303 UTL_TCP

With the `UTL_TCP` package and its procedures and functions, PL/SQL applications can communicate with external TCP/IP-based servers using TCP/IP. Because many Internet application protocols are based on TCP/IP, this package is useful to PL/SQL applications that use Internet protocols and e-mail.

This chapter contains the following topics:

UTL_TCP Overview

The `UTL_TCP` package provides TCP/IP client-side access functionality in PL/SQL.

UTL_TCP Security Model

This package is an invoker's rights package and the invoking user needs the connect privilege granted in the access control list assigned to the remote network host to which he wants to connect.

Note:

For more information about managing fine-grained access, see *Oracle Database Security Guide*

UTL_TCP Types

The `UTL_TCP` package includes a CONNECTION type and a carriage-return line-feed (CRLF) type.

CONNECTION Type

This is a PL/SQL record type used to represent a TCP/IP connection.

Syntax

```
TYPE connection IS RECORD (
    remote_host    VARCHAR2(255),
    remote_port    PLS_INTEGER,
    local_host     VARCHAR2(255),
    local_port     PLS_INTEGER,
    charset        VARCHAR2(30),
    newline        VARCHAR2(2),
    tx_timeout     PLS_INTEGER,
    private_sd     PLS_INTEGER);
```

Fields

Table 303-1 Connection Record Type Fields

Field	Description
`remote_host`	Name of the remote host when connection is established. `NULL` when no connection is established.
`remote_port`	Port number of the remote host connected. `NULL` when no connection is established.
`local_host`	Name of the local host used to establish the connection. `NULL` when no connection is established.
`local_port`	Port number of the local host used to establish the connection. `NULL` when no connection is established.
`charset`	The on-the-wire character set. Since text messages in the database may be encoded in a character set that is different from the one expected on the wire (that is, the character set specified by the communication protocol, or the one stipulated by the other end of the communication), text messages in the database are converted to and from the on-the-wire character set as they are sent and received on the network.
`newline`	Newline character sequence. This newline character sequence is appended to the text line sent by `WRITE_LINE` API.
`tx_timeout`	Time in seconds that the UTL_TCP package waits before giving up in a read or write operation in this connection. In read operations, this package gives up if no data is available for reading immediately. In write operations, this package gives up if the output buffer is full and no data is to be sent in the network without being blocked. Zero (0) indicates not to wait at all. `NULL` indicates to wait forever.

Usage Notes

The fields in a connection record are used to return information about the connection, which is often made using `OPEN_CONNECTION`. Changing the values of those fields has no effect on the connection. The fields `private_XXXX` are for implementation use only. You should not modify the values.

In the current release of the `UTL_TCP` package, the parameters `local_host` and `local_port` are ignored when `open_connection` makes a TCP/IP connection. It does not attempt to use the specified local host and port number when the connection is made. The `local_host` and `local_port` fields are not set in the connection record returned by the function.

Time out on write operations is not supported in the current release of the `UTL_TCP` package.

CRLF

The character sequence carriage-return line-feed. It is the newline sequence commonly used by many communication standards.

Syntax

```
CRLF CONSTANT VARCHAR2(2 CHAR);
```

Usage Notes

This package variable defines the newline character sequence commonly used in many Internet protocols. This is the default value of the newline character sequence for `WRITE_LINE`, specified when a connection is opened. While such protocols use `<CR><LF>` to denote a new line, some implementations may choose to use just line-feed to denote a new line. In such cases, users can specify a different newline character sequence when a connection is opened.

UTL_TCP Exceptions

`UTL_TCP` will raise an exception when it encounters a processing issue.

The exceptions raised by the TCP/IP package are listed in the following table.

