

Using Oracle-Supplied Packages

What Will I Learn?

In this lesson, you will learn to:

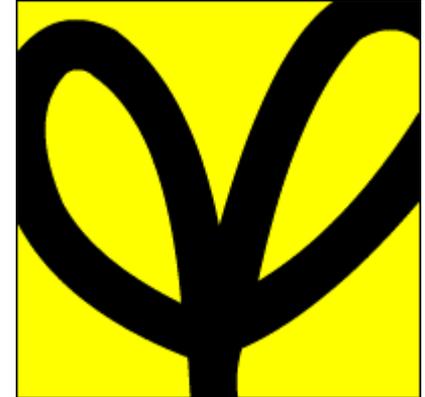
- Describe two common uses for the `DBMS_OUTPUT` server-supplied package
- Recognize the correct syntax to specify messages for the `DBMS_OUTPUT` package
- Describe the purpose for the `UTL_FILE` server-supplied package.
- Recall the exceptions used in conjunction with the `UTL_FILE` server-supplied package.





Why Learn It?

You already know that Oracle supplies a number of SQL functions (UPPER, TO_CHAR, and so on) that you can use in your SQL statements when required. It would be wasteful for everyone to have to “re-invent the wheel” by writing their own functions to do these things.

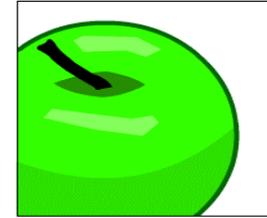


In the same way, Oracle supplies a number of ready-made PL/SQL packages to do things that most application developers and/or database administrators need to do from time to time.

In this lesson, you learn how to use two of the Oracle-supplied PL/SQL packages. These packages focus on generating text output and manipulating text files.

Tell Me / Show Me

Using Oracle-Supplied Packages



You can use these packages directly by invoking them from your own application, exactly as you would invoke packages that you had written yourself.

Or, you can use these packages as ideas when you create your own subprograms.

Think of these packages as ready-made “building blocks” that you can invoke from your own applications.

 **Tell Me / Show Me****List of Some Oracle-Supplied Packages**

DBMS_LOB	Enables manipulation of Oracle Large Object column datatypes: CLOB, BLOB and BFILE
DBMS_LOCK	Used to request, convert, and release locks in the database through Oracle Lock Management services
DBMS_OUTPUT	Provides debugging and buffering of messages
HTP	Writes HTML-tagged data into database buffers
UTL_FILE	Enables reading and writing of operating system text files
UTL_MAIL	Enables composing and sending of e-mail messages
DBMS_SCHEDULER	Enables scheduling of PL/SQL blocks, stored procedures, and external procedures or executables

Tell Me / Show Me

The DBMS_OUTPUT Package

The DBMS_OUTPUT package sends text messages from any PL/SQL block into a private memory area, from which the message can be displayed on the screen.

Common uses of DBMS_OUTPUT include:

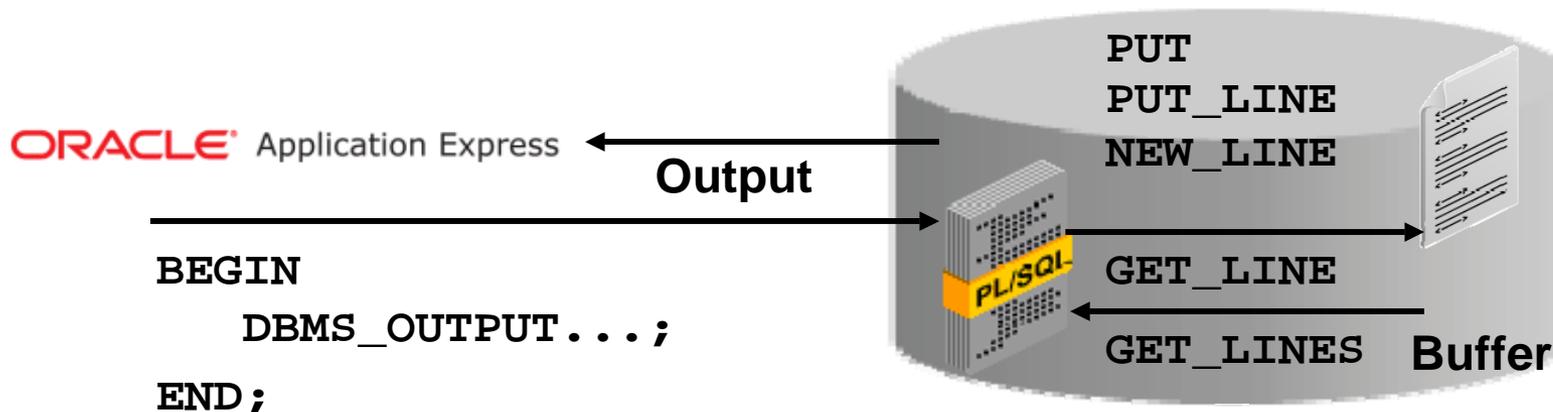
- You can output results back to the developer during testing for debugging purposes.
- You can trace the code execution path for a function or procedure.

