Oracle® Application Server 10*g*

PL/SQL Web Toolkit Reference

10*g* (9.0.4)

Part No. B12098-01

September 2003

ORACLE®

Oracle Application Server 10*g* PL/SQL Web Toolkit Reference, 10*g* (9.0.4)

Part No. B12098-01

Primary Author: Peter Lubbers

Contributors: Pravin Prabhakar, Pushkar Kapasi, and Eric Lee.

Contents

9 The owa_util Package

Send Us Your Comments

Oracle Application Server 10*g* PL/SQL Web Toolkit Reference, 10*g* (9.0.4)

Part No. B12098-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors in this document, or have suggestions for improvement, please send your comments to appserverdocs_us@oracle.com. Please indicate the title and part number of the documentation and the chapter, section, and page number (if available). If you would like a reply, please give your name, address, telephone number, and email address.

If you have problems with the software, go to http://www.oracle.com/forums and find the appropriate forum to which to direct your question. These forums are constantly monitored and you should receive a response shortly. Alternatively, you can contact your local Oracle Support Services.

Preface

Intended Audience

This guide is a reference for the PL/SQL Web Toolkit for Oracle Application Server 10*g* (9.0.4).

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle Corporation is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation This documentation may contain links to Web sites of other companies or organizations that Oracle Corporation does not own or control. Oracle Corporation neither

evaluates nor makes any representations regarding the accessibility of these Web sites.

Structure

The packages in the the PL/SQL Web Toolkit are described in the following chapters:

- Chapter 1 describes the htp and htf packages which generate HTML tags.
- Chapter 2 describes the owa_cache package which contains functions and procedures that enable the PL/SQL Gateway cache.
- Chapter 3 describes the owa_cookie package which contains subprograms that send and retrieve HTTP cookies from the client's browser.
- Chapter 4 describes the owa_image package which is used when you have any image map whose destination links invoke the PL/SQL Gateway.
- Chapter 5 describes the owa_lock package which contains subprograms that are used to implement database locking for PL/SQL Gateway.
- Chapter 6 describes the owa_pattern package which locates text patterns within strings and replaces the matched string with another string.
- Chapter 7 describes the owa_sec package which contains functions, procedures, and data types to implement security.
- Chapter 8 describes the owa_text package which contains subprograms used by owa_pattern for manipulating strings.
- Chapter 9 describes the owa_util package which contains utility subprograms.

Related Documents

For more information, see the following manuals:

- *Oracle Application Server 10g mod_plsql User's Guide*
- *Oracle Application Server 10g Concepts*

Conventions

The following conventions are used in this manual:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted
boldface text	Boldface type in text indicates a term defined in the text, the glossary, or in both locations.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

1

The htp and htf Packages

The htp (hypertext procedures) and htf (hypertext functions) packages generate HTML tags. For instance, the htp.anchor procedure generates the HTML anchor tag, <A>. The following commands generate a simple HTML document:

```
create or replace procedure hello AS
BEGIN
    htp.htmlopen;           -- generates <HTML>
    htp.headopen;           -- generates <HEAD>
    htp.title('Hello');     -- generates <TITLE>Hello</TITLE>
    htp.headclose;          -- generates </HEAD>
    htp.bodyopen;           -- generates <BODY>
    htp.header(1, 'Hello'); -- generates <H1>Hello</H1>
    htp.bodyclose;          -- generates </BODY>
    htp.htmlclose;          -- generates </HTML>
END;
```

For every htp procedure that generates HTML tags, there is a corresponding htf function with identical parameters. The function versions do not directly generate output in your web page. Instead, they pass their output as return values to the statements that invoked them. Use these functions when you need to nest calls. To look up htf functions, see the entry for the corresponding htp procedures.

To print the output of htf functions, call the functions from within the htp.print procedure. It then prints its parameters to the generated web page.

> **Note:** If you use values of the LONG data type in procedures such as htp.print, htp.prn, htp.prints, htp.pa or owa_util.cellsprint, only the first 32 K of the LONG data is used. The LONG data is bound to a varchar2 data type in the procedure or function.