Table 303-2 ***TCP/IP Exceptions***

Exception	Description
`BUFFER_TOO_SMALL`	Buffer is too small for input that requires look-ahead
`END_OF_INPUT`	Raised when no more data is available to read from the connection
`NETWORK_ERROR`	Generic network error
`BAD_ARGUMENT`	Bad argument passed in an API call (for example, a negative buffer size)
`TRANSFER_TIMEOUT`	No data is read and a read time out occurred
`PARTIAL_MULTIBYTE_CHAR`	No complete character is read and a partial multibyte character is found at the end of the input

UTL_TCP Rules and Limits

The interface provided in the package only allows connections to be initiated by the PL/SQL program. It does not allow the PL/SQL program to accept connections initiated outside the program.

UTL_TCP Examples

Some possible uses for `UTL_TCP` include retrieving a Web page over HTTP or sending an e-mail.

The following code example illustrates how the TCP/IP package can be used to retrieve a Web page over HTTP. It connects to a Web server listening at port 80 (standard port for HTTP) and requests the root document.

```
DECLARE
  c  utl_tcp.connection;  -- TCP/IP connection to the Web server
  ret_val pls_integer;
BEGIN
  c := utl_tcp.open_connection(remote_host => 'www.acme.com',
                               remote_port =>  80,
                               charset     => 'US7ASCII');  -- open connection
  ret_val := utl_tcp.write_line(c, 'GET / HTTP/1.0');    -- send HTTP request
  ret_val := utl_tcp.write_line(c);
  BEGIN
    LOOP
      dbms_output.put_line(utl_tcp.get_line(c, TRUE));  -- read result
    END LOOP;
  EXCEPTION
    WHEN utl_tcp.end_of_input THEN
      NULL; -- end of input
  END;
  utl_tcp.close_connection(c);
END;
```

The following code example illustrates how the TCP/IP package can be used by an application to send e-mail (also known as email from PL/SQL). The application connects to an SMTP server at port 25 and sends a simple text message.

```
PROCEDURE send_mail (sender    IN VARCHAR2,
                     recipient IN VARCHAR2,
                   message   IN VARCHAR2) IS
    mailhost   VARCHAR2(30) := 'mailhost.mydomain.com';
    smtp_error  EXCEPTION;
    mail_conn   utl_tcp.connection;
    PROCEDURE smtp_command(command IN VARCHAR2,
                          ok      IN VARCHAR2 DEFAULT '250')
    IS
        response varchar2(3);
        len pls_integer;
    BEGIN
        len := utl_tcp.write_line(mail_conn, command);
        response := substr(utl_tcp.get_line(mail_conn), 1, 3);
        IF (response <> ok) THEN
            RAISE smtp_error;
        END IF;
    END;

BEGIN
    mail_conn := utl_tcp.open_connection(remote_host => mailhost,
                                         remote_port => 25,
                                         charset     => 'US7ASCII');
    smtp_command('HELO ' || mailhost);
    smtp_command('MAIL FROM: ' || sender);
    smtp_command('RCPT TO: ' || recipient);
    smtp_command('DATA', '354');
    smtp_command(message);
    smtp_command('QUIT', '221');
    utl_tcp.close_connection(mail_conn);
EXCEPTION
    WHEN OTHERS THEN
        -- Handle the error
END;
```

Summary of UTL_TCP Subprograms

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

Table 303-3 UTL_TCP Package Subprograms

Subprogram	Description
AVAILABLE Function	Determines the number of bytes available for reading from a TCP/IP connection
CLOSE_ALL_CONNECTIONS Procedure	Closes all open TCP/IP connections
CLOSE_CONNECTION Procedure	Closes an open TCP/IP connection
FLUSH Procedure	Transmits immediately to the server all data in the output buffer, if a buffer is used
GET_LINE Function	Returns the line of data read
GET_LINE_NCHAR Function	Returns the line of data read in NCHAR form
GET_RAW Function	Return the data read instead of the amount of data read
GET_TEXT Function	Returns the text data read
GET_TEXT_NCHAR Function	Returns the text data read in NCHAR form
OPEN_CONNECTION Function	Opens a TCP/IP connection to a specified service
READ_LINE Function	Receives a text line from a service on an open connection
READ_RAW Function	Receives binary data from a service on an open connection
READ_TEXT Function	Receives text data from a service on an open connection
SECURE_CONNECTION Procedure	Secures a TCP/IP connection using SSL/TLS
WRITE_LINE Function	Transmits a text line to a service on an open connection
WRITE_RAW Function	Transmits a binary message to a service on an open connection
WRITE_TEXT Function	Transmits a text message to a service on an open connection

AVAILABLE Function

This function determines the number of bytes available for reading from a TCP/IP connection. It is the number of bytes that can be read immediately without blocking. Determines if data is ready to be read from the connection.

