285 UTL_CALL_STACK

The `UTL_CALL_STACK` package provides an interface to provide information about currently executing subprograms.

Functions return subprogram names, unit names, owner names, edition names, and line numbers for given dynamic depths. Other functions return error stack information.

This chapter contains the following topics:

See Also:

- *Oracle Database PL/SQL Language Reference* regarding Conditional Compilation
- *Oracle Database Development Guide* regarding Using PL/Scope and Using the PL/SQL Hierarchical Profiler

UTL_CALL_STACK Overview

The `UTL_CALL_STACK` package provides an interface for PL/SQL programmers to obtain information about currently executing programs including the subprogram name from dynamic and lexical stacks and the depths of those stacks.

Individual functions return subprogram names, unit names, owner names, edition names, and line numbers for given dynamic depths. More functions return error stack information. Such information can be used to create more revealing error logs and application execution traces.

Dynamic Depth

The dynamic depth of an executing instance of a PL/SQL subprogram is defined recursively.

- The dynamic depth of the currently executing subprogram instance is one.
- Otherwise, the dynamic depth of the subprogram instance is one more than the dynamic depth of the subprogram it invoked.
- If there is a SQL, Java, or other non-PL/SQL context that invoked or was invoked by an executing subprogram, it occupies a level on the call stack as if it were a subprogram.

In the case of a call stack in which `A` calls `B`, which calls `C`, which calls `D`, which calls `E`, which calls `F`, which calls `E`, this stack can be written as a line with the dynamic depths underneath:

```
A B C D E F E
7 6 5 4 3 2 1
```

Lexical Depth

The lexical depth of a PL/SQL subprogram is defined recursively.

- The lexical depth of a unit, an anonymous block, trigger, or ADT is one (1).
- The lexical depth of a subprogram defined within another object is one plus the lexical depth of that object.

Blocks do not affect lexical depth.

Error Depth

The error depth is the number of errors on the error stack.

For example, consider the following anonymous block.

```
BEGIN
  BEGIN
    ... (1)
    raise zero_divide;
  EXCEPTION
    when others then
      raise no_data_found;
  END;
EXCEPTION
  WHEN others THEN
    ... (2)
END;
```

The error depth at (1) is zero and at (2) is two.

Backtrace

The backtrace is a trace from where the exception was thrown to where the backtrace was examined.

Consider a call stack in which A calls B which calls C and C raises an exception. If the backtrace was examined in C, the backtrace would have one unit, C, and the backtrace depth would be one. If it was examined in A, it would have three units, A, B and C, and backtrace depth would be three.

The depth of a backtrace is zero in the absence of an exception.

UTL_CALL_STACK Security Model

`EXECUTE` on `UTL_CALL_STACK` is granted to `PUBLIC`.

The `UTL_CALL_STACK` package does not show wrapped program units. For example, consider a call stack in which program unit `A` calls `B`, which calls `C`, and in turn calls `UTL_CALL_STACK` to determine the subprogram list. If program unit `B` is wrapped, then the subprogram list onlys shows program unit `C`.

UTL_CALL_STACK Operational Notes

Certain operational notes apply to `UTL_CALL_STACK`.

- Compiler optimizations can change lexical, dynamic and backtrace depth.
- `UTL_CALL_STACK` is not supported past RPC boundaries. For example, if A calls remote procedure B, B will not be able to obtain information about A using `UTL_CALL_STACK`.
- Lexical unit information is available through the PL/SQL conditional compilation feature and is therefore not exposed through `UTL_CALL_STACK`.

UTL_CALL_STACK Exceptions

This table lists the exceptions raised by `UTL_CALL_STACK`.

Table 285-1 Exceptions Raised by UTL_CALL_STACK

Exception	Error Code	Description
`BAD_DEPTH_INDICATOR`	`64610`	This exception is raised when a provided depth is out of bounds. Dynamic and lexical depth are positive integer values. Error and backtrace depths are non-negative integer values and are zero only in the absence of an exception.

UTL_CALL_STACK Data Structures

The `UTL_CALL_STACK` package defines a `VARRAY` type, `UNIT_QUALIFIED_NAME`.

