https://oracle-base.com/articles/misc/email-from-oracle-plsql

Simple Emails

In it's simplest form a single string or variable can be sent as the message body using the following procedure. In this case we have not included any header information or subject line in the message, so it is not very useful, but it is small.

CREATE OR REPLACE PROCEDURE send\_mail (p\_to IN VARCHAR2,

p\_from IN VARCHAR2,

p\_message IN VARCHAR2,

p\_smtp\_host IN VARCHAR2,

p\_smtp\_port IN NUMBER DEFAULT 25)

AS

l\_mail\_conn UTL\_SMTP.connection;

BEGIN

l\_mail\_conn := UTL\_SMTP.open\_connection(p\_smtp\_host, p\_smtp\_port);

UTL\_SMTP.helo(l\_mail\_conn, p\_smtp\_host);

UTL\_SMTP.mail(l\_mail\_conn, p\_from);

UTL\_SMTP.rcpt(l\_mail\_conn, p\_to);

UTL\_SMTP.data(l\_mail\_conn, p\_message || UTL\_TCP.crlf || UTL\_TCP.crlf);

UTL\_SMTP.quit(l\_mail\_conn);

END;

/

The code below shows how the procedure is called.

BEGIN

send\_mail(p\_to => 'me@mycompany.com',

p\_from => 'admin@mycompany.com',

p\_message => 'This is a test message.',

p\_smtp\_host => 'smtp.mycompany.com');

END;

/

Multi-Line Emails

Multi-line messages can be written by expanding the UTL\_SMTP.DATA command using the UTL\_SMTP.WRITE\_DATA command as follows. This is a better method to use as the total message size is no longer constrained by the 32K limit on a VARCHAR2 variable. In the following example the header information has been included in the message also.

CREATE OR REPLACE PROCEDURE send\_mail (p\_to IN VARCHAR2,

p\_from IN VARCHAR2,

p\_subject IN VARCHAR2,

p\_message IN VARCHAR2,

p\_smtp\_host IN VARCHAR2,

p\_smtp\_port IN NUMBER DEFAULT 25)

AS

l\_mail\_conn UTL\_SMTP.connection;

BEGIN

l\_mail\_conn := UTL\_SMTP.open\_connection(p\_smtp\_host, p\_smtp\_port);

UTL\_SMTP.helo(l\_mail\_conn, p\_smtp\_host);

UTL\_SMTP.mail(l\_mail\_conn, p\_from);

UTL\_SMTP.rcpt(l\_mail\_conn, p\_to);

UTL\_SMTP.open\_data(l\_mail\_conn);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Date: ' || TO\_CHAR(SYSDATE, 'DD-MON-YYYY HH24:MI:SS') || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'To: ' || p\_to || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'From: ' || p\_from || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Subject: ' || p\_subject || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Reply-To: ' || p\_from || UTL\_TCP.crlf || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, p\_message || UTL\_TCP.crlf || UTL\_TCP.crlf);

UTL\_SMTP.close\_data(l\_mail\_conn);

UTL\_SMTP.quit(l\_mail\_conn);

END;

/

The code below shows how the procedure is called.

BEGIN

send\_mail(p\_to => 'me@mycompany.com',

p\_from => 'admin@mycompany.com',

p\_subject => 'Test Message',

p\_message => 'This is a test message.',

p\_smtp\_host => 'smtp.mycompany.com');

END;

/

HTML Emails

The following procedure builds on the previous version, allowing it include plain text and/or HTML versions of the email. The format of the message is explained [here](http://www.w3.org/TR/1998/NOTE-HTMLThreading-0105).

CREATE OR REPLACE PROCEDURE send\_mail (p\_to IN VARCHAR2,

p\_from IN VARCHAR2,

p\_subject IN VARCHAR2,

p\_text\_msg IN VARCHAR2 DEFAULT NULL,

p\_html\_msg IN VARCHAR2 DEFAULT NULL,

p\_smtp\_host IN VARCHAR2,

p\_smtp\_port IN NUMBER DEFAULT 25)

AS

l\_mail\_conn UTL\_SMTP.connection;

l\_boundary VARCHAR2(50) := '----=\*#abc1234321cba#\*=';

BEGIN

l\_mail\_conn := UTL\_SMTP.open\_connection(p\_smtp\_host, p\_smtp\_port);

UTL\_SMTP.helo(l\_mail\_conn, p\_smtp\_host);

UTL\_SMTP.mail(l\_mail\_conn, p\_from);

UTL\_SMTP.rcpt(l\_mail\_conn, p\_to);