Syntax

```
UTL_TCP.AVAILABLE (
   c        IN OUT NOCOPY connection,
   timeout  IN PLS_INTEGER DEFAULT 0)
RETURN PLS_INTEGER;
```

Parameters

Table 303-4 AVAILABLE Function Parameters

Parameter	Description
`c`	TCP connection to determine the amount of data that is available to be read

Table 303-4 (Cont.) AVAILABLE Function Parameters

Parameter	Description
`timeout`	Time in seconds to wait before giving up and reporting that no data is available. Zero (0) indicates not to wait at all. `NULL` indicates to wait forever.

Return Values

The number of bytes available for reading without blocking

Usage Notes

The connection must have already been opened through a call to `OPEN_CONNECTION`. Users may use this API to determine if data is available to be read before calling the read API so that the program are not blocked because data is not ready to be read from the input.

The number of bytes available for reading returned by this function may be less than what is actually available. On some platforms, this function may only return 1, to indicate that some data is available. If you are concerned about the portability of your application, then assume that this function returns a positive value when data is available for reading, and 0 when no data is available. This function returns a positive value when all the data at a particular connection has been read and the next read result in the `END_OF_INPUT` exception.

The following example illustrates using this function in a portable manner:

```
DECLARE
   c    utl_tcp.connection
   data VARCHAR2(256);
   len  PLS_INTEGER;
BEGIN
   c := utl_tcp.open_connection(...);
   LOOP
      IF (utl_tcp.available(c) > 0) THEN
         len := utl_tcp.read_text(c, data, 256);
      ELSE
         ---do some other things
        . . . .
      END IF
   END LOOP;
END;
```

CLOSE_ALL_CONNECTIONS Procedure

This procedure closes all open TCP/IP connections.

Syntax

```
UTL_TCP.CLOSE_ALL_CONNECTIONS;
```

Usage Notes

This call is provided to close all connections before a PL/SQL program ends to avoid dangling connections.

CLOSE_CONNECTION Procedure

This procedure closes an open TCP/IP connection.

Syntax

```
UTL_TCP.CLOSE_CONNECTION (
   c IN OUT NOCOPY connection);
```

Parameters

Table 303-5 CLOSE_CONNECTION Procedure Parameters

Parameter	Description
`c`	TCP connection to close

Usage Notes

Connection must have been opened by a previous call to `OPEN_CONNECTION`. The fields `remote_host, remote_port, local_host, local_port` and `charset` of `c` are reset after the connection is closed.

An open connection must be closed explicitly. An open connection remains open when the PL/SQL record variable that stores the connection goes out-of-scope in the PL/SQL program. Failing to close unwanted connections may result in unnecessary tying up of local and remote system resources.

FLUSH Procedure

This procedure transfers immediately to the server all data in the output buffer, if a buffer is used.

Syntax

```
UTL_TCP.FLUSH (
   c IN OUT NOCOPY connection);
```

Parameters

Table 303-6 FLUSH Procedure Parameters

Parameter	Description
`c`	TCP connection to which to send data

Usage Notes

The connection must have already been opened through a call to `OPEN_CONNECTION`.

GET_LINE Function

This function returns the line of data read.

