289
UTL_FILE

With the UTL_FILE package, PL/SQL programs can read and write operating system text files. UTL_FILE provides a restricted version of operating system stream file I/O.

This chapter contains the following topics:

UTL_FILE Security Model

The set of files and directories that are accessible to the user through UTL_FILE is controlled by a number of factors and database parameters. Foremost of these is the set of directory objects that have been granted to the user.

The nature of directory objects is discussed in the *Oracle Database SQL Language Reference*.

Assuming the user has both READ and WRITE access to the directory object USER_DIR, the user can open a file located in the operating system directory described by USER_DIR, but not in subdirectories or parent directories of this directory.

Lastly, the client (text I/O) and server implementations are subject to operating system file permission checking.

UTL_FILE provides file access both on the client side and on the server side. When run on the server, UTL_FILE provides access to all operating system files that are accessible from the server. On the client side, as in the case for Forms applications, UTL_FILE provides access to operating system files that are accessible from the client.

Directory objects offer more flexibility and granular control to the UTL_FILE application administrator, can be maintained dynamically (that is, without shutting down the database), and are consistent with other Oracle tools. CREATE ANY DIRECTORY privilege is granted only to SYS and SYSTEM by default.

> **Note:**
>
> Use the CREATE DIRECTORY feature for directory access verification.

Note that neither hard nor symbolic links are supported.

On UNIX systems, the owner of a file created by the FOPEN function is the owner of the shadow process running the instance. Normally, this owner is ORACLE. Files created using FOPEN are always writable and readable using the UTL_FILE subprograms. However, non-privileged operating system users who need to read these files outside of PL/SQL may need access from a system administrator.

> **WARNING:**
>
> The privileges needed to access files in a directory object are operating system specific. UTL_FILE directory object privileges give you read and write access to all files within the specified directory.

UTL_FILE Operational Notes

Keep these notes in mind when using UTL_FILE.

The file location and file name parameters are supplied to the FOPEN function as separate strings, so that the file location can be checked against the list of accessible directories as specified by the ALL_DIRECTORIES view of accessible directory objects. Together, the file location and name must represent a legal filename on the system, and the directory must be accessible. A subdirectory of an accessible directory is not necessarily also accessible; it too must be specified using a complete path name matching an ALL_DIRECTORIES object.

UTL_FILE implicitly interprets line terminators on read requests, thereby affecting the number of bytes returned on a GET_LINE call. For example, the len parameter of UTL_FILE.GET_LINE specifies the requested number of bytes of character data. The number of bytes actually returned to the user will be the lesser of:

- The GET_LINE len parameter, or
- The number of bytes until the next line terminator character, or
- The max_linesize parameter specified by UTL_FILE.FOPEN

The FOPEN max_linesize parameter must be a number in the range 1 and 32767. If unspecified, Oracle supplies a default value of 1024. The GET_LINE len parameter must be a number in the range 1 and 32767. If unspecified, Oracle supplies the default value of max_linesize. If max_linesize and len are defined to be different values, then the lesser value takes precedence.

UTL_FILE.GET_RAW ignores line terminators.

UTL_FILE expects that files opened by UTL_FILE.FOPEN in text mode are encoded in the database character set. It expects that files opened by UTL_FILE.FOPEN_NCHAR in text mode are encoded in the UTF8 character set. If an opened file is not encoded in the expected character set, the result of an attempt to read the file is indeterminate. When data encoded in one character set is read and Globalization Support is told (such as by means of NLS_LANG) that it is encoded in another character set, the result is indeterminate. If NLS_LANG is set, it should therefore be the same as the database character set.

UTL_FILE Rules and Limits

Operating system-specific parameters, such as C-shell environment variables under UNIX, cannot be used in the file location or file name parameters.

`UTL_FILE` I/O capabilities are similar to standard operating system stream file I/O (`OPEN`, `GET`, `PUT`, `CLOSE`) capabilities, but with some limitations. For example, you call the `FOPEN` function to return a file handle, which you use in subsequent calls to `GET_LINE` or `PUT` to perform stream I/O to a file. When file I/O is done, you call `FCLOSE` to complete any output and free resources associated with the file.

> **Note:**
>
> The `UTL_FILE` package is similar to the client-side `TEXT_IO` package currently provided by Oracle Procedure Builder. Restrictions for a server implementation require some API differences between `UTL_FILE` and `TEXT_IO`. In PL/SQL file I/O, errors are returned using PL/SQL exceptions.

UTL_FILE Exceptions

This table describes exceptions raised by `UTL_FILE` subprograms.