1.1 Summary

HTML, HEAD, and BODY Tags

htp.htmlOpen, htp.htmlClose - generate <HTML> and </HTML>

htp.headOpen, htp.headClose - generate <HEAD> and </HEAD>

htp.bodyOpen, htp.bodyClose - generate <BODY> and </BODY>

Comment Tag

htp.comment - generates <!-- and -->

Tags in the <HEAD> Area

htp.base - generates <BASE>

htp.linkRel - generates <LINK> with the REL attribute

htp.linkRev - generates <LINK> with the REV attribute

htp.title - generates <TITLE>

htp.meta - generates <META>

htp.script - generates <SCRIPT>

htp.style - generates <STYLE>

htp.isindex - generates <ISINDEX>

Applet Tags

htp.appletopen, htp.appletclose - generate <APPLET> and </APPLET>

htp.param - generates <PARAM>

List Tags

htp.olistOpen, htp.olistClose - generate <OL> and </OL>

htp.ulistOpen, htp.ulistClose - generate <UL> and </UL>

htp.dlistOpen, htp.dlistClose - generate <DL> and </DL>

htp.dlistTerm - generates <DT>

htp.dlistDef - generates <DD>

htp.dirlistOpen, htp.dirlistClose - generate <DIR> and </DIR>

htp.listHeader - generates <LH>

htp.listingOpen, htp.listingClose - generate <LISTING> and </LISTING>

htp.menulistOpen, htp.menulistClose - generate <MENU> and </MENU>

htp.listItem - generates <LI>

Form Tags

htp.formOpen, htp.formClose - generate <FORM> and </FORM>

htp.formCheckbox - generates <INPUT TYPE="CHECKBOX">

htp.formHidden - generates <INPUT TYPE="HIDDEN">

htp.formImage - generates <INPUT TYPE="IMAGE">

htp.formPassword - generates <INPUT TYPE="PASSWORD">

htp.formRadio - generates <INPUT TYPE="RADIO">

htp.formSelectOpen, htp.formSelectClose - generate <SELECT> and </SELECT>

htp.formSelectOption - generates <OPTION>

htp.formText - generates <INPUT TYPE="TEXT">

htp.formTextarea, htp.formTextarea2 - generate <TEXTAREA>

htp.formTextareaOpen, htp.formTextareaOpen2, htp.formTextareaClose - generate <TEXTAREA> and </TEXTAREA>

htp.formReset - generates <INPUT TYPE="RESET">

htp.formSubmit - generates <INPUT TYPE="SUBMIT">

Table Tags

htp.tableOpen, htp.tableClose - generate <TABLE> and </TABLE>

htp.tableCaption - generates <CAPTION>

htp.tableRowOpen, htp.tableRowClose - generate <TR> and </TR>

htp.tableHeader - generates <TH>

htp.tableData - generates <TD>

htf.format_cell - generates <TD>

IMG, HR, and A Tags

htp.line, htp.hr - generate <HR>

htp.img, htp.img2 - generate <IMG>

htp.anchor, htp.anchor2 - generates <A>

htp.mapOpen, htp.mapClose - generate <MAP> and </MAP>

Paragraph Formatting Tags

htp.header - generates heading tags (<H1> to <H6>)

htp.para, htp.paragraph - generate <P>

htp.print, htp.prn - generate any text that is passed in

htp.prints, htp.ps - generate any text that is passed in; special characters in HTML are escaped

htp.preOpen, htp.preClose - generate <PRE> and </PRE>

htp.blockquoteOpen, htp.blockquoteClose - generate <BLOCKQUOTE> and </BLOCKQUOTE>