VARRAY Type

- UNIT_QUALIFIED_NAME

UNIT_QUALIFIED_NAME

This data structure is a varray whose individual elements are, in order, the unit name, any lexical parents of the subprogram, and the subprogram name.

```
TYPE UNIT_QUALIFIED_NAME IS VARRAY(256) OF VARCHAR2(32767);
```

Example

Consider the following contrived PL/SQL procedure:

```
PROCEDURE topLevel IS
  FUNCTION localFunction(...) RETURNS VARCHAR2 IS
    FUNCTION innerFunction(...) RETURNS VARCHAR2 IS
      BEGIN
        DECLARE
          localVar PLS_INTEGER;
        BEGIN
          ... (1)
        END;
      END;
  BEGIN
```

```
    ...
  END;
```

The unit qualified name at (1) would be

```
["topLevel", "localFunction", "innerFunction"]
```

If the unit were an anonymous block, the unit name would be "`__anonymous_block`"

Summary of UTL_CALL_STACK Subprograms

This table lists the subprograms in the `UTL_CALL_STACK` package.

Table 285-2 UTL_CALL_STACK Package Subprograms

Subprogram	Description
BACKTRACE_DEPTH Function	Returns the number of backtrace items in the backtrace
BACKTRACE_LINE Function	Returns the line number of the unit at the specified backtrace depth
BACKTRACE_UNIT Function	Returns the name of the unit at the specified backtrace depth
CURRENT_EDITION Function	Returns the current edition name of the unit of the subprogram at the specified dynamic depth
CONCATENATE_SUBPROGRAM Function	Returns a concatenated form of a unit-qualified name
DYNAMIC_DEPTH Function	Returns the number of subprograms on the call stack
ERROR_DEPTH Function	Returns the number of errors on the error stack
ERROR_MSG Function	Returns the error message of the error at the specified error depth
ERROR_NUMBER Function	Returns the error number of the error at the specified error depth
LEXICAL_DEPTH Function	Returns the lexical nesting level of the subprogram at the specified dynamic depth
OWNER Function	Returns the owner name of the unit of the subprogram at the specified dynamic depth
UNIT_LINE Function	Returns the line number of the unit of the subprogram at the specified dynamic depth
SUBPROGRAM Function	Returns the unit-qualified name of the subprogram at the specified dynamic depth

BACKTRACE_DEPTH Function

This function returns the number of backtrace items in the backtrace.

Syntax

```
UTL_CALL_STACK.BACKTRACE_DEPTH
 RETURN PLS_INTEGER;
```

Return Values

The number of backtrace items in the backtrace, zero in the absence of an exception.

BACKTRACE_LINE Function

This function returns the line number of the unit at the specified backtrace depth.

Syntax

```
UTL_CALL_STACK.BACKTRACE_LINE (
   backtrace_depth   IN   PLS_INTEGER)
 RETURN PLS_INTEGER;
```

Parameters

Table 285-3 ***BACKTRACE_LINE Function Parameters***

Parameter	Description
`backtrace_depth`	Depth in backtrace

Return Values

The line number of the unit at the specified backtrace depth

BACKTRACE_UNIT Function

This function returns the name of the unit at the specified backtrace depth.

Syntax

```
UTL_CALL_STACK.BACKTRACE_UNIT (
   backtrace_depth   IN   PLS_INTEGER)
 RETURN VARCHAR2;
```

Parameters

Table 285-4 ***BACKTRACE_UNIT Function Parameters***

Parameter	Description
`backtrace_depth`	Depth in backtrace

Return Values

The name of the unit at the specified backtrace depth

CURRENT_EDITION Function

This function returns the current edition name of the unit of the subprogram at the specified dynamic depth.

Syntax

```
UTL_CALL_STACK.CURRENT_EDITION (
   dynamic_depth    IN    PLS_INTEGER)
 RETURN VARCHAR2;
```

Parameters

Table 285-5 *CURRENT_EDITION Function Parameters*

Parameter	Description
`dynamic_depth`	Depth in the error stack

Return Values

The current edition name of the unit of the subprogram at the specified dynamic depth

CONCATENATE_SUBPROGRAM Function

This function returns a concatenated form of a unit-qualified name.