UTL\_SMTP.open\_data(l\_mail\_conn);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Date: ' || TO\_CHAR(SYSDATE, 'DD-MON-YYYY HH24:MI:SS') || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'To: ' || p\_to || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'From: ' || p\_from || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Subject: ' || p\_subject || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Reply-To: ' || p\_from || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'MIME-Version: 1.0' || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Content-Type: multipart/alternative; boundary="' || l\_boundary || '"' || UTL\_TCP.crlf || UTL\_TCP.crlf);

IF p\_text\_msg IS NOT NULL THEN

UTL\_SMTP.write\_data(l\_mail\_conn, '--' || l\_boundary || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Content-Type: text/plain; charset="iso-8859-1"' || UTL\_TCP.crlf || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, p\_text\_msg);

UTL\_SMTP.write\_data(l\_mail\_conn, UTL\_TCP.crlf || UTL\_TCP.crlf);

END IF;

IF p\_html\_msg IS NOT NULL THEN

UTL\_SMTP.write\_data(l\_mail\_conn, '--' || l\_boundary || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Content-Type: text/html; charset="iso-8859-1"' || UTL\_TCP.crlf || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, p\_html\_msg);

UTL\_SMTP.write\_data(l\_mail\_conn, UTL\_TCP.crlf || UTL\_TCP.crlf);

END IF;

UTL\_SMTP.write\_data(l\_mail\_conn, '--' || l\_boundary || '--' || UTL\_TCP.crlf);

UTL\_SMTP.close\_data(l\_mail\_conn);

UTL\_SMTP.quit(l\_mail\_conn);

END;

/

The code below shows how the procedure is called.

DECLARE

l\_html VARCHAR2(32767);

BEGIN

l\_html := '<html>

<head>

<title>Test HTML message</title>

</head>

<body>

<p>This is a <b>HTML</b> <i>version</i> of the test message.</p>

<p><img src="http://oracle-base.com/images/site\_logo.gif" alt="Site Logo" />

</body>

</html>';

send\_mail(p\_to => 'me@mycompany.com',

p\_from => 'admin@mycompany.com',

p\_subject => 'Test Message',

p\_text\_msg => 'This is a test message.',

p\_html\_msg => l\_html,

p\_smtp\_host => 'smtp.mycompany.com');

END;

/

Emails with Attachments

Sending an email with an attachment is similar to the previous example as the message and the attachment must be separated by a boundary and identified by a name and mime type.

BLOB Attachment

Attaching a BLOB requires the binary data to be encoded and converted to text so it can be sent using SMTP.

CREATE OR REPLACE PROCEDURE send\_mail (p\_to IN VARCHAR2,

p\_from IN VARCHAR2,

p\_subject IN VARCHAR2,

p\_text\_msg IN VARCHAR2 DEFAULT NULL,

p\_attach\_name IN VARCHAR2 DEFAULT NULL,

p\_attach\_mime IN VARCHAR2 DEFAULT NULL,

p\_attach\_blob IN BLOB DEFAULT NULL,

p\_smtp\_host IN VARCHAR2,

p\_smtp\_port IN NUMBER DEFAULT 25)

AS

l\_mail\_conn UTL\_SMTP.connection;

l\_boundary VARCHAR2(50) := '----=\*#abc1234321cba#\*=';

l\_step PLS\_INTEGER := 12000; -- make sure you set a multiple of 3 not higher than 24573

BEGIN

l\_mail\_conn := UTL\_SMTP.open\_connection(p\_smtp\_host, p\_smtp\_port);

UTL\_SMTP.helo(l\_mail\_conn, p\_smtp\_host);

UTL\_SMTP.mail(l\_mail\_conn, p\_from);

UTL\_SMTP.rcpt(l\_mail\_conn, p\_to);

UTL\_SMTP.open\_data(l\_mail\_conn);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Date: ' || TO\_CHAR(SYSDATE, 'DD-MON-YYYY HH24:MI:SS') || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'To: ' || p\_to || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'From: ' || p\_from || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Subject: ' || p\_subject || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Reply-To: ' || p\_from || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'MIME-Version: 1.0' || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Content-Type: multipart/mixed; boundary="' || l\_boundary || '"' || UTL\_TCP.crlf || UTL\_TCP.crlf);

IF p\_text\_msg IS NOT NULL THEN

UTL\_SMTP.write\_data(l\_mail\_conn, '--' || l\_boundary || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Content-Type: text/plain; charset="iso-8859-1"' || UTL\_TCP.crlf || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, p\_text\_msg);

UTL\_SMTP.write\_data(l\_mail\_conn, UTL\_TCP.crlf || UTL\_TCP.crlf);

END IF;

