298 UTL_RAW

The UTL_RAW package provides SQL functions for manipulating RAW datatypes.

This chapter contains the following topics:

- Overview
- Operational Notes
- Summary of UTL_RAW Subprograms

UTL_RAW Overview

This package is necessary because normal SQL functions do not operate on RAWs, and PL/SQL does not allow overloading between a RAW and a CHAR datatype. UTL_RAW also includes subprograms that convert various COBOL number formats to, and from, RAWs.

UTL_RAW is not specific to the database environment, and it may actually be used in other environments. For this reason, the prefix UTL has been given to the package, instead of DBMS.

UTL_RAW Operational Notes

UTL_RAW allows a RAW "record" to be composed of many elements. By using the RAW datatype, character set conversion will not be performed, keeping the RAW in its original format when being transferred through remote procedure calls.

With the RAW functions, you can manipulate binary data that was previously limited to the hextoraw and rawtohex functions.

Note:

Notes on datatypes:

- The PLS_INTEGER and BINARY_INTEGER datatypes are identical. This document uses BINARY_INTEGER to indicate datatypes in reference information (such as for table types, record types, subprogram parameters, or subprogram return values), but may use either in discussion and examples.
- The INTEGER and NUMBER(38) datatypes are also identical. This document uses INTEGER throughout.

Summary of UTL_RAW Subprograms

This table lists the UTL_RAW subprograms and briefly describes them.

Table 298-1 *UTL_RAW Package Subprograms*

Subprogram	Description
BIT_AND Function	Performs bitwise logical "and" of the values in `RAW r1` with `RAW r2` and returns the "anded" result `RAW`
BIT_COMPLEMENT Function	Performs bitwise logical "complement" of the values in `RAW r` and returns the "complement'ed" result `RAW`
BIT_OR Function	Performs bitwise logical "or" of the values in `RAW r1` with `RAW r2` and returns the "or'd" result `RAW`
BIT_XOR Function	Performs bitwise logical "exclusive or" of the values in `RAW r1` with `RAW r2` and returns the "xor'd" result `RAW`
CAST_FROM_BINARY_DOUBLE Function	Returns the `RAW` binary representation of a `BINARY_DOUBLE` value
CAST_FROM_BINARY_FLOAT Function	Returns the `RAW` binary representation of a `BINARY_FLOAT` value
CAST_FROM_BINARY_INTEGER Function	Returns the `RAW` binary representation of a `BINARY_INTEGER` value
CAST_FROM_NUMBER Function	Returns the `RAW` binary representation of a `NUMBER` value
CAST_TO_BINARY_DOUBLE Function	Casts the `RAW` binary representation of a `BINARY_DOUBLE` into a `BINARY_DOUBLE`
CAST_TO_BINARY_FLOAT Function	Casts the `RAW` binary representation of a `BINARY_FLOAT` into a `BINARY_FLOAT`
CAST_TO_BINARY_INTEGER Function	Casts the `RAW` binary representation of a `BINARY_INTEGER` into a `BINARY_INTEGER`
CAST_TO_NUMBER Function	Casts the `RAW` binary representation of a `NUMBER` into a `NUMBER`
CAST_TO_NVARCHAR2 Function	Converts a `RAW` value into a `VARCHAR2` value
CAST_TO_RAW Function	Converts a `VARCHAR2` value into a `RAW` value
CAST_TO_VARCHAR2 Function	Converts a RAW value into a `VARCHAR2` value
COMPARE Function	Compares `RAW r1` against `RAW r2`
CONCAT Function	Concatenates up to 12 `RAW`s into a single `RAW`
CONVERT Function	Converts `RAW r` from character set `from_charset` to character set `to_charset` and returns the resulting `RAW`
COPIES Function	Returns `n` copies of `r` concatenated together
LENGTH Function	Returns the length in bytes of a `RAW r`
OVERLAY Function	Overlays the specified portion of target `RAW` with overlay `RAW`, starting from byte position `pos` of target and proceeding for `len` bytes
REVERSE Function	Reverses a byte sequence in `RAW r` from end to end
SUBSTR Function	Returns `len` bytes, starting at `pos` from `RAW r`
TRANSLATE Function	Translates the bytes in the input `RAW r` according to the bytes in the translation `RAW`s `from_set` and `to_set`
TRANSLITERATE Function	Converts the bytes in the input `RAW r` according to the bytes in the transliteration `RAW`s `from_set` and `to_set`

Table 298-1 (Cont.) *UTL_RAW Package Subprograms*

Subprogram	Description
XRANGE Function	Returns a `RAW` containing all valid 1-byte encodings in succession, beginning with the value `start_byte` and ending with the value `end_byte`

BIT_AND Function

This function performs bitwise logical "and" of the values in `RAW r1` with `RAW r2` and returns the "anded" result `RAW`.