Table 289-1 UTL_FILE Package Exceptions

Exception Name	Description
`INVALID_PATH`	File location is invalid.
`INVALID_MODE`	The `open_mode` parameter in `FOPEN` is invalid.
`INVALID_FILEHANDLE`	File handle is invalid.
`INVALID_OPERATION`	File could not be opened or operated on as requested.
`READ_ERROR`	Destination buffer too small, or operating system error occurred during the read operation
`WRITE_ERROR`	Operating system error occurred during the write operation.
`INTERNAL_ERROR`	Unspecified PL/SQL error
`CHARSETMISMATCH`	A file is opened using `FOPEN_NCHAR`, but later I/O operations use nonchar functions such as `PUTF` or `GET_LINE`.
`FILE_OPEN`	The requested operation failed because the file is open.
`INVALID_MAXLINESIZE`	The `MAX_LINESIZE` value for `FOPEN()` is invalid; it should be within the range 1 to 32767.
`INVALID_FILENAME`	The filename parameter is invalid.
`ACCESS_DENIED`	Permission to access to the file location is denied.
`INVALID_OFFSET`	Causes of the `INVALID_OFFSET` exception: • `ABSOLUTE_OFFSET` = `NULL` and `RELATIVE_OFFSET = NULL`, or • `ABSOLUTE_OFFSET` < 0, or • Either offset caused a seek past the end of the file

Table 289-1 (Cont.) UTL_FILE Package Exceptions

Exception Name	Description
DELETE_FAILED	The requested file delete operation failed.
RENAME_FAILED	The requested file rename operation failed.

Procedures in UTL_FILE can also raise predefined PL/SQL exceptions such as NO_DATA_FOUND or VALUE_ERROR.

UTL_FILE Examples

These two examples show use of the procedure.

Example 1

> **Note:**
>
> The examples are UNIX-specific.

Given the following:

```
SQL> CREATE DIRECTORY log_dir AS '/appl/gl/log';
SQL> GRANT READ ON DIRECTORY log_dir TO DBA;
SQL> GRANT WRITE ON DIRECTORY log_dir TO DBA;

SQL> CREATE DIRECTORY USER_DIR AS '/appl/gl/user'';
SQL> GRANT READ ON DIRECTORY USER_DIR TO PUBLIC;
SQL> GRANT WRITE ON DIRECTORY USER_DIR TO PUBLIC;
```

The following file locations and filenames are valid and accessible as follows:

File Location	Filename	READ and WRITE
/appl/gl/log	L12345.log	Users with DBA privilege
/appl/gl/user	u12345.tmp	All users

The following file locations and filenames are invalid:

File Location	Filename	Invalid Because
/appl/gl/log/backup	L12345.log	# subdirectories are not accessible
/APPL/gl/log	L12345.log	# directory strings must follow case sensitivity rules as required by the O/S
/appl/gl/log	backup/L1234.log	# filenames may not include portions of directory paths
/user/tmp	L12345.log	# no corresponding CREATE DIRECTORY command has been issued

Example 2

```
DECLARE
  V1 VARCHAR2(32767);
  F1 UTL_FILE.FILE_TYPE;
BEGIN
  -- In this example MAX_LINESIZE is less than GET_LINE's length request
  -- so the number of bytes returned will be 256 or less if a line terminator is seen.
  F1 := UTL_FILE.FOPEN('USER_DIR','u12345.tmp','R',256);
  UTL_FILE.GET_LINE(F1,V1,32767);
  UTL_FILE.FCLOSE(F1);

  -- In this example, FOPEN's MAX_LINESIZE is NULL and defaults to 1024,
  -- so the number of bytes returned will be 1024 or less if a line terminator is seen.
  F1 := UTL_FILE.FOPEN('USER_DIR','u12345.tmp','R');
  UTL_FILE.GET_LINE(F1,V1,32767);
  UTL_FILE.FCLOSE(F1);

  -- In this example, GET_LINE doesn't specify a number of bytes, so it defaults to
  -- the same value as FOPEN's MAX_LINESIZE which is NULL in this case and defaults to
1024.
  -- So the number of bytes returned will be 1024 or less if a line terminator is seen.
  F1 := UTL_FILE.FOPEN('USER_DIR','u12345.tmp','R');
  UTL_FILE.GET_LINE(F1,V1);
  UTL_FILE.FCLOSE(F1);
END;
```

UTL_FILE Data Structures

The `UTL_FILE` package defines a RECORD type.

Record Types

- FILETYPE Record Type

FILETYPE Record Type

The contents of `FILE_TYPE` are private to the `UTL_FILE` package. You should not reference or change components of this record.

```
TYPE file_type IS RECORD (
   id          BINARY_INTEGER,
   datatype    BINARY_INTEGER,
   byte_mode   BOOLEAN);
```

Fields

Table 289-2 FILE_TYPE Fields

Field	Description
`id`	A numeric value indicating the internal file handle number
`datatype`	Indicates whether the file is a `CHAR` file, `Nchar` file or other (binary)
`byte_mode`	Indicates whether the file was open as a binary file, or as a text file

> **Caution:**
>
> Oracle does not guarantee the persistence of `FILE_TYPE` values between database sessions or within a single session. Attempts to clone file handles or use dummy file handles may have inderterminate outcomes.

Summary of UTL_FILE Subprograms

This table lists the `UTL_FILE` subprograms and briefly describes them.