IF p\_attach\_name IS NOT NULL THEN

UTL\_SMTP.write\_data(l\_mail\_conn, '--' || l\_boundary || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Content-Type: ' || p\_attach\_mime || '; name="' || p\_attach\_name || '"' || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Content-Transfer-Encoding: base64' || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Content-Disposition: attachment; filename="' || p\_attach\_name || '"' || UTL\_TCP.crlf || UTL\_TCP.crlf);

FOR i IN 0 .. TRUNC((DBMS\_LOB.getlength(p\_attach\_blob) - 1 )/l\_step) LOOP

UTL\_SMTP.write\_data(l\_mail\_conn, UTL\_RAW.cast\_to\_varchar2(UTL\_ENCODE.base64\_encode(DBMS\_LOB.substr(p\_attach\_blob, l\_step, i \* l\_step + 1))));

END LOOP;

UTL\_SMTP.write\_data(l\_mail\_conn, UTL\_TCP.crlf || UTL\_TCP.crlf);

END IF;

UTL\_SMTP.write\_data(l\_mail\_conn, '--' || l\_boundary || '--' || UTL\_TCP.crlf);

UTL\_SMTP.close\_data(l\_mail\_conn);

UTL\_SMTP.quit(l\_mail\_conn);

END;

/

The code below shows how the procedure is called.

DECLARE

l\_name images.name%TYPE := 'site\_logo.gif';

l\_blob images.image%TYPE;

BEGIN

SELECT image

INTO l\_blob

FROM images

WHERE name = l\_name;

send\_mail(p\_to => 'me@mycompany.com',

p\_from => 'admin@mycompany.com',

p\_subject => 'Test Message',

p\_text\_msg => 'This is a test message.',

p\_attach\_name => 'site\_logo.gif',

p\_attach\_mime => 'image/gif',

p\_attach\_blob => l\_blob,

p\_smtp\_host => 'smtp.mycompany.com');

END;

/

CLOB Attachment

Attaching a CLOB is similar to attaching a BLOB, but we don't have to worry about encoding the data because it is already plain text.

CREATE OR REPLACE PROCEDURE send\_mail (p\_to IN VARCHAR2,

p\_from IN VARCHAR2,

p\_subject IN VARCHAR2,

p\_text\_msg IN VARCHAR2 DEFAULT NULL,

p\_attach\_name IN VARCHAR2 DEFAULT NULL,

p\_attach\_mime IN VARCHAR2 DEFAULT NULL,

p\_attach\_clob IN CLOB DEFAULT NULL,

p\_smtp\_host IN VARCHAR2,

p\_smtp\_port IN NUMBER DEFAULT 25)

AS

l\_mail\_conn UTL\_SMTP.connection;

l\_boundary VARCHAR2(50) := '----=\*#abc1234321cba#\*=';

l\_step PLS\_INTEGER := 12000; -- make sure you set a multiple of 3 not higher than 24573

BEGIN

l\_mail\_conn := UTL\_SMTP.open\_connection(p\_smtp\_host, p\_smtp\_port);

UTL\_SMTP.helo(l\_mail\_conn, p\_smtp\_host);

UTL\_SMTP.mail(l\_mail\_conn, p\_from);

UTL\_SMTP.rcpt(l\_mail\_conn, p\_to);

UTL\_SMTP.open\_data(l\_mail\_conn);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Date: ' || TO\_CHAR(SYSDATE, 'DD-MON-YYYY HH24:MI:SS') || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'To: ' || p\_to || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'From: ' || p\_from || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Subject: ' || p\_subject || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Reply-To: ' || p\_from || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'MIME-Version: 1.0' || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Content-Type: multipart/mixed; boundary="' || l\_boundary || '"' || UTL\_TCP.crlf || UTL\_TCP.crlf);

IF p\_text\_msg IS NOT NULL THEN

UTL\_SMTP.write\_data(l\_mail\_conn, '--' || l\_boundary || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Content-Type: text/plain; charset="iso-8859-1"' || UTL\_TCP.crlf || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, p\_text\_msg);

UTL\_SMTP.write\_data(l\_mail\_conn, UTL\_TCP.crlf || UTL\_TCP.crlf);

END IF;

IF p\_attach\_name IS NOT NULL THEN

UTL\_SMTP.write\_data(l\_mail\_conn, '--' || l\_boundary || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Content-Type: ' || p\_attach\_mime || '; name="' || p\_attach\_name || '"' || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Content-Disposition: attachment; filename="' || p\_attach\_name || '"' || UTL\_TCP.crlf || UTL\_TCP.crlf);

FOR i IN 0 .. TRUNC((DBMS\_LOB.getlength(p\_attach\_clob) - 1 )/l\_step) LOOP

UTL\_SMTP.write\_data(l\_mail\_conn, DBMS\_LOB.substr(p\_attach\_clob, l\_step, i \* l\_step + 1));

END LOOP;

UTL\_SMTP.write\_data(l\_mail\_conn, UTL\_TCP.crlf || UTL\_TCP.crlf);

END IF;

UTL\_SMTP.write\_data(l\_mail\_conn, '--' || l\_boundary || '--' || UTL\_TCP.crlf);

UTL\_SMTP.close\_data(l\_mail\_conn);

UTL\_SMTP.quit(l\_mail\_conn);

END;

/

The code below shows how the procedure is called.