Syntax

```
UTL_RAW.BIT_AND (
   r1 IN RAW,
   r2 IN RAW)
RETURN RAW;
```

Pragmas

```
pragma restrict_references(bit_and, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-2 BIT_AND Function Parameters

Parameter	Description
`r1`	`RAW` to "and" with `r2`
`r2`	`RAW` to "and" with `r1`

Return Values

Table 298-3 BIT_AND Function Return Values

Return	Description
`RAW`	Containing the "and" of `r1` and `r2`
`NULL`	Either `r1` or `r2` input parameter was `NULL`

Usage Notes

If `r1` and r2 differ in length, the and operation is terminated after the last byte of the shorter of the two `RAW`s, and the unprocessed portion of the longer `RAW` is appended to the partial result. The result length equals the longer of the two input `RAW`s.

BIT_COMPLEMENT Function

This function performs bitwise logical "complement" of the values in `RAW r` and returns the complemented result `RAW`. The result length equals the input `RAW r` length.

Syntax

```
UTL_RAW.BIT_COMPLEMENT (
   r IN RAW)
  RETURN RAW;
```

Pragmas

```
pragma restrict_references(bit_complement, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-4 BIT_COMPLEMENT Function Parameters

Parameter	Description
`r`	`RAW` to perform "complement" operation

Return Values

Table 298-5 BIT_COMPLEMENT Function Return Values

Return	Description
`RAW`	The "complement" of `r1`
`NULL`	If `r` input parameter was `NULL`

BIT_OR Function

This function performs bitwise logical "or" of the values in `RAW r1` with `RAW r2` and returns the or'd result `RAW`.

Syntax

```
UTL_RAW.BIT_OR (
   r1 IN RAW,
   r2 IN RAW)
  RETURN RAW;
```

Pragmas

```
pragma restrict_references(bit_or, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-6 BIT_OR Function Parameters

Parameters	Description
`r1`	`RAW` to "or" with `r2`

Table 298-6 (Cont.) BIT_OR Function Parameters

Parameters	Description
`r2`	`RAW` to "or" with `r1`

Return Values

Table 298-7 BIT_OR Function Return Values

Return	Description
`RAW`	Containing the "or" of `r1` and `r2`
`NULL`	Either `r1` or `r2` input parameter was `NULL`

Usage Notes

If `r1` and `r2` differ in length, then the "or" operation is terminated after the last byte of the shorter of the two `RAW`s, and the unprocessed portion of the longer `RAW` is appended to the partial result. The result length equals the longer of the two input `RAW`s.

BIT_XOR Function

This function performs bitwise logical "exclusive or" of the values in `RAW r1` with `RAW r2` and returns the xor'd result `RAW`.

Syntax

```
UTL_RAW.BIT_XOR (
   r1 IN RAW,
   r2 IN RAW)
  RETURN RAW;
```

Pragmas

```
pragma restrict_references(bit_xor, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-8 BIT_XOR Function Parameters

Parameter	Description
`r1`	`RAW` to "xor" with `r2`
`r2`	`RAW` to "xor" with `r1`

Return Values

Table 298-9 BIT_XOR Function Return Values

Return	Description
`RAW`	Containing the "xor" of `r1` and `r2`

Table 298-9 (Cont.) BIT_XOR Function Return Values

Return	Description
`NULL`	If either `r1` or `r2` input parameter was `NULL`

Usage Notes

If `r1` and `r2` differ in length, then the "xor" operation is terminated after the last byte of the shorter of the two `RAW`s, and the unprocessed portion of the longer `RAW` is appended to the partial result. The result length equals the longer of the two input `RAW`s.

CAST_FROM_BINARY_DOUBLE Function

This function returns the `RAW` binary representation of a `BINARY_DOUBLE` value.

Syntax

```
UTL_RAW.CAST_FROM_BINARY_DOUBLE(
   n          IN BINARY_DOUBLE,
   endianess IN PLS_INTEGER DEFAULT 1)
RETURN RAW;
```

Pragmas

```
pragma restrict_references(cast_from_binary_double, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-10 CAST_FROM_BINARY_DOUBLE Function Parameters

Parameter	Description
`n`	`BINARY_DOUBLE` value
`endianess`	A `BINARY_INTEGER` value indicating the endianess. The function recognizes the defined constants `big_endian` (1), `little_endian` (2), and `machine_endian` (3). The default is `big_endian`. A setting of `machine_endian` has the same effect as `big_endian` on a big endian machine, or the same effect as `little_endian` on a little endian machine.

Return Values

The binary representation of the `BINARY_DOUBLE` value, or `NULL` if the input is `NULL`.

Usage Notes

- An 8-byte `binary_double` value maps to the IEEE 754 double-precision format as follows:

```
byte 0: bit 63 ~ bit 56
byte 1: bit 55 ~ bit 48
byte 2: bit 47 ~ bit 40
byte 3: bit 39 ~ bit 32
byte 4: bit 31 ~ bit 24
byte 5: bit 23 ~ bit 16
byte 6: bit 15 ~ bit  8
byte 7: bit  7 ~ bit  0
```

- The parameter endianess describes how the bytes of `BINARY_DOUBLE` are mapped to the bytes of `RAW`. In the following matrix, rb0 ~ rb7 refer to the bytes in raw and db0 ~ db7 refer to the bytes in `BINARY_DOUBLE`.

endianess	rb0	rb1	rb2	rb3	rb4	rb5	rb6	rb7
big_endian	db0	db1	db2	db3	db4	db5	db6	db7
little_endian	db7	db6	db5	db4	db3	db2	db1	db0

- In case of machine-endian, the 8 bytes of the `BINARY_DOUBLE` argument are copied straight across into the `RAW` return value. The effect is the same if the user has passed `big_endian` on a big-endian machine, or `little_endian` on a little-endian machine.

CAST_FROM_BINARY_FLOAT Function

This function returns the `RAW` binary representation of a `BINARY_FLOAT` value.

Syntax

```
UTL_RAW.CAST_FROM_BINARY_FLOAT(
   n           IN BINARY_FLOAT,
   endianess IN PLS_INTEGER DEFAULT 1)
RETURN RAW;
```

Pragmas

```
pragma restrict_references(cast_from_binary_float, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-11 CAST_FROM_BINARY_FLOAT Function Parameters

Parameter	Description
`n`	`BINARY_FLOAT` value
`endianess`	A `BINARY_INTEGER` value indicating the endianess. The function recognizes the defined constants `big_endian` (1), `little_endian` (2), and `machine_endian` (3). The default is `big_endian`. A setting of `machine_endian` has the same effect as `big_endian` on a big endian machine, or the same effect as `little_endian` on a little endian machine.

Return Values

The binary representation (RAW) of the `BINARY_FLOAT` value, or `NULL` if the input is `NULL`.

Usage Notes

- A 4-byte `binary_float` value maps to the IEEE 754 single-precision format as follows:

```
byte 0: bit 31 ~ bit 24
byte 1: bit 23 ~ bit 16
byte 2: bit 15 ~ bit  8
byte 3: bit 7 ~  bit  0
```

- The parameter endianess describes how the bytes of `BINARY_FLOAT` are mapped to the bytes of `RAW`. In the following matrix, rb0 ~ rb3 refer to the bytes in `RAW` and fb0 ~ fb3 refer to the bytes in `BINARY_FLOAT`.

Endianess	rb0	rb1	rb2	rb3
big_endian	fb0	fb1	fb2	fb3
little_endian	fb3	fb2	fb1	fb0

- In case of machine-endian, the 4 bytes of the `BINARY_FLOAT` argument are copied straight across into the `RAW` return value. The effect is the same if the user has passed `big_endian` on a big-endian machine, or `little_endian` on a little-endian machine.

CAST_FROM_BINARY_INTEGER Function

This function returns the `RAW` binary representation of a `BINARY_INTEGER` value.

Syntax

```
UTL_RAW.CAST_FROM_BINARY_INTEGER (
   n           IN BINARY_INTEGER
   endianess   IN PLS_INTEGER DEFAULT BIG_ENDIAN)
RETURN RAW;
```

Pragmas

```
pragma restrict_references(cast_from_binary_integer, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-12 CAST_FROM_BINARY_INTEGER Function Parameters

Parameter	Description
`n`	`BINARY_INTEGER` value.
`endianess`	A `BINARY_INTEGER` value indicating the endianess. The function recognizes the defined constants `big_endian` (1), `little_endian` (2), and `machine_endian` (3). The default is `big_endian`. A setting of `machine_endian` has the same effect as `big_endian` on a big endian machine, or the same effect as `little_endian` on a little endian machine.

Return Values

The binary representation of the `BINARY_INTEGER` value.

CAST_FROM_NUMBER Function

This function returns the `RAW` binary representation of a `NUMBER` value.

Syntax

```
UTL_RAW.CAST_FROM_NUMBER (
   n  IN NUMBER)
 RETURN RAW;
```

Pragmas

```
pragma restrict_references(cast_from_number, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-13 CAST_FROM_NUMBER Function Parameters

Parameter	Description
`n`	`NUMBER` value

Return Values

The binary representation of the `NUMBER` value.