Table 289-3 UTL_FILE Subprograms

Subprogram	Description
FCLOSE Procedure	Closes a file
FCLOSE_ALL Procedure	Closes all open file handles
FCOPY Procedure	Copies a contiguous portion of a file to a newly created file
FFLUSH Procedure	Physically writes all pending output to a file
FGETATTR Procedure	Reads and returns the attributes of a disk file
FGETPOS Function	Returns the current relative offset position within a file, in bytes
FOPEN Function	Opens a file for input or output
FOPEN_NCHAR Function	Opens a file in Unicode for input or output
FREMOVE Procedure	Deletes a disk file, assuming that you have sufficient privileges
FRENAME Procedure	Renames an existing file to a new name, similar to the UNIX `mv` function
FSEEK Procedure	Adjusts the file pointer forward or backward within the file by the number of bytes specified
GET_LINE Procedure	Reads text from an open file
GET_LINE_NCHAR Procedure	Reads text in Unicode from an open file
GET_RAW Procedure	Reads a `RAW` string value from a file and adjusts the file pointer ahead by the number of bytes read
IS_OPEN Function	Determines if a file handle refers to an open file
NEW_LINE Procedure	Writes one or more operating system-specific line terminators to a file
PUT Procedure	Writes a string to a file
PUT_LINE Procedure	Writes a line to a file, and so appends an operating system-specific line terminator
PUT_LINE_NCHAR Procedure	Writes a Unicode line to a file
PUT_NCHAR Procedure	Writes a Unicode string to a file
PUTF Procedure	A `PUT` procedure with formatting
PUTF_NCHAR Procedure	A `PUT_NCHAR` procedure with formatting, and writes a Unicode string to a file, with formatting
PUT_RAW Procedure	Accepts as input a `RAW` data value and writes the value to the output buffer

FCLOSE Procedure

This procedure closes an open file identified by a file handle.

Syntax

```
UTL_FILE.FCLOSE (
   file IN OUT FILE_TYPE);
```

Parameters

Table 289-4 FCLOSE Procedure Parameters

Parameter	Description
file	Active file handle returned by an FOPEN or FOPEN_NCHAR call

Usage Notes

If there is buffered data yet to be written when FCLOSE runs, then you may receive a WRITE_ERROR exception when closing a file.

Exceptions

```
WRITE_ERROR
INVALID_FILEHANDLE
```

FCLOSE_ALL Procedure

This procedure closes all open file handles for the session. This should be used as an emergency cleanup procedure, for example, when a PL/SQL program exits on an exception.

Syntax

```
UTL_FILE.FCLOSE_ALL;
```

Usage Notes

> **Note:**
>
> FCLOSE_ALL does not alter the state of the open file handles held by the user. This means that an IS_OPEN test on a file handle after an FCLOSE_ALL call still returns TRUE, even though the file has been closed. No further read or write operations can be performed on a file that was open before an FCLOSE_ALL.

Exceptions

```
WRITE_ERROR
```

FCOPY Procedure

This procedure copies a contiguous portion of a file to a newly created file.

By default, the whole file is copied if the `start_line` and `end_line` parameters are omitted. The source file is opened in read mode. The destination file is opened in write mode. A starting and ending line number can optionally be specified to select a portion from the center of the source file for copying.

Syntax

```
UTL_FILE.FCOPY (
   src_location    IN VARCHAR2,
   src_filename    IN VARCHAR2,
   dest_location   IN VARCHAR2,
   dest_filename   IN VARCHAR2,
   start_line      IN BINARY_INTEGER DEFAULT 1,
   end_line        IN BINARY_INTEGER DEFAULT NULL);
```

Parameters

Table 289-5 FCOPY Procedure Parameters

Parameters	Description
`src_location`	Directory location of the source file, a `DIRECTORY_NAME` from the `ALL_DIRECTORIES` view (case sensitive)
`src_filename`	Source file to be copied
`dest_location`	Destination directory where the destination file is created
`dest_filename`	Destination file created from the source file
`start_line`	Line number at which to begin copying. The default is `1` for the first line
`end_line`	Line number at which to stop copying. The default is `NULL`, signifying end of file

Exceptions

`INVALID_FILENAME`

`INVALID_PATH`

`INVALID_OPERATION`

`INVALID_OFFSET`

`READ_ERROR`

`WRITE_ERROR`

FFLUSH Procedure

`FFLUSH` physically writes pending data to the file identified by the file handle. Normally, data being written to a file is buffered. The `FFLUSH` procedure forces the buffered data to be written to the file. The data must be terminated with a newline character.

Flushing is useful when the file must be read while still open. For example, debugging messages can be flushed to the file so that they can be read immediately.