Syntax

```
UTL_TCP.GET_LINE (
   c           IN OUT NOCOPY connection,
   remove_crlf IN            BOOLEAN DEFAULT FALSE,
   peek        IN            BOOLEAN DEFAULT FALSE)
 RETURN VARCHAR2;
```

Parameters

Table 303-7 GET_LINE Function Parameters

Parameter	Description
`c`	TCP connection from which to receive data
`remove_crlf`	If `TRUE`, then one ore more trailing CRLF characters are removed from the received message.
`peek`	Normally, you want to read the data and remove it from the input queue, that is, consume it. In some situations, you may just want to look ahead at the data, that is, peek at it, without removing it from the input queue, so that it is still available for reading (or even peeking) in the next call. To keep the data in the input queue, set this flag to `TRUE` and set up an input buffer before the connection is opened. The amount of data you can peek at (that is, read but keep in the input queue) must be less than the size of input buffer.

Return Values

The text line read

Usage Notes

- The connection must have already been opened through a call to `OPEN_CONNECTION`.
- See `READ_LINE` for the read time out, character set conversion, buffer size, and multibyte character issues.

GET_LINE_NCHAR Function

This function returns the line of data read in NCHAR form.

Syntax

```
UTL_TCP.GET_LINE_NCHAR (
   c           IN OUT NOCOPY connection,
   remove_crlf IN            BOOLEAN DEFAULT FALSE,
   peek        IN            BOOLEAN DEFAULT FALSE)
 RETURN NVARCHAR2;
```

Parameters

Table 303-8 GET_LINE_NCHAR Function Parameters

Parameter	Description
`c`	TCP connection from which to receive data
`remove_crlf`	If `TRUE`, then one ore more trailing CRLF characters are removed from the received message.
`peek`	Normally, you want to read the data and remove it from the input queue, that is, consume it. In some situations, you may just want to look ahead at the data, that is, peek at it, without removing it from the input queue, so that it is still available for reading (or even peeking) in the next call. To keep the data in the input queue, set this flag to `TRUE` and set up an input buffer before the connection is opened. The amount of data you can peek at (that is, read but keep in the input queue) must be less than the size of input buffer.

Return Values

The text line read

Usage Notes

- The connection must have already been opened through a call to `OPEN_CONNECTION`.
- See `READ_LINE` for the read time out, character set conversion, buffer size, and multibyte character issues.

GET_RAW Function

This function returns the data read instead of the amount of data read.

Syntax

```
UTL_TCP.GET_RAW (
   c      IN OUT NOCOPY connection,
   len    IN            PLS_INTEGER DEFAULT 1,
   peek   IN            BOOLEAN     DEFAULT FALSE)
 RETURN RAW;
```

Parameters

Table 303-9 GET_RAW Function Parameters

Parameter	Description
`c`	TCP connection from which to receive data
`len`	The number of bytes (or characters for `VARCHAR2`) of data to receive. Default is 1.

Table 303-9 (Cont.) GET_RAW Function Parameters

Parameter	Description
`peek`	Normally, you want to read the data and remove it from the input queue, that is, consume it. In some situations, you may just want to look ahead at the data, that is, peek at it, without removing it from the input queue, so that it is still available for reading (or even peeking) in the next call. To keep the data in the input queue, set this flag to `TRUE` and set up an input buffer before the connection is opened. The amount of data you can peek at (that is, read but keep in the input queue) must be less than the size of input buffer.
`remove_crlf`	If `TRUE`, then one ore more trailing CRLF characters are removed from the received message.

Return Values

The binary data read

Usage Notes

The connection must have already been opened through a call to `OPEN_CONNECTION`.

For all the `get_*` APIs described in this section, see the corresponding `READ_*` API for the read time out issue. For `GET_TEXT` and `GET_LINE`, see the corresponding `READ_*` API for character set conversion, buffer size, and multibyte character issues.

GET_TEXT Function

This function returns the text data read.

Syntax

```
UTL_TCP.GET_TEXT (
   c    IN OUT NOCOPY connection,
   len  IN            PLS_INTEGER DEFAULT 1,
   peek IN            BOOLEAN     DEFAULT FALSE)
 RETURN VARCHAR2;
```

Parameters

Table 303-10 GET_TEXT Function Parameters

Parameter	Description
`c`	TCP connection from which to receive data
`len`	Number of bytes (or characters for `VARCHAR2`) of data to receive. Default is 1.
`peek`	Normally, you want to read the data and remove it from the input queue, that is, consume it. In some situations, you may just want to look ahead at the data, that is, peek at it, without removing it from the input queue, so that it is still available for reading (or even peeking) in the next call. To keep the data in the input queue, set this flag to `TRUE` and set up an input buffer before the connection is opened. The amount of data you can peek at (that is, read but keep in the input queue) must be less than the size of input buffer.