CAST_TO_BINARY_DOUBLE Function

This function casts the RAW binary representation of a `BINARY_DOUBLE` into a `BINARY_DOUBLE`.

Syntax

```
UTL_RAW.CAST_TO_BINARY_DOUBLE (
   r          IN RAW
   endianess  IN PLS_INTEGER DEFAULT 1)
RETURN BINARY_DOUBLE;
```

Pragmas

```
pragma restrict_references(cast_to_binary_double, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-14 CAST_TO_BINARY_DOUBLE Function Parameters

Parameter	Description
`r`	Binary representation of a `BINARY_DOUBLE`
`endianess`	A `PLS_INTEGER` representing big-endian or little-endian architecture. The default is big-endian.

Return Values

The `BINARY_DOUBLE` value.

Usage Notes

- If the `RAW` argument is more than 8 bytes, only the first 8 bytes are used and the rest of the bytes are ignored. If the result is `-0`, `+0` is returned. If the result is `NaN`, the value `BINARY_DOUBLE_NAN` is returned.
- If the `RAW` argument is less than 8 bytes, a `VALUE_ERROR` exception is raised.
- An 8-byte binary_double value maps to the IEEE 754 double-precision format as follows:

  ```
  byte 0: bit 63 ~ bit 56
  byte 1: bit 55 ~ bit 48
  byte 2: bit 47 ~ bit 40
  ```

```
byte 3: bit 39 ~ bit 32
byte 4: bit 31 ~ bit 24
byte 5: bit 23 ~ bit 16
byte 6: bit 15 ~ bit  8
byte 7: bit  7 ~ bit  0
```

- The parameter endianess describes how the bytes of `BINARY_DOUBLE` are mapped to the bytes of `RAW`. In the following matrix, rb0 ~ rb7 refer to the bytes in raw and db0 ~ db7 refer to the bytes in `BINARY_DOUBLE`.

Architecture	rb0	rb1	rb2	rb3	rb4	rb5	rb6	rb7
big_endian	db0	db1	db2	db3	db4	db5	db6	db7
little_endian	db7	db6	db5	db4	db3	db2	db1	db0

- In case of machine-endian, the 8 bytes of the `RAW` argument are copied straight across into the `BINARY_DOUBLE` return value. The effect is the same if the user has passed `big_endian` on a big-endian machine, or `little_endian` on a little-endian machine.

CAST_TO_BINARY_FLOAT Function

This function casts the RAW binary representation of a `BINARY_FLOAT` into a `BINARY_FLOAT`.

Syntax

```
UTL_RAW.CAST_TO_BINARY_FLOAT (
   r           IN RAW
   endianess   IN PLS_INTEGER DEFAULT 1)
RETURN BINARY_FLOAT;
```

Pragmas

```
pragma restrict_references(cast_to_binary_float, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-15 CAST_TO_BINARY_FLOAT Function Parameters

Parameter	Description
`r`	Binary representation of a `BINARY_FLOAT`
`endianess`	A `PLS_INTEGER` representing big-endian or little-endian architecture. The default is big-endian.

Return Values

The `BINARY_FLOAT` value.

Usage Notes

- If the `RAW` argument is more than 4 bytes, only the first 4 bytes are used and the rest of the bytes are ignored. If the result is -0, +0 is returned. If the result is NaN, the value `BINARY_FLOAT_NAN` is returned.
- If the `RAW` argument is less than 4 bytes, a `VALUE_ERROR` exception is raised.
- A 4-byte `binary_float` value maps to the IEEE 754 single-precision format as follows:

```
byte 0: bit 31 ~ bit 24
byte 1: bit 23 ~ bit 16
byte 2: bit 15 ~ bit  8
byte 3: bit 7 ~  bit  0
```

- The parameter endianess describes how the bytes of `BINARY_FLOAT` are mapped to the bytes of `RAW`. In the following matrix, rb0 ~ rb3 refer to the bytes in `RAW` and fb0 ~ fb3 refer to the bytes in `BINARY_FLOAT`.

Endianness	rb0	rb1	rb2	rb3
big_endian	`fbo`	`fb1`	`fb2`	`fb3`
little_endian	`fb3`	`fb2`	`fb1`	`fb0`

- In case of machine-endian, the 4 bytes of the `RAW` argument are copied straight across into the `BINARY_FLOAT` return value. The effect is the same if the user has passed `big_endian` on a big-endian machine, or `little_endian` on a little-endian machine.

CAST_TO_BINARY_INTEGER Function

This function casts the RAW binary representation of a `BINARY_INTEGER` into a `BINARY_INTEGER`.