htp.div - generates <DIV>

htp.nl, htp.br - generate

htp.nobr - generates <NOBR>

htp.wbr - generates <WBR>

htp.plaintext - generates <PLAINTEXT>

htp.address - generates <ADDRESS>

htp.mailto - generates <A> with the MAILTO attribute

htp.area - generates <AREA>

htp.bgsound - generates <BGSOUND>

Character Formatting Tags

htp.basefont - generates <BASEFONT>

htp.big - generates <BIG>

htp.bold - generates <B>

htp.center - generates <CENTER> and </CENTER>

htp.centerOpen, htp.centerClose - generate <CENTER> and </CENTER>

htp.cite - generates <CITE>

htp.code - generates <CODE>

htp.dfn - generates <DFN>

htp.get_download_files_list - generate <EM>

htp.fontOpen, htp.fontClose - generate <FONT> and </FONT>

htp.italic - generates <I>

htp.keyboard, htp.kbd - generate <KBD> and </KBD>

htp.s - generates <S>

htp.sample - generates <SAMP>

htp.small - generates <SMALL>

htp.strike - generates <STRIKE>

htp.strong - generates <STRONG>

htp.sub - generates <SUB>

htp.sup - generates <SUP>

htp.teletype - generates <TT>

htp.underline - generates <U>

htp.variable - generates <VAR>

Frame Tags

htp.frame - generates <FRAME>

htp.framesetOpen, htp.framesetClose - generate <FRAMESET> and </FRAMESET>

htp.noframesOpen, htp.noframesClose - generate <NOFRAMES> and </NOFRAMES>

1.2 htp.address

This generates the <ADDRESS> and </ADDRESS> tags, which specify the address, author and signature of a document.

Table 1–1 htp.address

Properties	Definitions

Syntax:	`htp.address (`
	`cvalue in varchar2`
	`cnowrap in varchar2 DEFAULT NULL`
	`cclear in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT`
	`NULL);`
	`htf.address (cvalue, cnowrap, cclear, cattributes) return varchar2;`

Parameters:	cvalue - the string that goes between the <ADDRESS> and </ADDRESS> tags. cnowrap - if the value for this parameter is not NULL, the NOWRAP attribute is included in the tag. cclear - the value for the CLEAR attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<ADDRESS CLEAR="*cclear*" NOWRAP *cattributes*>*cvalue*</ADDRESS>

1.3 htp.anchor, htp.anchor2

These generate the <A> and </A> HTML tags, which specify the source or destination of a hypertext link. This tag accepts several attributes, but either HREF or NAME is required. HREF specifies to where to link. NAME allows this tag to be a target of a hypertext link. The difference between these subprograms is that htp.anchor2 provides a target and therefore can be used for a frame.

Table 1–2 htp.anchor,htp.anchor2

Properties	Definitions

Syntax:	`htp.anchor (`
	`curl in varchar2`
	`ctext in varchar2`
	`cname in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.anchor (curl, ctext, cname, cattributes) return varchar2;`
	
	`htp.anchor2 (`
	`curl in varchar2`
	`ctext in varchar2`
	`cname in varchar2 DEFAULT NULL`
	`ctarget in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.anchor2 (curl, ctext, cname, ctarget, cattributes) return varchar2;`

Parameters:	curl - the value for the HREF attribute. ctext - the string that goes between the <A> and </A> tags. cname - the value for the NAME attribute. ctarget - the value for the TARGET attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<A HREF="*curl*" NAME="*cname*" *cattributes*>*ctext*</A> <A HREF="*curl*" NAME="*cname*" TARGET = "*ctarget*" *cattributes*>*ctext*</A>

1.4 htp.appletopen, htp.appletclose

htp.appletopen generates the <APPLET> HTML tag, which begins the invocation of a Java applet. Close the applet invocation with htp.appletclose, which generates the </APPLET> HTML tag.

Specify parameters to the Java applet using the htp.param procedure.

Use the cattributes parameter to specify the CODEBASE attribute since the PL/SQL cartridge does not know where to find the class files. The CODEBASE attribute specifies the virtual path containing the class files.

Table 1–3 htp.appleton, htp.appletclose

Properties	Definitions

Syntax:	`htp.appletopen(`
	`ccode in varchar2`
	`cheight in number`
	`cwidth in number`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.appletopen(ccode, cheight, cwidth, cattributes) return varchar2;`
	
	`htp.appletclose;`
	`htf.appletclose return varchar2;`

Parameters:	ccode - the value for the CODE attribute, which specifies the name of the applet class. cheight - the value for the HEIGHT attribute. cwidth - the value for the WIDTH attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<APPLET CODE=*ccode* HEIGHT=*cheight* WIDTH=*cwidth cattributes*> </APPLET>

Example:	`htp.appletopen('testclass.class', 100, 200, 'CODEBASE="/ows-applets"')`
	`generates`
	
	`<APPLET CODE="testclass.class" height=100 width=200 CODEBASE="/ows-applets">`

1.5 htp.area

This generates the <AREA> HTML tag, which defines a client-side image map. The <AREA> tag defines areas within the image and destinations for the areas.