Table 303-10 (Cont.) GET_TEXT Function Parameters

Parameter	Description
`remove_crlf`	If `TRUE`, then one ore more trailing CRLF characters are removed from the received message.

Return Values

The text data read

Usage Notes

The connection must have already been opened through a call to `OPEN_CONNECTION`.

For all the `get_*` APIs described in this section, see the corresponding `read_*` API for the read time out issue. For `GET_TEXT` and `GET_LINE`, see the corresponding `READ_*` API for character set conversion, buffer size, and multibyte character issues.

GET_TEXT_NCHAR Function

This function returns the text data read in NCHAR form.

Syntax

```
UTL_TCP.GET_TEXT_NCHAR (
   c    IN OUT NOCOPY connection,
   len  IN            PLS_INTEGER DEFAULT 1,
   peek IN            BOOLEAN     DEFAULT FALSE)
 RETURN NVARCHAR2;
```

Parameters

Table 303-11 GET_TEXT_NCHAR Function Parameters

Parameter	Description
`c`	TCP connection from which to receive data
`len`	The number of bytes (or characters for `VARCHAR2`) of data to receive. Default is 1.
`peek`	Normally, you want to read the data and remove it from the input queue, that is, consume it. In some situations, you may just want to look ahead at the data, that is, peek at it, without removing it from the input queue, so that it is still available for reading (or even peeking) in the next call. To keep the data in the input queue, set this flag to `TRUE` and set up an input buffer before the connection is opened. The amount of data you can peek at (that is, read but keep in the input queue) must be less than the size of input buffer.
`remove_crlf`	If `TRUE`, then one ore more trailing CRLF characters are removed from the received message.

Return Values

The text data read

Usage Notes

The connection must have already been opened through a call to `OPEN_CONNECTION`.

For all the `get_*` APIs described in this section, see the corresponding `read_*` API for the read time out issue. For `GET_TEXT` and `GET_LINE`, see the corresponding `READ_*` API for character set conversion, buffer size, and multibyte character issues.

OPEN_CONNECTION Function

This function opens a TCP/IP connection to a specified service.

Syntax

```
UTL_TCP.OPEN_CONNECTION  (
   remote_host          IN VARCHAR2,
   remote_port          IN PLS_INTEGER,
   local_host           IN VARCHAR2 DEFAULT NULL,
   local_port           IN PLS_INTEGER DEFAULT NULL,
   in_buffer_size       IN PLS_INTEGER DEFAULT NULL,
   out_buffer_size      IN PLS_INTEGER DEFAULT NULL,
   charset              IN VARCHAR2 DEFAULT NULL,
   newline              IN VARCHAR2 DEFAULT CRLF,
   tx_timeout           IN PLS_INTEGER DEFAULT NULL,
   wallet_path          IN  VARCHAR2 DEFAULT NULL,
   wallet_password      IN  VARCHAR2 DEFAULT NULL,
  RETURN connection;
```

Parameters

Table 303-12   OPEN_CONNECTION Function Parameters

Parameter	Description
`remote_host`	Name of the host providing the service. When `remote_host` is `NULL`, it connects to the local host.
`remote_port`	Port number on which the service is listening for connections
`local_host`	Name of the host providing the service. `NULL` means does not care.
`local_port`	Port number on which the service is listening for connections. `NULL` means don't care.
`in_buffer_size`	The size of input buffer. The use of an input buffer can speed up execution performance in receiving data from the server. The appropriate size of the buffer depends on the flow of data between the client and the server, and the traffic/latency on the network. A zero value means no buffer should be used. A `NULL` value means the caller does not care if a buffer is used or not. The maximum size of the input buffer is 32767 bytes.
`out_buffer_size`	The size of output buffer. The use of an output buffer can speed up execution performance in sending data to the server. The appropriate size of buffer depends on the flow of data between the client and the server, and the network condition. A zero value means no buffer should be used. A `NULL` value means the caller does not care if a buffer is used or not. The maximum size of the output buffer is 32767 bytes.