DECLARE

l\_clob CLOB := 'This is a very small CLOB!';

BEGIN

send\_mail(p\_to => 'me@mycompany.com',

p\_from => 'admin@mycompany.com',

p\_subject => 'Test Message',

p\_text\_msg => 'This is a test message.',

p\_attach\_name => 'test.txt',

p\_attach\_mime => 'text/plain',

p\_attach\_clob => l\_clob,

p\_smtp\_host => 'smtp.mycompany.com');

END;

/

Multiple Recipients

When dealing with multiple recipients, the UTL\_SMTP.RCPT procedure needs to be called for each recipient, whether they are a "TO", "CC" or "BCC". The destinction between the types of recipient is made in the descriptions in the WRITE\_DATA calls. The following procedure accepts comma separated "TO", "CC" and "BCC" parameters. The "TO" is mandatory, but the others are optional. If present, they are processed appropriately by splitting the strings up using the [string\_api](https://oracle-base.com/dba/script.php?category=miscellaneous&file=string_api.sql) package.

CREATE OR REPLACE PROCEDURE send\_mail (p\_to IN VARCHAR2,

p\_cc IN VARCHAR2 DEFAULT NULL,

p\_bcc IN VARCHAR2 DEFAULT NULL,

p\_from IN VARCHAR2,

p\_subject IN VARCHAR2,

p\_message IN VARCHAR2,

p\_smtp\_host IN VARCHAR2,

p\_smtp\_port IN NUMBER DEFAULT 25)

AS

l\_mail\_conn UTL\_SMTP.connection;

PROCEDURE process\_recipients(p\_mail\_conn IN OUT UTL\_SMTP.connection,

p\_list IN VARCHAR2)

AS

l\_tab string\_api.t\_split\_array;

BEGIN

IF TRIM(p\_list) IS NOT NULL THEN

l\_tab := string\_api.split\_text(p\_list);

FOR i IN 1 .. l\_tab.COUNT LOOP

UTL\_SMTP.rcpt(p\_mail\_conn, TRIM(l\_tab(i)));

END LOOP;

END IF;

END;

BEGIN

l\_mail\_conn := UTL\_SMTP.open\_connection(p\_smtp\_host, p\_smtp\_port);

UTL\_SMTP.helo(l\_mail\_conn, p\_smtp\_host);

UTL\_SMTP.mail(l\_mail\_conn, p\_from);

process\_recipients(l\_mail\_conn, p\_to);

process\_recipients(l\_mail\_conn, p\_cc);

process\_recipients(l\_mail\_conn, p\_bcc);

UTL\_SMTP.open\_data(l\_mail\_conn);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Date: ' || TO\_CHAR(SYSDATE, 'DD-MON-YYYY HH24:MI:SS') || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'To: ' || p\_to || UTL\_TCP.crlf);

IF TRIM(p\_cc) IS NOT NULL THEN

UTL\_SMTP.write\_data(l\_mail\_conn, 'CC: ' || REPLACE(p\_cc, ',', ';') || UTL\_TCP.crlf);

END IF;

IF TRIM(p\_bcc) IS NOT NULL THEN

UTL\_SMTP.write\_data(l\_mail\_conn, 'BCC: ' || REPLACE(p\_bcc, ',', ';') || UTL\_TCP.crlf);

END IF;

UTL\_SMTP.write\_data(l\_mail\_conn, 'From: ' || p\_from || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Subject: ' || p\_subject || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, 'Reply-To: ' || p\_from || UTL\_TCP.crlf || UTL\_TCP.crlf);

UTL\_SMTP.write\_data(l\_mail\_conn, p\_message || UTL\_TCP.crlf || UTL\_TCP.crlf);

UTL\_SMTP.close\_data(l\_mail\_conn);

UTL\_SMTP.quit(l\_mail\_conn);

END;

/

BEGIN

send\_mail(p\_to => 'to@silknet.com,'to@silknet1.com,'to@silknet2.com’,

p\_cc => 'p\_cc@p\_bcc.com',

p\_bcc => 'p\_bcc@p\_bcc.com',

p\_from => 'admin@mycompany.com',

p\_subject => 'Test Message',

p\_message => 'This is a test message.',

p\_smtp\_host => 'mail. 'mail.com',

p\_smtp\_port => '25');

END;