Syntax

```
UTL_FILE.FFLUSH (
   file  IN FILE_TYPE);
```

Parameters

Table 289-6 FFLUSH Procedure Parameters

Parameters	Description
file	Active file handle returned by an FOPEN or FOPEN_NCHAR call

Exceptions

INVALID_FILENAME

INVALID_MAXLINESIZE

INVALID_OPERATION

WRITE_ERROR

FGETATTR Procedure

This procedure reads and returns the attributes of a disk file.

Syntax

```
UTL_FILE.FGETATTR(
   location     IN VARCHAR2,
   filename     IN VARCHAR2,
   fexists      OUT BOOLEAN,
   file_length  OUT NUMBER,
   block_size   OUT BINARY_INTEGER);
```

Parameters

Table 289-7 FGETATTR Procedure Parameters

Parameters	Description
location	Directory location of the source file, a DIRECTORY_NAME from the ALL_DIRECTORIES view (case sensitive)
filename	Name of the file to be examined
fexists	A BOOLEAN for whether or not the file exists
file_length	Length of the file in bytes. NULL if file does not exist.
block_size	File system block size in bytes. NULL if the file does not exist.

Exceptions

INVALID_PATH

INVALID_FILENAME

INVALID_OPERATION

READ_ERROR

ACCESS_DENIED

FGETPOS Function

This function returns the current relative offset position within a file, in bytes.

Syntax

```
UTL_FILE.FGETPOS (
   file IN FILE_TYPE)
 RETURN PLS_INTEGER;
```

Parameters

Table 289-8 FGETPOS Parameters

Parameters	Description
file	Directory location of the source file

Return Values

FGETPOS returns the relative offset position for an open file, in bytes. It raises an exception if the file is not open. It returns 0 for the beginning of the file.

Exceptions

INVALID_FILEHANDLE

INVALID_OPERATION

READ_ERROR

Usage Notes

If file is opened for byte mode operations, then the INVALID OPERATION exception is raised.

FOPEN Function

This function opens a file. You can specify the maximum line size and have a maximum of 50 files open simultaneously.

See also FOPEN_NCHAR Function.

Syntax

```
UTL_FILE.FOPEN (
   location     IN VARCHAR2,
   filename     IN VARCHAR2,
   open_mode    IN VARCHAR2,
   max_linesize IN BINARY_INTEGER DEFAULT 1024)
  RETURN FILE_TYPE;
```

Parameters

Table 289-9 FOPEN Function Parameters

Parameter	Description
location	Directory location of file. This string is a directory object name and must be specified in upper case. Read privileges must be granted on this directory object for the UTL_FILE user to run FOPEN.
filename	File name, including extension (file type), without directory path. If a directory path is given as a part of the filename, it is ignored by FOPEN. On Unix, the filename cannot end with /.
open_mode	Specifies how the file is opened. Modes include: • r -- read text • w -- write text • a -- append text • rb -- read byte mode • wb -- write byte mode • ab -- append byte mode If you try to open a file specifying 'a' or 'ab' for open_mode but the file does not exist, the file is created in write mode.
max_linesize	Maximum number of bytes for each line, including the newline character, for this file (minimum value 1, maximum value 32767). If unspecified, Oracle supplies a default value of 1024.

Return Values

FOPEN returns a file handle, which must be passed to all subsequent procedures that operate on that file. The specific contents of the file handle are private to the UTL_FILE package, and individual components should not be referenced or changed by the UTL_FILE user.

Table 289-10 FOPEN Function Return Values

Return	Description
FILE_TYPE	Handle to open file

Exceptions

INVALID_MAXILINESIZE

INVALID_MODE

INVALID_OPERATION

INVALID_PATH

INVALID_FILENAME

Usage Notes

The file location and file name parameters must be supplied to the FOPEN function as quoted strings so that the file location can be checked against the list of accessible directories as specified by the ALL_DIRECTORIES view of accessible directory objects.

FOPEN_NCHAR Function

This function opens a file in national character set mode for input or output, with the maximum line size specified. With this function, you can read or write a text file in Unicode instead of in the database character set.

You can have a maximum of 50 files open simultaneously.

Even though the contents of an `NVARCHAR2` buffer may be AL16UTF16 or UTF8 (depending on the national character set of the database), the contents of the file are always read and written in UTF8. `UTL_FILE` converts between UTF8 and AL16UTF16 as necessary.

See also FOPEN Function.

Syntax

```
UTL_FILE.FOPEN_NCHAR (
   location     IN VARCHAR2,
   filename     IN VARCHAR2,
   open_mode    IN VARCHAR2,
   max_linesize IN BINARY_INTEGER DEFAULT 1024)
RETURN FILE_TYPE;
```

Parameters

Table 289-11 FOPEN_NCHAR Function Parameters

Parameter	Description
`location`	Directory location of file
`filename`	File name (including extension)
`open_mode`	Open mode (`r`,`w`,`a`,`rb`,`wb`,`ab`)
`max_linesize`	Maximum number of characters for each line, including the newline character, for this file (minimum value 1, maximum value 32767)

Return Values

`FOPEN_NCHAR` returns a file handle, which must be passed to all subsequent procedures that operate on that file. The specific contents of the file handle are private to the `UTL_FILE` package, and individual components should not be referenced or changed by the `UTL_FILE` user.