Table 303-12 (Cont.) OPEN_CONNECTION Function Parameters

Parameter	Description
`charset`	The on-the-wire character set. Since text messages in the database may be encoded in a character set that is different from the one expected on the wire (that is, the character set specified by the communication protocol, or the one stipulated by the other end of the communication), text messages in the database are converted to and from the on-the-wire character set as they are sent and received on the network using `READ_TEXT`, `READ_LINE`, `WRITE_TEXT` and `WRITE_LINE`. Set this parameter to `NULL` when no conversion is needed.
`newline`	Newline character sequence. This newline character sequence is appended to the text line sent by `WRITE_LINE` API.
`tx_timeout`	Time in seconds that the `UTL_TCP` package should wait before giving up in a read or write operations in this connection. In read operations, this package gives up if no data is available for reading immediately. In write operations, this package gives up if the output buffer is full and no data is to be sent in the network without being blocked. Zero (0) indicates not to wait at all. `NULL` indicates to wait forever.
`wallet_path`	Directory path that contains the Oracle wallet for SSL/TLS. The format is `file:`*`directory-path`*. If you want to use the operating system certificate store to act in place of the Oracle wallet, then set the `path` parameter to `system:` (include the colon). Doing so greatly improves performance in the database.
`wallet_password`	Password to open the wallet. When the wallet is auto-login enabled, the password may be set to `NULL`. If you set `path` to `system:`, then omit the password by setting it to `NULL`.

Return Values

A connection to the targeted TCP/IP service

Usage Notes

- Note that connections opened by this `UTL_TCP` package can remain open and be passed from one database call to another in a shared server configuration. However, the connection must be closed explicitly. The connection remains open when the PL/SQL record variable that stores the connection goes out-of-scope in the PL/SQL program. Failing to close unwanted connections may result in unnecessary tying up of local and remote system resources.
- In the current release of the `UTL_TCP` package, the parameters `local_host` and `local_port` are ignored when `open_connection` makes a TCP/IP connection. It does not attempt to use the specified local host and port number when the connection is made. The `local_host` and `local_port` fields is not set in the connection record returned by the function.
- `tx_timeout` is intended to govern both the read operations and the write operations. However, an implementation restriction prevents `tx_timeout` from governing write operations in the current release.

Examples

```
DECLARE
  c UTL_TCP.CONNECTION;
BEGIN
  c := UTL_TCP.OPEN_CONNECTION(
   host            => 'www.example.com',
   port            => 443,
   wallet_path     => 'file:/oracle/wallets/smtp_wallet',
   wallet_password => '****');
  UTL_TCP.SECURE_CONNECTION (c => c);
END;
```

READ_LINE Function

This function receives a text line from a service on an open connection.

A line is terminated by a line-feed, a carriage-return or a carriage-return followed by a line-feed.

Syntax

```
UTL_TCP.READ_LINE (
   c           IN OUT NOCOPY connection,
   data        IN OUT NOCOPY VARCHAR2 CHARACTER SET ANY_CS,
   peek        IN            BOOLEAN DEFAULT FALSE)
 RETURN PLS_INTEGER;
```

Parameters

Table 303-13 READ_LINE Function Parameters

Parameter	Description
`c`	TCP connection from which to receive data
`data`	Data received.
`remove_crlf`	If `TRUE`, then one ore more trailing CRLF characters are removed from the received message.
`peek`	Normally, you want to read the data and remove it from the input queue, that is, consume it. In some situations, you may just want to look ahead at the data, that is, peek at it, without removing it from the input queue, so that it is still available for reading (or even peeking) in the next call. To keep the data in the input queue, set this flag to `TRUE` and set up an input buffer before the connection is opened. The amount of data you can peek at (that is, read but keep in the input queue) must be less than the size of input buffer.