Tell Me / Show Me

How the DBMS_OUTPUT Package Works

The DBMS_OUTPUT package enables you to send messages from stored subprograms and anonymous blocks.

- PUT places text in the buffer.
- NEW_LINE sends the buffer to the screen.
- PUT_LINE does a PUT followed by a NEW_LINE.
- GET_LINE and GET_LINES read the buffer.
- Messages are not sent until after the calling block finishes.





Tell Me / Show Me

Using DBMS_OUTPUT: Example 1

You have already used `DBMS_OUTPUT.PUT_LINE`. This writes a text message into a buffer, then displays the buffer on the screen:

```
BEGIN
  DBMS_OUTPUT.PUT_LINE('The cat sat on the mat');
END;
```

If you wanted to build a message a little at a time, you could code:

```
BEGIN
  DBMS_OUTPUT.PUT('The cat sat ');
  DBMS_OUTPUT.PUT('on the mat');
  DBMS_OUTPUT.NEW_LINE;
END;
```



Tell Me / Show Me

Using DBMS_OUTPUT: Example 2

You can trace the flow of execution of a block with complex IF ... ELSE, CASE or looping structures:

```
DECLARE
  v_bool1    BOOLEAN := true;
  v_bool2    BOOLEAN := false;
  v_number   NUMBER;
BEGIN
  ...
  IF v_bool1 AND NOT v_bool2 AND v_number < 25 THEN
    DBMS_OUTPUT.PUT_LINE('IF branch was executed');
  ELSE
    DBMS_OUTPUT.PUT_LINE('ELSE branch was executed');
  END IF;
  ...
END;
```



Tell Me / Show Me

DBMS_OUTPUT Is Designed for Debugging Only

You would not use `DBMS_OUTPUT` in PL/SQL programs that are called from a “real” application, which can include its own application code to display results on the user’s screen. Instead, you would return the text to be displayed as an `OUT` argument from the subprogram. For example:

```
PROCEDURE do_some_work (...) IS BEGIN
    ... DBMS_OUTPUT.PUT_LINE('string'); ... END;
```

Would be converted to:

```
PROCEDURE do_some_work (... p_output OUT VARCHAR2) IS
BEGIN
    ... p_output := 'string'; ...
END;
```



Tell Me / Show Me

DBMS_OUTPUT Is Designed for Debugging Only (continued)

For this reason, you should not use DBMS_OUTPUT in subprograms, but only in anonymous PL/SQL blocks for testing purposes. Instead of:

```
CREATE OR REPLACE PROCEDURE do_some_work IS BEGIN
    ... DBMS_OUTPUT.PUT_LINE('string'); ... END;

BEGIN    do_some_work;    END;    -- Test the procedure
```

You should use:

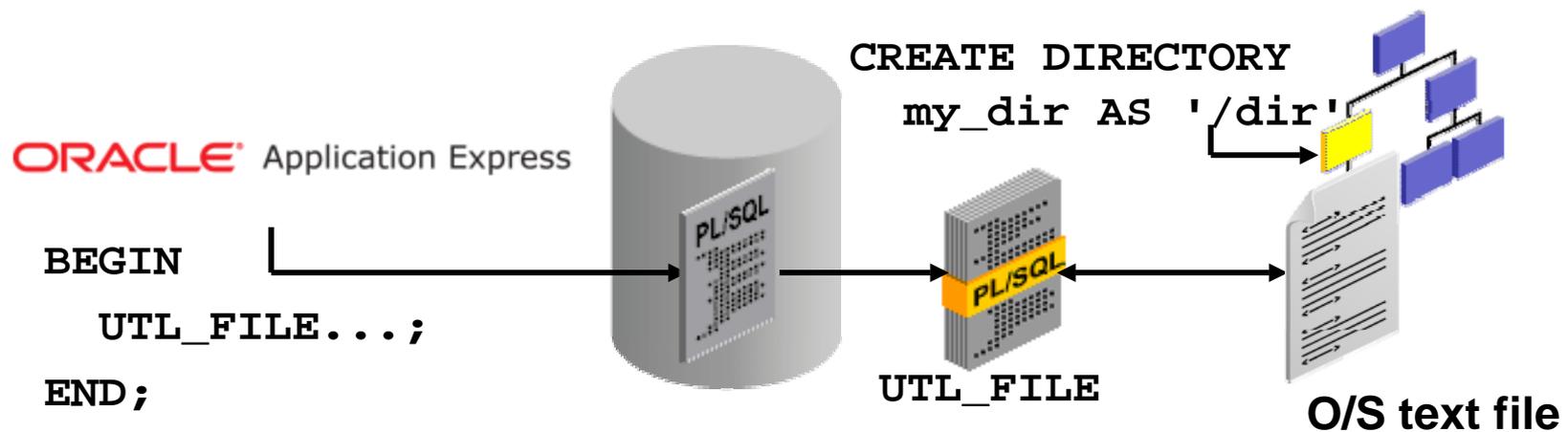
```
CREATE OR REPLACE PROCEDURE do_some_work
    (p_output OUT VARCHAR2) IS BEGIN
    ... p_output := 'string'; ... END;

DECLARE v_output VARCHAR2(100); BEGIN --Test the procedure
    do_some_work(v_output);
    DBMS_OUTPUT.PUT_LINE(v_output);    END;
```

Tell Me / Show Me

The UTL_FILE Package:

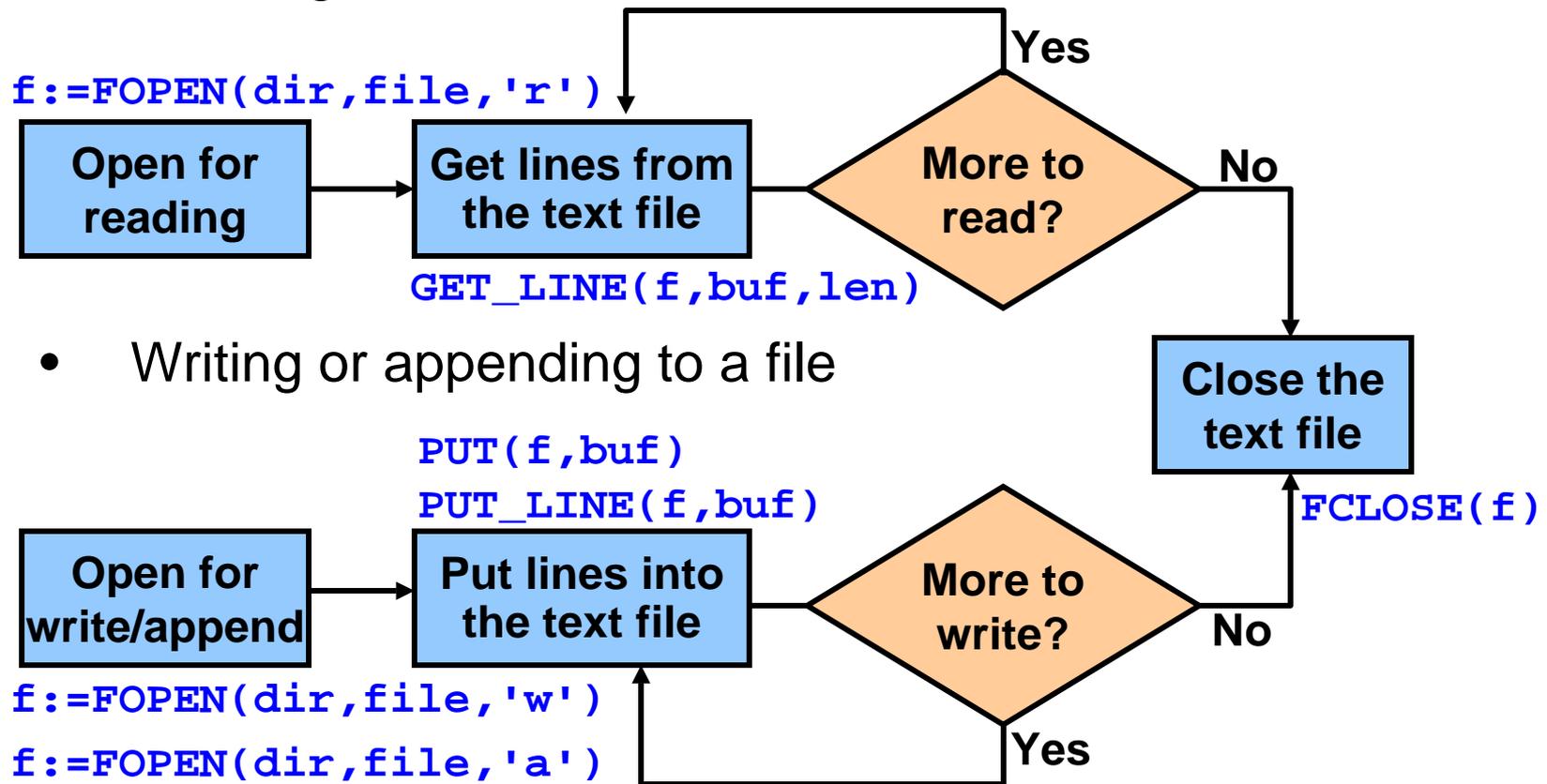
- Allows PL/SQL programs to read and write operating system text files.
- Can access text files in operating system directories defined by a `CREATE DIRECTORY` statement. You can also use the `utl_file_dir` database parameter.