Table 289-12 FOPEN_NCHAR Function Return Values

Return	Description
`FILE_TYPE`	Handle to open file

Exceptions

`INVALID_MAXILINESIZE`

`INVALID_MODE`

`INVALID_OPERATION`

```
INVALID_PATH
```

FREMOVE Procedure

This procedure deletes a disk file, assuming that you have sufficient privileges.

Syntax

```
UTL_FILE.FREMOVE (
   location IN VARCHAR2,
   filename IN VARCHAR2);
```

Parameters

Table 289-13 FREMOVE Procedure Parameters

Parameters	Description
location	Directory location of the file, a DIRECTORY_NAME from ALL_DIRECTORIES (case sensitive)
filename	Name of the file to be deleted

Exceptions

```
ACCESS_DENIED

DELETE_FAILED

INVALID_FILENAME

INVALID_OPERATION

INVALID_PATH
```

Usage Notes

The FREMOVE procedure does not verify privileges before deleting a file. The O/S verifies file and directory permissions. An exception is returned on failure.

FRENAME Procedure

This procedure renames an existing file to a new name, similar to the UNIX mv function.

Syntax

```
UTL_FILE.FRENAME (
   src_location      IN   VARCHAR2,
   src_filename      IN   VARCHAR2,
   dest_location     IN   VARCHAR2,
   dest_filename     IN   VARCHAR2,
   overwrite         IN   BOOLEAN DEFAULT FALSE);
```

Parameters

Table 289-14 FRENAME Procedure Parameters

Parameters	Description
src_location	Directory location of the source file, a DIRECTORY_NAME from the ALL_DIRECTORIES view (case sensitive)
src_filename	Source file to be renamed
dest_location	Destination directory of the destination file, a DIRECTORY_NAME from the ALL_DIRECTORIES view (case sensitive)
dest_filename	New name of the file
overwrite	Default is FALSE. Permission on both the source and destination directories must be granted. You can use the overwrite parameter to specify whether or not to overwrite a file if one exists in the destination directory. The default is FALSE for no overwrite.

Exceptions

ACCESS_DENIED

INVALID_FILENAME

INVALID_PATH

RENAME_FAILED

FSEEK Procedure

This procedure adjusts the file pointer forward or backward within the file by the number of bytes specified.

Syntax

```
UTL_FILE.FSEEK (
   file             IN OUT  UTL_FILE.FILE_TYPE,
   absolute_offset  IN      PL_INTEGER DEFAULT NULL,
   relative_offset  IN      PLS_INTEGER DEFAULT NULL);
```

Parameters

Table 289-15 FSEEK Procedure Parameters

Parameters	Description
file	File handle
absolute_offset	Absolute location to which to seek; default = NULL
relative_offset	Number of bytes to seek forward or backward; positive = forward, negative integer = backward, zero = current position, default = NULL

Exceptions

INVALID_FILEHANDLE

INVALID_OFFSET

INVALID_OPERATION

READ_ERROR

Usage Notes

- Using FSEEK, you can read previous lines in the file without first closing and reopening the file. You must know the number of bytes by which you want to navigate.
- If relative_offset, the procedure seeks forward. If relative_offset > 0, or backward, if relative_offset < 0, the procedure seeks through the file by the number of relative_offset bytes specified.
- If the beginning of the file is reached before the number of bytes specified, then the file pointer is placed at the beginning of the file. If the end of the file is reached before the number of bytes specified, then an INVALID_OFFSET error is raised.
- If absolute_offset, the procedure seeks to an absolute location specified in bytes.
- If file is opened for byte mode operations, then the INVALID OPERATION exception is raised.

GET_LINE Procedure

This procedure reads text from the open file identified by the file handle and places the text in the output buffer parameter. Text is read up to, but not including, the line terminator, or up to the end of the file, or up to the end of the len parameter. It cannot exceed the max_linesize specified in FOPEN.

Syntax

```
UTL_FILE.GET_LINE (
   file        IN  FILE_TYPE,
   buffer      OUT VARCHAR2,
   len         IN  PLS_INTEGER DEFAULT NULL);
```

Parameters

Table 289-16 GET_LINE Procedure Parameters

Parameters	Description
file	Active file handle returned by an FOPEN call. The file must be open for reading (mode r); otherwise an INVALID_OPERATION exception is raised.
buffer	Data buffer to receive the line read from the file
len	The number of bytes read from the file. Default is NULL. If NULL, Oracle supplies the value of max_linesize.

Exceptions

INVALID_FILEHANDLE

INVALID_OPERATION

NO_DATA_FOUND

READ_ERROR

Usage Notes

If the line does not fit in the buffer, a `READ_ERROR` exception is raised. If no text was read due to end of file, the `NO_DATA_FOUND` exception is raised. If the file is opened for byte mode operations, the `INVALID_OPERATION` exception is raised.