Return Values

The number of characters of data received

Usage Notes

The connection must have already been opened through a call to `OPEN_CONNECTION`. This function does not return until the end-of-line have been reached, or the end of input has been reached. Text messages is converted from the on-the-wire character set, specified when the connection was opened, to the database character set before they are returned to the caller.

If transfer time out is set when the connection is opened, then this function waits for each data packet to be ready to read until time out occurs. If it occurs, then this function stops reading and returns all the data read successfully. If no data is read successfully, then the `transfer_timeout` exception is raised. The exception can be handled and the read operation can be retried later.

If a partial multibyte character is found at the end of input, then this function stops reading and returns all the complete multibyte characters read successfully. If no complete character is read successfully, then the `partial_multibyte_char` exception is raised. The exception can be handled and the bytes of that partial multibyte character can be read as binary by the `READ_RAW` function. If a partial multibyte character is seen in the middle of the input because the remaining bytes of the character have not arrived and read time out occurs, then the `transfer_timeout` exception is raised instead. The exception can be handled and the read operation can be retried later.

READ_RAW Function

This function receives binary data from a service on an open connection.

Syntax

```
UTL_TCP.READ_RAW (
   c     IN OUT NOCOPY connection,
   data  IN OUT NOCOPY RAW,
   len   IN            PLS_INTEGER DEFAULT 1,
   peek  IN            BOOLEAN     DEFAULT FALSE)
 RETURN PLS_INTEGER;
```

Parameters

Table 303-14 READ_RAW Function Parameters

Parameter	Description
`c`	TCP connection from which to receive data
`data (IN OUT COPY)`	Data received
`len`	Number of bytes of data to receive
`peek`	Normally, you want to read the data and remove it from the input queue, that is, consume it. In some situations, you may just want to look ahead at the data, that is, peek at it, without removing it from the input queue, so that it is still available for reading (or even peeking) in the next call. To keep the data in the input queue, set this flag to `TRUE` and set up an input buffer before the connection is opened. The amount of data you can peek at (that is, read but keep in the input queue) must be less than the size of input buffer.

Return Values

The number of bytes of data received

Usage Notes

The connection must have already been opened through a call to `OPEN_CONNECTION`. This function does not return until the specified number of bytes have been read, or the end of input has been reached.

If transfer time out is set when the connection is opened, then this function waits for each data packet to be ready to read until time out occurs. If it occurs, then this function stops reading and returns all the data read successfully. If no data is read successfully, then the `transfer_timeout` exception is raised. The exception can be handled and the read operation can be retried later.

READ_TEXT Function

This function receives text data from a service on an open connection.

Syntax

```
UTL_TCP.READ_TEXT (
   c    IN OUT NOCOPY connection,
   data IN OUT NOCOPY VARCHAR2 CHARACTER SET ANY_CS,
   len  IN            PLS_INTEGER DEFAULT 1,
   peek IN            BOOLEAN     DEFAULT FALSE)
RETURN PLS_INTEGER;
```

Parameters

Table 303-15 READ_TEXT Function Parameters

Parameter	Description
`c`	TCP connection from which to receive data
`data`	Data received
`len`	Number of characters of data to receive
`peek`	Normally, users want to read the data and remove it from the input queue, that is, consume it. In some situations, users may just want to look ahead at the data without removing it from the input queue so that it is still available for reading (or even peeking) in the next call. To keep the data in the input queue, set this flag to `TRUE` and an input buffer must be set up when the connection is opened. The amount of data that you can peek at (that is, read but keep in the input queue) must be less than the size of input buffer.

Return Values

The number of characters of data received

Usage Notes

The connection must have already been opened through a call to `OPEN_CONNECTION`. This function does not return until the specified number of characters has been read, or the end of input has been reached. Text messages is converted from the on-the-wire character set, specified when the connection was opened, to the database character set before they are returned to the caller.