Syntax

```
UTL_RAW.CAST_TO_BINARY_INTEGER (
   r           IN RAW
   endianess   IN PLS_INTEGER DEFAULT BIG_ENDIAN)
RETURN BINARY_INTEGER;
```

Pragmas

```
pragma restrict_references(cast_to_binary_integer, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-16 CAST_TO_BINARY_INTEGER Function Parameters

Parameter	Description
`r`	Binary representation of a `BINARY_INTEGER`
`endianess`	A `PLS_INTEGER` representing big-endian or little-endian architecture. The default is big-endian.

Return Values

The `BINARY_INTEGER` value

CAST_TO_NUMBER Function

This function casts the RAW binary representation of a `NUMBER` into a `NUMBER`.

Syntax

```
UTL_RAW.CAST_TO_NUMBER (
   r  IN RAW)
 RETURN NUMBER;
```

Pragmas

```
pragma restrict_references(cast_to_number, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-17 CAST_TO_NUMBER function Parameters

Parameter	Description
`r`	Binary representation of a `NUMBER`

Return Values

The `NUMBER` value.

CAST_TO_NVARCHAR2 Function

This function converts a `RAW` value represented using some number of data bytes into an `NVARCHAR2` value with that number of data bytes.

Note:

When casting to a `NVARCHAR2`, the current Globalization Support character set is used for the characters within that `NVARCHAR2` value.

Syntax

```
UTL_RAW.CAST_TO_NVARCHAR2 (
   r IN RAW)
RETURN NVARCHAR2;
```

Pragmas

```
pragma restrict_references(cast_to_NVARCHAR2, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-18 CAST_TO_NVARCHAR2 Function Parameters

Parameter	Description
`r`	`RAW` (without leading length field) to be changed to a `NVARCHAR2`)

Return Values

Table 298-19 CAST_TO_NVARCHAR2 Function Return Values

Return	Description
`NVARCHAR2`	Containing having the same data as the input `RAW`
`NULL`	If `r` input parameter was `NULL`

CAST_TO_RAW Function

This function converts a VARCHAR2 value represented using some number of data bytes into a RAW value with that number of data bytes. The data itself is not modified in any way, but its datatype is recast to a RAW datatype.

Syntax

```
UTL_RAW.CAST_TO_RAW (
   c  IN VARCHAR2)
RETURN RAW;
```

Pragmas

```
pragma restrict_references(cast_to_raw, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-20 CAST_TO_RAW Function Parameters

Parameter	Description
c	VARCHAR2 to be changed to a RAW

Return Values

Table 298-21 CAST_TO_RAW Function Return Values

Return	Description
RAW	Containing the same data as the input VARCHAR2 and equal byte length as the input VARCHAR2 and without a leading length field
NULL	If c input parameter was NULL

CAST_TO_VARCHAR2 Function

This function converts a RAW value represented using some number of data bytes into a VARCHAR2 value with that number of data bytes.

Note:

When casting to a VARCHAR2, the current Globalization Support character set is used for the characters within that VARCHAR2.

Syntax

```
UTL_RAW.CAST_TO_VARCHAR2 (
   r IN RAW)
RETURN VARCHAR2;
```

Pragmas

```
pragma restrict_references(cast_to_VARCHAR2, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-22 CAST_TO_VARCHAR2 Function Parameters

Parameter	Description
`r`	`RAW` (without leading length field) to be changed to a `VARCHAR2`

Return Values

Table 298-23 CAST_TO_VARCHAR2 Function Return Values

Return	Description
`VARCHAR2`	Containing having the same data as the input `RAW`
`NULL`	If `r` input parameter was `NULL`

COMPARE Function

This function compares two `RAW` values. If they differ in length, then the shorter is extended on the right according to the optional `pad` parameter.

Syntax

```
UTL_RAW.COMPARE (
   r1   IN RAW,
   r2   IN RAW,
   pad IN RAW DEFAULT NULL)
  RETURN NUMBER;
```

Pragmas

```
pragma restrict_references(compare, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-24 COMPARE Function Parameters

Parameter	Description
`r1`	1st `RAW` to be compared, may be `NULL` or 0 length
`r2`	2nd `RAW` to be compared, may be `NULL` or 0 length
`pad`	This is an optional parameter. Byte to extend whichever of `r1` or `r2` is shorter. The default: x'00'

Return Values

Table 298-25 COMPARE Function Return Values

Return	Description
NUMBER	Equals 0 if RAW byte strings are both NULL or identical; or, Equals position (numbered from 1) of the first mismatched byte

CONCAT Function

This function concatenates up to 12 RAWs into a single RAW. If the concatenated size exceeds 32K, then an error is returned