Because the line terminator character is not read into the buffer, reading blank lines returns empty strings.

The maximum size of the `buffer` parameter is 32767 bytes unless you specify a smaller size in `FOPEN`.If unspecified, Oracle supplies a default value of 1024. See also "GET_LINE_NCHAR Procedure".

GET_LINE_NCHAR Procedure

This procedure reads text from the open file identified by the file handle and places the text in the output buffer parameter. With this function, you can read a text file in Unicode instead of in the database character set.

The file must be opened in national character set mode, and must be encoded in the UTF8 character set. The expected buffer datatype is `NVARCHAR2`. If a variable of another datatype, such as `NCHAR`, `NCLOB`, or `VARCHAR2` is specified, PL/SQL will perform standard implicit conversion from `NVARCHAR2` after the text is read.

See also GET_LINE Procedure

Syntax

```
UTL_FILE.GET_LINE_NCHAR (
   file        IN  FILE_TYPE,
   buffer      OUT NVARCHAR2,
   len         IN  PLS_INTEGER DEFAULT NULL);
```

Parameters

Table 289-17 GET_LINE_NCHAR Procedure Parameters

Parameters	Description
`file`	Active file handle returned by an `FOPEN_NCHAR` call. The file must be open for reading (mode `r`). If the file is opened by `FOPEN` instead of `FOPEN_NCHAR`, a `CHARSETMISMATCH` exception is raised.
`buffer`	Data buffer to receive the line read from the file
`len`	The number of bytes read from the file. Default is `NULL`. If `NULL`, Oracle supplies the value of max_linesize.

Exceptions

`INVALID_FILEHANDLE`

`INVALID_OPERATION`

`NO_DATA_FOUND`

`READ_ERROR`

GET_RAW Procedure

This procedure reads a `RAW` string value from a file and adjusts the file pointer ahead by the number of bytes read. `UTL_FILE.GET_RAW` ignores line terminators.

Syntax

```
UTL_FILE.GET_RAW (
   file       IN            UTL_FILE.FILE_TYPE,
   buffer     OUT NOCOPY    RAW,
   len        IN            PLS_INTEGER DEFAULT NULL);
```

Parameters

Table 289-18 GET_RAW Procedure Parameters

Parameters	Description
`file`	File handle
`buffer`	`RAW` data
`len`	The number of bytes read from the file. Default is `NULL`. If `NULL`, `len` is assumed to be the maximum length of `RAW`.

Exceptions

`INVALID_FILEHANDLE`

`INVALID_OPERATION`

`LENGTH_MISMATCH`

`NO_DATA_FOUND`

`READ_ERROR`

Usage Notes

The subprogram will raise `No_Data_Found` when it attempts to read past the end of the file. Your application should allow for this by catching the exception in its processing loop.

```
PROCEDURE Sys.p (n IN VARCHAR2) IS
      h     UTL_FILE.FILE_TYPE := UTL_FILE.FOPEN('D', n, 'r', 32767);
      Buf   RAW(32767);
      Amnt  CONSTANT PLS_INTEGER := 32767;
    BEGIN
      LOOP
        BEGIN
          Utl_File.Get_Raw(h, Buf, Amnt);
          -- Do something with this chunk
        EXCEPTION WHEN No_Data_Found THEN EXIT; END;
      END LOOP;
      UTL_FILE.FCLOSE (h);
    END;
```

IS_OPEN Function

This function tests a file handle to see if it identifies an open file.

`IS_OPEN` reports only whether a file handle represents a file that has been opened, but not yet closed. It does not guarantee that there will be no operating system errors when you attempt to use the file handle.

Syntax

```
UTL_FILE.IS_OPEN (
   file  IN FILE_TYPE)
  RETURN BOOLEAN;
```

Parameters

Table 289-19 IS_OPEN Function Parameters

Parameter	Description
`file`	Active file handle returned by an `FOPEN` or `FOPEN_NCHAR` call

Return Values

`TRUE` or `FALSE`

Exceptions

`INVALID_FILEHANDLE`

NEW_LINE Procedure

This procedure writes one or more line terminators to the file identified by the input file handle.

This procedure is separate from `PUT` because the line terminator is a platform-specific character or sequence of characters.

Syntax

```
UTL_FILE.NEW_LINE (
   file     IN FILE_TYPE,
   lines    IN BINARY_INTEGER := 1);
```

Parameters

Table 289-20 NEW_LINE Procedure Parameters

Parameters	Description
`file`	Active file handle returned by an `FOPEN` or `FOPEN_NCHAR` call
`lines`	Number of line terminators to be written to the file

Exceptions

`INVALID_FILEHANDLE`

INVALID_OPERATION

WRITE_ERROR

PUT Procedure

PUT writes the text string stored in the buffer parameter to the open file identified by the file handle.

The file must be open for write operations. No line terminator is appended by PUT; use NEW_LINE to terminate the line or use PUT_LINE to write a complete line with a line terminator. See also "PUT_NCHAR Procedure".