Unless explicitly overridden, the size of a `VARCHAR2` buffer is specified in terms of bytes, while the parameter `len` refers to the maximum number of characters to be read. When the database character set is multibyte, where a single character may consist of more than 1 byte, you should ensure that the buffer can hold the maximum of characters. In general, the size of the `VARCHAR2` buffer should equal the number of characters to be read, multiplied by the maximum number of bytes of a character of the database character set.

If transfer time out is set when the connection is opened, then this function waits for each data packet to be ready to read until time out occurs. If it occurs, then this function stops reading and returns all the data read successfully. If no data is read successfully, then the `transfer_timeout` exception is raised. The exception can be handled and the read operation can be retried later.

If a partial multibyte character is found at the end of input, then this function stops reading and returns all the complete multibyte characters read successfully. If no complete character is read successfully, then the `partial_multibyte_char` exception is raised. The exception can be handled and the bytes of that partial multibyte character can be read as binary by the `READ_RAW` function. If a partial multibyte character is seen in the middle of the input because the remaining bytes of the character have not arrived and read time out occurs, then the `transfer_timeout` exception is raised instead. The exception can be handled and the read operation can be retried later.

SECURE_CONNECTION Procedure

This procedure secures a TCP/IP connection using SSL/TLS.

SSL/TLS requires an Oracle wallet which must be specified when the connection was opened by the OPEN_CONNECTION Function.

Syntax

```
UTL_TCP.SECURE_CONNECTION (
   c     IN OUT NOCOPY connection);
```

Parameters

Table 303-16 SECURE_CONNECTION Procedure Parameters

Parameter	Description
`c`	TCP connection from which to receive data

WRITE_LINE Function

This function transmits a text line to a service on an open connection. The `newline` character sequence is appended to the message before it is transmitted.

Syntax

```
UTL_TCP.WRITE_LINE (
   c     IN OUT NOCOPY connection,
   data  IN            VARCHAR2 DEFAULT NULL CHARACTER SET ANY_CS)
  RETURN PLS_INTEGER;
```

Parameters

Table 303-17 WRITE_LINE Function Parameters

Parameter	Description
`c`	TCP connection to which to send data
`data`	Buffer containing the data to be sent

Return Values

The actual number of characters of data transmitted

Usage Notes

The connection must have already been opened through a call to `OPEN_CONNECTION`. Text messages are converted to the on-the-wire character set, specified when the connection was opened, before they are transmitted on the wire.

WRITE_RAW Function

This function transmits a binary message to a service on an open connection. The function does not return until the specified number of bytes have been written.

Syntax

```
UTL_TCP.WRITE_RAW (
   c    IN OUT NOCOPY connection,
   data IN            RAW,
   len  IN            PLS_INTEGER DEFAULT NULL)
 RETURN PLS_INTEGER;
```

Parameters

Table 303-18   WRITE_RAW Function Parameters

Parameter	Description
`c`	TCP connection to which to send data
`data`	Buffer containing the data to be sent
`len`	The number of bytes of data to transmit. When `len` is `NULL`, the whole length of data is written.

Return Values

The number of bytes of data transmitted

Usage Notes

The connection must have already been opened through a call to `OPEN_CONNECTION`.

WRITE_TEXT Function

This function transmits a text message to a service on an open connection.

Syntax

```
UTL_TCP.WRITE_TEXT (
   c    IN OUT NOCOPY connection,
   data IN            VARCHAR2 CHARACTER SET ANY_CS,
   len  IN            PLS_INTEGER DEFAULT NULL)
 RETURN num_chars PLS_INTEGER;
```

Parameters

Table 303-19 WRITE_TEXT Function Parameters

Parameter	Description
`c`	TCP connection to which to send data
`data`	Buffer containing the data to be sent
`len`	The number of characters of data to transmit. When `len` is `NULL`, the whole length of data is written. The actual amount of data written may be less because of network condition.

Return Values

The actual number of characters of data transmitted

Usage Notes

The connection must have already been opened through a call to `OPEN_CONNECTION`. Text messages are converted to the on-the-wire character set, specified when the connection was opened, before they are transmitted on the wire.