Tell Me / Show Me

File Processing Using the UTL_FILE Package

- Reading a file





Tell Me / Show Me

File Processing Using the `UTL_FILE` Package

You open files for reading or writing with the `FOPEN` function. You then either read from or write or append to the file until processing is done. Then close the file by using the `FCLOSE` procedure. The following are the main subprograms:

- The `FOPEN` function opens a file in a specified directory for input/output (I/O) and returns a file handle used in later I/O operations.
- The `IS_OPEN` function checks whether the file is already open, returning a Boolean.
- The `GET_LINE` procedure reads a line of text from the file into an output buffer parameter. The maximum input record size is 1,023 bytes.
- The `PUT` and `PUT_LINE` procedures write text to the opened file.
- The `NEW_LINE` procedure terminates a line in an output file.
- The `FCLOSE` procedure closes an opened file.

Tell Me / Show Me

Exceptions in the UTL_FILE Package

You might have to handle one or more of these exceptions when using UTL_FILE subprograms:

- INVALID_PATH
- INVALID_MODE
- INVALID_FILEHANDLE
- INVALID_OPERATION
- READ_ERROR
- WRITE_ERROR
- INTERNAL_ERROR

The other exceptions not specific to the UTL_FILE package are:

- NO_DATA_FOUND and VALUE_ERROR



Tell Me / Show Me

FOPEN and IS_OPEN Function Parameters

```
FUNCTION FOPEN (location IN VARCHAR2,  
               filename IN VARCHAR2,  
               open_mode IN VARCHAR2)  
RETURN UTL_FILE.FILE_TYPE;
```

```
FUNCTION IS_OPEN (file IN FILE_TYPE)  
RETURN BOOLEAN;
```

Example:

```
PROCEDURE read(dir VARCHAR2, filename VARCHAR2) IS  
  file UTL_FILE.FILE_TYPE;  
BEGIN  
  IF NOT UTL_FILE.IS_OPEN(file) THEN  
    file := UTL_FILE.FOPEN (dir, filename, 'r');  
  END IF; ...  
END read;
```



Tell Me / Show Me

Using UTL_FILE: Example (continued on next two slides)

In this example, the `sal_status` procedure uses `UTL_FILE` to create a text report of employees for each department, along with their salaries. In the code, the variable `v_file` is declared as `UTL_FILE.FILE_TYPE`, a `BINARY_INTEGER` datatype that is declared globally by the `UTL_FILE` package.