Syntax

```
UTL_FILE.PUT (
   file      IN FILE_TYPE,
   buffer    IN VARCHAR2);
```

Parameters

Table 289-21 PUT Procedure Parameters

Parameters	Description
file	Active file handle returned by an FOPEN_NCHAR call. The file must be open for writing.
buffer	Buffer that contains the text to be written to the file. User must have opened the file using mode w or mode a; otherwise, an INVALID_OPERATION exception is raised.

Usage Notes

The maximum size of the buffer parameter is 32767 bytes unless you specify a smaller size in FOPEN. If unspecified, Oracle supplies a default value of 1024. The sum of all sequential PUT calls cannot exceed 32767 without intermediate buffer flushes.

Exceptions

INVALID_FILEHANDLE

INVALID_OPERATION

WRITE_ERROR

PUT_LINE Procedure

This procedure writes the text string stored in the buffer parameter to the open file identified by the file handle.

The file must be open for write operations. PUT_LINE terminates the line with the platform-specific line terminator character or characters.

See also "PUT_LINE_NCHAR Procedure".

Syntax

```
UTL_FILE.PUT_LINE (
   file      IN FILE_TYPE,
   buffer    IN VARCHAR2,
   autoflush IN BOOLEAN DEFAULT FALSE);
```

Parameters

Table 289-22 PUT_LINE Procedure Parameters

Parameters	Description
`file`	Active file handle returned by an `FOPEN` call
`buffer`	Text buffer that contains the lines to be written to the file
`autoflush`	Flushes the buffer to disk after the `WRITE`

Exceptions

`INVALID_FILEHANDLE`

`INVALID_OPERATION`

`WRITE_ERROR`

Usage Notes

- The maximum size of the `buffer` parameter is 32767 bytes unless you specify a smaller size in `FOPEN`. If unspecified, Oracle supplies a default value of 1024. The sum of all sequential `PUT` calls cannot exceed 32767 without intermediate buffer flushes.
- If file is opened for byte mode operations, then the `INVALID OPERATION` exception is raised.

PUT_LINE_NCHAR Procedure

This procedure writes the text string stored in the buffer parameter to the open file identified by the file handle. With this function, you can write a text file in Unicode instead of in the database character set.

This procedure is equivalent to the PUT_NCHAR Procedure, except that the line separator is appended to the written text. See also PUT_LINE Procedure.

Syntax

```
UTL_FILE.PUT_LINE_NCHAR (
   file    IN FILE_TYPE,
   buffer  IN NVARCHAR2);
```

Parameters

Table 289-23 PUT_LINE_NCHAR Procedure Parameters

Parameters	Description
`file`	Active file handle returned by an `FOPEN_NCHAR` call. The file must be open for writing.

Table 289-23 (Cont.) PUT_LINE_NCHAR Procedure Parameters

Parameters	Description
buffer	Text buffer that contains the lines to be written to the file

Exceptions

INVALID_FILEHANDLE

INVALID_OPERATION

WRITE_ERROR

Usage Notes

- The maximum size of the buffer parameter is 32767 bytes unless you specify a smaller size in FOPEN. If unspecified, Oracle supplies a default value of 1024. The sum of all sequential PUT calls cannot exceed 32767 without intermediate buffer flushes.
- If file is opened for byte mode operations, then the INVALID OPERATION exception is raised.

PUT_NCHAR Procedure

This procedure writes the text string stored in the buffer parameter to the open file identified by the file handle.

With this function, you can write a text file in Unicode instead of in the database character set. The file must be opened in the national character set mode. The text string will be written in the UTF8 character set. The expected buffer datatype is NVARCHAR2. If a variable of another datatype is specified, PL/SQL will perform implicit conversion to NVARCHAR2 before writing the text.

Syntax

```
UTL_FILE.PUT_NCHAR (
   file      IN FILE_TYPE,
   buffer    IN NVARCHAR2);
```

Parameters

Table 289-24 PUT_NCHAR Procedure Parameters

Parameters	Description
file	Active file handle returned by an FOPEN_NCHAR call. If the file is opened by FOPEN instead of FOPEN_NCHAR, a CHARSETMISMATCH exception is raised.
buffer	Buffer that contains the text to be written to the file. User must have opened the file using mode w or mode a; otherwise, an INVALID_OPERATION exception is raised.

Exceptions

INVALID_FILEHANDLE

INVALID_OPERATION

```
WRITE_ERROR
```

Usage Notes

The maximum size of the `buffer` parameter is 32767 bytes unless you specify a smaller size in `FOPEN`. If unspecified, Oracle supplies a default value of 1024. The sum of all sequential `PUT` calls cannot exceed 32767 without intermediate buffer flushes.

Related Topics

- PUT Procedure
  `PUT` writes the text string stored in the buffer parameter to the open file identified by the file handle.

PUTF Procedure

This procedure is a formatted `PUT` procedure.