Syntax

```
UTL_RAW.CONCAT (
   r1  IN RAW DEFAULT NULL,
   r2  IN RAW DEFAULT NULL,
   r3  IN RAW DEFAULT NULL,
   r4  IN RAW DEFAULT NULL,
   r5  IN RAW DEFAULT NULL,
   r6  IN RAW DEFAULT NULL,
   r7  IN RAW DEFAULT NULL,
   r8  IN RAW DEFAULT NULL,
   r9  IN RAW DEFAULT NULL,
   r10 IN RAW DEFAULT NULL,
   r11 IN RAW DEFAULT NULL,
   r12 IN RAW DEFAULT NULL)
  RETURN RAW;
```

Pragmas

```
pragma restrict_references(concat, WNDS, RNDS, WNPS, RNPS);
```

Parameters

r1....r12 are the RAW items to concatenate.

Return Values

Table 298-26 CONCAT Function Return Values

Return	Description
RAW	Containing the items concatenated

Exceptions

There is an error if the sum of the lengths of the inputs exceeds the maximum allowable length for a RAW, which is 32767 bytes.

CONVERT Function

This function converts `RAW` `r` from character set `from_charset` to character set `to_charset` and returns the resulting `RAW`.

Both `from_charset` and `to_charset` must be supported character sets defined to the Oracle server.

Syntax

```
UTL_RAW.CONVERT (
   r             IN RAW,
   to_charset    IN VARCHAR2,
   from_charset  IN VARCHAR2)
  RETURN RAW;
```

Pragmas

```
pragma restrict_references(convert, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-27   CONVERT Function Parameters

Parameter	Description
`r`	`RAW` byte-string to be converted
`to_charset`	Name of the character set to which `r` is converted
`from_charset`	Name of the character set in which `r` is supplied

Return Values

Table 298-28   CONVERT Function Return Values

Return	Description
`RAW`	Byte string `r` converted according to the specified character sets.

Exceptions

Table 298-29   CONVERT Function Exceptions

Error	Description
`ORA-06502`	PL/SQL: numeric or value error
`ORA-12703`	This character set conversion is not supported
`ORA-12705`	Cannot access NLS data files or invalid environment specified

Usage Notes

- The NLS_LANG parameter form *language_territory.character set* is also accepted for `to_charset` and `from_charset`. However, this form is deprecated and should be avoided. Note that *language* and *territory* are ignored by this subprogram.

- The converted value is silently truncated if it exceeds the maximum length of a RAW value, which is 32767 bytes. Do not convert values longer than floor(32767/4) = 8191 bytes if you want to avoid this truncation for all possible combinations of to_charset and from_charset. You can use the maximum character width of the target character set to_charset, if known, to expand the limit to a less pessimistic value. For example, if the target character set is ZHS16GBK, the maximum safe source string length is floor(32767/2) = 16383 bytes. For single-byte target character sets, no truncation is ever necessary.

COPIES Function

This function returns n copies of r concatenated together.

Syntax

```
UTL_RAW.COPIES (
   r  IN RAW,
   n  IN NUMBER)
  RETURN RAW;
```

Pragmas

```
pragma restrict_references(copies, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-30 COPIES Function Parameters

Parameters	Description
r	RAW to be copied
n	Number of times to copy the RAW (must be positive)

Return Values

This returns the RAW copied n times.

Exceptions

Table 298-31 COPIES Function Exceptions

Error	Description
VALUE_ERROR	Either: - r is missing, NULL or 0 length - n < 1 - Length of result exceeds maximum length of a RAW

LENGTH Function

This function returns the length in bytes of a `RAW r`.

Syntax

```
UTL_RAW.LENGTH (
   r  IN RAW)
RETURN NUMBER;
```

Pragmas

```
pragma restrict_references(length, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-32   LENGTH Function Parameters

Parameter	Description
`r`	`RAW` byte stream to be measured

Return Values

Table 298-33   LENGTH Function Return Values

Return	Description
`NUMBER`	Current length of the `RAW`

OVERLAY Function

This function overlays the specified portion of target `RAW` with `overlay_str RAW`, starting from byte position `pos` of `target` and proceeding for `len` bytes.

Syntax

```
UTL_RAW.OVERLAY (
   overlay_str IN RAW,
   target      IN RAW,
   pos         IN BINARY_INTEGER DEFAULT 1,
   len         IN BINARY_INTEGER DEFAULT NULL,
   pad         IN RAW            DEFAULT NULL)
  RETURN RAW;
```

Pragmas

```
pragma restrict_references(overlay, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-34   OVERLAY Function Parameters

Parameters	Description
`overlay_str`	Byte-string used to overlay target

Table 298-34 (Cont.) OVERLAY Function Parameters

Parameters	Description
`target`	Byte-string which is to be overlaid
`pos`	Position in target (numbered from 1) to start overlay
`len`	The number of target bytes to overlay
`pad`	Pad byte used when overlay `len` exceeds `overlay_str` length or `pos` exceeds `target` length

Defaults and Optional Parameters

Table 298-35 OVERLAY Function Optional Parameters

Optional Parameter	Description
`pos`	1
`len`	To the length of `overlay_str`
`pad`	x'00'

Return Values

Table 298-36 OVERLAY Function Return Values

Return	Description
`RAW`	The target `byte_string` overlaid as specified.