Syntax

```
UTL_CALL_STACK.CONCATENATE_SUBPROGRAM (
   qualified_name    IN    UNIT_QUALIFIED_NAME)
 RETURN VARCHAR2;
```

Parameters

Table 285-6 *CONCATENATE_SUBPROGRAM Function Parameters*

Parameter	Description
`qualified_name`	A unit-qualified name

Return Values

A string of the form `UNIT.SUBPROGRAM.LOCAL_SUBPROGRAM`

DYNAMIC_DEPTH Function

This function returns the number of subprograms on the call stack.

Syntax

```
UTL_CALL_STACK.DYNAMIC_DEPTH
 RETURN PLS_INTEGER;
```

Return Values

The number of subprograms on the call stack

ERROR_DEPTH Function

This function returns the number of errors on the error stack.

Syntax

```
UTL_CALL_STACK.ERROR_DEPTH
 RETURN PLS_INTEGER;
```

Return Values

The number of errors on the error stack

ERROR_MSG Function

This function returns the error message of the error at the specified error depth.

Syntax

```
UTL_CALL_STACK.ERROR_MSG (
   error_depth    IN    PLS_INTEGER)
 RETURN VARCHAR2;
```

Parameters

Table 285-7 ***ERROR_MSG Function Parameters***

Parameter	Description
`error_depth`	Depth in the error stack

Return Values

The error message of the error at the specified error depth.

ERROR_NUMBER Function

This function returns the error number of the error at the specified error depth.

Syntax

```
UTL_CALL_STACK.ERROR_NUMBER (
   error_depth    IN    PLS_INTEGER)
 RETURN PLS_INTEGER;
```

Parameters

Table 285-8 ***ERROR_NUMBER Function Parameters***

Parameter	Description
`error_depth`	Depth in the call stack

Return Values

The error number of the error at the specified error depth

LEXICAL_DEPTH Function

This function returns the lexical nesting level of the subprogram at the specified dynamic depth.

Syntax

```
UTL_CALL_STACK.LEXICAL_DEPTH (
   dynamic_depth    IN    PLS_INTEGER)
 RETURN PLS_INTEGER;
```

Parameters

Table 285-9 ***LEXICAL_DEPTH Function Parameters***

Parameter	Description
`dynamic_depth`	Depth in the call stack

Return Values

The lexical nesting level of the subprogram at the specified dynamic depth

OWNER Function

This function returns the owner name of the unit of the subprogram at the specified dynamic depth.

Syntax

```
UTL_CALL_STACK.OWNER (
   dynamic_depth    IN    PLS_INTEGER)
 RETURN VARCHAR2;
```

Parameters

Table 285-10 ***OWNER Function Parameters***

Parameter	Description
`dynamic_depth`	Depth in the call stack

Return Values

The owner name of the unit of the subprogram at the specified dynamic depth

UNIT_LINE Function

This function returns the line number of the unit of the subprogram at the specified dynamic depth.

Syntax

```
UTL_CALL_STACK.UNIT_LINE (
   dynamic_depth    IN    PLS_INTEGER)
 RETURN PLS_INTEGER;
```

Parameters

Table 285-11 ***UNIT_LINE Function Parameters***

Parameter	Description
`dynamic_depth`	Depth in the call stack

Return Values

The line number of the unit of the subprogram at the specified dynamic depth

SUBPROGRAM Function

This function returns the unit-qualified name of the subprogram at the specified dynamic depth.

Syntax

```
UTL_CALL_STACK.SUBPROGRAM (
   dynamic_depth    IN    PLS_INTEGER)
 RETURN UNIT_QUALIFIED_NAME;
```

Parameters

Table 285-12 ***SUBPROGRAM Function Parameters***

Parameter	Description
`dynamic_depth`	Depth in the call stack

Return Values

Returns the unit-qualified name of the subprogram at the specified dynamic depth