The `sal_status` procedure accepts two `IN` parameters: `p_dir` for the name of the directory in which to write the text file, and `p_filename` to specify the name of the file.

To invoke the procedure, use (for example):

```
BEGIN sal_status('MY_DIR', 'salreport.txt'); END;
```



Tell Me / Show Me

Using UTL_FILE: Example (continued on next slide)

```
CREATE OR REPLACE PROCEDURE sal_status(  
  p_dir IN VARCHAR2, p_filename IN VARCHAR2) IS  
  v_file UTL_FILE.FILE_TYPE;  
  CURSOR empc IS  
    SELECT last_name, salary, department_id  
           FROM employees ORDER BY department_id;  
  v_newdeptno employees.department_id%TYPE;  
  v_olddeptno employees.department_id%TYPE := 0;  
BEGIN  
  v_file:= UTL_FILE.FOPEN (p_dir, p_filename, 'w');      -- 1  
  UTL_FILE.PUT_LINE(v_file,  
    'REPORT: GENERATED ON ' || SYSDATE);                -- 2  
  UTL_FILE.NEW_LINE (v_file); ...                        -- 3
```

 **Tell Me / Show Me****Using UTL_FILE: Example (continued)**

```
FOR emp_rec IN empc LOOP
    UTL_FILE.PUT_LINE(v_file,                                --4
        'EMPLOYEE: ' || emp_rec.last_name ||
        'earns: ' || emp_rec.salary);
END LOOP;
UTL_FILE.PUT_LINE(v_file, '*** END OF REPORT ***'); --5
UTL_FILE.FCLOSE(v_file); --6
EXCEPTION
WHEN UTL_FILE.INVALID_FILEHANDLE THEN --7
    RAISE APPLICATION_ERROR(-20001, 'Invalid File.');
```

```
WHEN UTL_FILE.WRITE_ERROR THEN --8
    RAISE_APPLICATION_ERROR (-20002, 'Unable to
                                write to file');
END sal_status;
```



Tell Me / Show Me

Using UTL_FILE: Example: Invocation and Output Report

Suppose you invoke your procedure by:

```
BEGIN    sal_status('MYDIR', 'salreport.txt');    END;
```

The output contained in the file is:

```
SALARY REPORT: GENERATED ON 29-NOV-06  
EMPLOYEE: Whalen earns: 4400  
EMPLOYEE: Hartstein earns: 13000  
EMPLOYEE: Fay earns: 6000  
...  
EMPLOYEE: Higgins earns: 12000  
EMPLOYEE: Gietz earns: 8300  
EMPLOYEE: Grant earns: 7000  
*** END OF REPORT ***
```

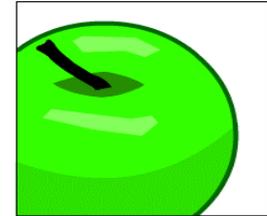
Tell Me / Show Me

Terminology

Key terms used in this lesson include:

DBMS_OUTPUT package

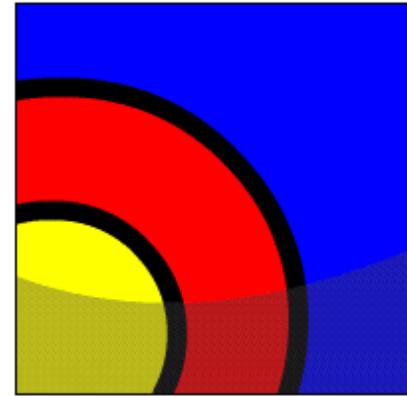
UTL_FILE package



Summary

In this lesson, you learned to:

- Describe two common uses for the `DBMS_OUTPUT` server-supplied package
- Recognize the correct syntax to specify messages for the `DBMS_OUTPUT` package
- Describe the purpose for the `UTL_FILE` server-supplied package.
- Recall the exceptions used in conjunction with the `UTL_FILE` server-supplied package.



Try It / Solve It

The exercises in this lesson cover the following topics:

- Describing two common uses for the `DBMS_OUTPUT` server-supplied package
- Recognizing the correct syntax to specify messages for the `DBMS_OUTPUT` package
- Describing the purpose for the `UTL_FILE` server-supplied package.
- Recalling the exceptions used in conjunction with the `UTL_FILE` server-supplied package.