Usage Notes

If `overlay_str` has less than `len` bytes, then it is extended to `len` bytes using the `pad` byte. If `overlay_str` exceeds `len` bytes, then the extra bytes in `overlay_str` are ignored. If `len` bytes beginning at position `pos` of `target` exceeds the length of `target`, then `target` is extended to contain the entire length of `overlay_str`.

`If len` is specified, it must be greater than or equal to `0`. If `pos` is specified, it must be greater than or equal to `1`. If `pos` exceeds the length of `target`, then `target` is padded with `pad` bytes to position `pos`, and `target` is further extended with `overlay_str` bytes.

Exceptions

Table 298-37 OVERLAY Function Exceptions

Error	Description
`VALUE_ERROR`	Either: - `Overlay_str` is `NULL` or has 0 length - Target is missing or undefined - Length of target exceeds maximum length of a `RAW` - `len` < 0 - `pos` < 1

REVERSE Function

This function reverses a byte sequence in RAW r from end to end.

For example, x'0102F3' would be reversed to x'F30201', and 'xyz' would be reversed to 'zyx'.The result length is the same as the input RAW length.

Syntax

```
UTL_RAW.REVERSE (
   r IN RAW)
  RETURN RAW;
```

Pragmas

```
pragma restrict_references(reverse, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-38 REVERSE Function Parameters

Parameter	Description
r	RAW to reverse

Return Values

Table 298-39 REVERSE Function Return Values

Return	Description
RAW	Containing the "reverse" of r

Exceptions

Table 298-40 REVERSE Function Exceptions

Error	Description
VALUE_ERROR	R is NULL or has 0 length

SUBSTR Function

This function returns len bytes, starting at pos from RAW r.

Syntax

```
UTL_RAW.SUBSTR (
   r   IN RAW,
   pos IN BINARY_INTEGER,
   len IN BINARY_INTEGER DEFAULT NULL)
  RETURN RAW;
```

Pragmas

```
pragma restrict_references(substr, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-41 SUBSTR Function Parameters

Parameter	Description
`r`	`RAW` byte-string from which a portion is extracted
`pos`	Byte position in `r` at which to begin extraction
`len`	Number of bytes from `pos` to extract from `r` (optional)

Defaults and Optional Parameters

Table 298-42 SUBSTR Function Optional Parameter

Optional Parameter	Description
`len`	Position `pos` through to the end of `r`

Return Values

Table 298-43 SUBSTR Function Return Values

Return	Description
`portion of r`	Beginning at `pos` for `len` bytes long
`NULL`	`r` input parameter was `NULL`

Usage Notes

- If `pos` is positive, then `SUBSTR` counts from the beginning of `r` to find the first byte. If `pos` is negative, then `SUBSTR` counts backward from the end of the `r`. The value `pos` cannot be 0.
- If `len` is omitted, then `SUBSTR` returns all bytes to the end of `r`. The value `len` cannot be less than 1.

Exceptions

Table 298-44 SUBSTR Function Exceptions

Error	Description
`VALUE_ERROR`	`VALUE_ERROR` is returned if: • pos = 0 or > length of `r` • len < 1 or > length of `r` - (pos-1)

TRANSLATE Function

This function translates the bytes in the input `RAW r` according to the bytes in the translation `RAWs from_set` and `to_set`.

If a byte in `r` has a matching byte in `from_set`, then it is replaced by the byte in the corresponding position in `to_set`, or deleted.

Bytes in r, but undefined in from_set, are copied to the result. Only the first (leftmost) occurrence of a byte in from_set is used. Subsequent duplicates are not scanned and are ignored.

Syntax

```
UTL_RAW.TRANSLATE (
   r         IN RAW,
   from_set  IN RAW,
   to_set    IN RAW)
  RETURN RAW;
```

Note:

Be aware that *to_set* and *from_set* are reversed in the calling sequence compared to TRANSLITERATE.

