE | MINUS | SESSION |  |
| DESC | MLSLABEL | SET |  |
| DISTINCT | MODE | SHARE |  |
| DROP | MODIFY | SIZE |  |
| ELSE | NOAUDIT | SMALLINT |  |
| EXCLUSIVE | NOCOMPRESS | START |  |
| EXISTS | NOT | SUCCESSFUL |  |