It works like a limited `printf`().

Syntax

```
UTL_FILE.PUTF (
   file    IN FILE_TYPE,
   format  IN VARCHAR2,
   [arg1   IN VARCHAR2  DEFAULT NULL,
   . . .
   arg5    IN VARCHAR2  DEFAULT NULL]);
```

Parameters

Table 289-25 PUTF Procedure Parameters

Parameters	Description
`file`	Active file handle returned by an `FOPEN` call
`format`	Format string that can contain text as well as the formatting characters `\n` and `%s`
`arg1..arg5`	From one to five operational argument strings. Argument strings are substituted, in order, for the `%s` formatters in the format string. If there are more formatters in the format parameter string than there are arguments, then an empty string is substituted for each `%s` for which there is no argument.

Usage Notes

- If file is opened for byte mode operations, then the `INVALID OPERATION` exception is raised.
- The format string can contain any text, but the character sequences `%s` and `\n` have special meaning.

Character Sequence	Meaning
`%s`	Substitute this sequence with the string value of the next argument in the argument list.
`\n`	Substitute with the appropriate platform-specific line terminator.

Exceptions

`INVALID_FILEHANDLE`

`INVALID_OPERATION`

`WRITE_ERROR`

Examples

The following example writes the lines:

```
Hello, world!
I come from Zork with greetings for all earthlings.

my_world  varchar2(4) := 'Zork';
...
PUTF(my_handle, 'Hello, world!\nI come from %s with %s.\n',
                my_world,
                'greetings for all earthlings');
```

If there are more `%s` formatters in the format parameter than there are arguments, then an empty string is substituted for each `%s` for which there is no matching argument.

Related Topics

- PUTF_NCHAR Procedure
  This procedure is a formatted version of a PUT_NCHAR Procedure.

PUTF_NCHAR Procedure

This procedure is a formatted version of a PUT_NCHAR Procedure.

Using `PUTF_NCHAR`, you can write a text file in Unicode instead of in the database character set. It accepts a format string with formatting elements `\n` and `%s`, and up to five arguments to be substituted for consecutive instances of `%s` in the format string. The expected datatype of the format string and the arguments is `NVARCHAR2`.

If variables of another datatype are specified, PL/SQL will perform implicit conversion to `NVARCHAR2` before formatting the text. Formatted text is written in the UTF8 character set to the file identified by the file handle. The file must be opened in the national character set mode.

Syntax

```
UTL_FILE.PUTF_NCHAR (
   file    IN FILE_TYPE,
   format  IN NVARCHAR2,
   [arg1   IN NVARCHAR2  DEFAULT NULL,
   . . .
   arg5    IN NVARCHAR2  DEFAULT NULL]);
```

Parameters

Table 289-26 PUTF_NCHAR Procedure Parameters

Parameters	Description
file	Active file handle returned by an FOPEN_NCHAR call. The file must be open for reading (mode r). If the file is opened by FOPEN instead of FOPEN_NCHAR, a CHARSETMISMATCH exception is raised.
format	Format string that can contain text as well as the formatting characters \n and %s
arg1..arg5	From one to five operational argument strings. Argument strings are substituted, in order, for the %s formatters in the format string. If there are more formatters in the format parameter string than there are arguments, then an empty string is substituted for each %s for which there is no argument.

Exceptions

INVALID_FILEHANDLE

INVALID_OPERATION

WRITE_ERROR

Usage Notes

- The maximum size of the buffer parameter is 32767 bytes unless you specify a smaller size in FOPEN. If unspecified, Oracle supplies a default value of 1024. The sum of all sequential PUT calls cannot exceed 32767 without intermediate buffer flushes.
- If file is opened for byte mode operations, then the INVALID OPERATION exception is raised.

Related Topics

- PUT_NCHAR Procedure
  This procedure writes the text string stored in the buffer parameter to the open file identified by the file handle.

PUT_RAW Procedure

This procedure accepts as input a RAW data value and writes the value to the output buffer.

Syntax

```
UTL_FILE.PUT_RAW (
   file          IN    UTL_FILE.FILE_TYPE,
   buffer        IN    RAW,
   autoflush     IN    BOOLEAN DEFAULT FALSE);
```

Parameters

Table 289-27 PUT_RAW Procedure Parameters

Parameters	Description
`file`	File handle
`buffer`	The `RAW` data written to the buffer
`autoflush`	If `TRUE`, then performs a flush after writing the value to the output buffer; default is `FALSE`.

Exceptions

`INVALID_FILEHANDLE`

`INVALID_OPERATION`

`WRITE_ERROR`

Usage Notes

You can request an automatic flush of the buffer by setting the third argument to `TRUE`.

The maximum size of the `buffer` parameter is 32767 bytes unless you specify a smaller size in `FOPEN`. If unspecified, Oracle supplies a default value of 1024. The sum of all sequential `PUT` calls cannot exceed 32767 without intermediate buffer flushes.