Pragmas

```
pragma restrict_references(translate, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-45 TRANSLATE Function Parameters

Parameter	Description
r	RAW source byte-string to be translated
from_set	RAW byte-codes to be translated, if present in r
to_set	RAW byte-codes to which corresponding from_str bytes are translated

Return Values

Table 298-46 TRANSLATE Function Return Values

Return	Description
RAW	Translated byte-string

Usage Notes

- If to_set is shorter than from_set, the extra from_set bytes have no corresponding translation bytes. Bytes from the input RAW that match any such from_set bytes are not translated or included in the result. They are effectively translated to NULL.
- If to_set is longer than from_set, the extra to_set bytes are ignored.
- If a byte value is repeated in from_set, the repeated occurrence is ignored.

Note:

Differences from the TRANSLITERATE Function:

- The from_set parameter comes before the to_set parameter in the calling sequence.
- Bytes from r that appear in from_set but have no corresponding values in to_set are not translated or included in the result.
- The resulting RAW value may be shorter than the input RAW value.

Note that TRANSLATE and TRANSLITERATE only differ in functionality when to_set has fewer bytes than from_set.

Exceptions

Table 298-47 TRANSLATE Function Exceptions

Error	Description
VALUE_ERROR	Either: - r is NULL or has 0 length - from_set is NULL or has 0 length - to_set is NULL or has 0 length

TRANSLITERATE Function

This function converts the bytes in the input RAW r according to the bytes in the transliteration RAWs from_set and to_set.

Successive bytes in r are looked up in the from_set, and, if not found, copied unaltered to the result RAW. If found, then they are replaced in the result RAW by either corresponding bytes in the to_set, or the pad byte when no correspondence exists.

Bytes in r, but undefined in from_set, are copied to the result. Only the first (leftmost) occurrence of a byte in from_set is used. Subsequent duplicates are not scanned and are ignored. The result RAW is always the same length as r.

Syntax

```
UTL_RAW.TRANSLITERATE (
   r         IN RAW,
   to_set    IN RAW DEFAULT NULL,
   from_set  IN RAW DEFAULT NULL,
   pad       IN RAW DEFAULT NULL)
  RETURN RAW;
```

Note:

Be aware that *to_set* and *from_set* are reversed in the calling sequence compared to TRANSLATE.

Pragmas

```
pragma restrict_references(transliterate, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-48 TRANSLITERATE Function Parameters

Parameter	Description
`r`	`RAW` input byte-string to be converted
`to_set`	`RAW` byte-codes to which corresponding `from_set` bytes are converted (any length)
`from_set`	`RAW` byte-codes to be converted, if presenting `r` (any length)
`pad`	1 byte used when to-set is shorter than the `from_set`

Defaults and Optional Parameters

Table 298-49 TRANSLITERATE Function Optional Parameters

Optional Parameter	Description
`to_set`	To the `NULL` string and effectively extended with `pad` to the length of `from_set` as necessary
`from_set`	`x'00'` through `x'fff'`
`pad`	`x'00'`

Return Values

Table 298-50 TRANSLITERATE Function Return Values

Return	Description
`RAW`	Converted byte-string.

Usage Notes

- If `to_set` is shorter than `from_set`, the extra `from_set` bytes have no corresponding conversion bytes. Bytes from the input `RAW` that match any such `from_set` bytes are converted in the result to the pad byte instead.
- If `to_set` is longer than `from_set`, the extra `to_set` bytes are ignored.
- If a byte value is repeated in `from_set`, the repeated occurrence is ignored.

Note:

Differences from the TRANSLATE Function:

- The to_set parameter comes before the from_set parameter in the calling sequence.
- Bytes from r that appear in from_set but have no corresponding values in to_set are replaced by pad in the result.
- The resulting RAW value always has the same length as the input RAW value.

Note that TRANSLATE and TRANSLITERATE only differ in functionality when to_set has fewer bytes than from_set.

Exceptions

Table 298-51 TRANSLITERATE Function Exceptions

Error	Description
VALUE_ERROR	R is NULL or has 0 length

XRANGE Function

This function returns a RAW value containing the succession of one-byte encodings beginning and ending with the specified byte-codes. The specified byte-codes must be single-byte RAW values. If the start_byte value is greater than the end_byte value, then the succession of resulting bytes begins with start_byte, wraps through x'FF' back to x'00', then ends at end_byte.

Syntax

```
UTL_RAW.XRANGE (
   start_byte IN RAW DEFAULT NULL,
   end_byte   IN RAW DEFAULT NULL)
  RETURN RAW;
```

Pragmas

```
pragma restrict_references(xrange, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 298-52 XRANGE Function Parameters

Parameters	Description
start_byte	Beginning byte-code value of resulting sequence. The default is x'00'.
end_byte	Ending byte-code value of resulting sequence. The default is x'FF'.

Return Values

Table 298-53 XRANGE Function Return Values

Return	Description
`RAW`	Containing succession of 1-byte hexadecimal encodings