Table 1–4 htp.area

Properties	Definitions

Syntax:	`htp.area(`
	`  ccoords  in  varchar2`
	`  cshape  in  varchar2  DEFAULT NULL`
	`  chref  in  varchar2  DEFAULT NULL`
	`  cnohref  in  varchar2  DEFAULT NULL`
	`  ctarget  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.area(ccoords, cshape, chref, cnohref, ctarget, cattributes)`
	`return varchar2;`

Parameters:	ccoords - the value for the COORDS attribute. cshape - the value for the SHAPE attribute. chref - the value for the HREF attribute. cnohref - if the value for this parameter is not NULL, the NOHREF attribute is added to the tag. ctarget - the value for the TARGET attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<AREA COORDS="*ccoords*" SHAPE="*cshape*" HREF="*chref*" NOHREF TARGET="*ctarget*" *cattributes*>

1.6 htp.base

This generates the <BASE> HTML tag, which records the URL of the document.

Table 1–5 htp.base

Properties	Definitions

Syntax:	`htp.base(`
	`  ctarget  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.base(ctarget, cattributes) return varchar2;`

Parameters:	ctarget - the value for the TARGET attribute, which establishes a window name to which all links in this document are targeted. cattributes - other attributes to be included as-is in the tag.
Generates:	<BASE HREF="<current URL>" TARGET="*ctarget*" *cattributes*>

1.7 htp.basefont

This generates the <BASEFONT> HTML tag, which specifies the base font size for a Web page.

Table 1–6 htp.basefont

Properties	Definitions
Syntax:	`htp.basefont(nsize in integer);` `htf.basefont(nsize) return varchar2;`
Parameters:	nsize - the value for the SIZE attribute.
Generates:	<BASEFONT SIZE="*nsize*">

1.8 htp.bgsound

This generates the <BGSOUND> HTML tag, which includes audio for a Web page.

Table 1–7 htp.bgsound

Properties	Definitions

Syntax:	`htp.bgsound(`
	`csrc in varchar2`
	`cloop in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.bgsound(csrc, cloop, cattributes) return varchar2;`

Parameters:	csrc - the value for the SRC attribute. cloop - the value for the LOOP attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<BGSOUND SRC="*csrc*" LOOP="*cloop*" *cattributes*>

1.9 htp.big

This generates the <BIG> and </BIG> tags, which direct the browser to render the text in a bigger font.

Table 1–8 htp.big

Properties	Definitions

Syntax:	`htp.big(`
	`  ctext  in  varchar2`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.big(ctext, cattributes) return varchar2;`

Parameters:	ctext - the text that goes between the tags. cattributes - other attributes to be included as-is in the tag.
Generates:	<BIG *cattributes*>***ctext***</BIG>

1.10 htp.blockquoteOpen, htp.blockquoteClose

This generates the <BLOCKQUOTE> and </BLOCKQUOTE> tag, which mark a section of quoted text.

Table 1–9 htp.blockquoteOpen, htp.blockquoteClose

Properties	Definitions

Syntax:	`htp.blockquoteOpen (`
	`  cnowrap  in  varchar2  DEFAULT NULL`
	`  cclear  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.blockquoteOpen (cnowrap, cclear, cattributes) return varchar2;`
	
	`htp.blockquoteClose;`
	`htf.blockquoteClose return varchar2;`

Parameters:	cnowrap - if the value for this parameter is not NULL, the NOWRAP attribute is added to the tag. cclear - the value for the CLEAR attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<BLOCKQUOTE CLEAR="*cclear*" NOWRAP *cattributes*> </BLOCKQUOTE>

1.11 htp.bodyOpen, htp.bodyClose

This generates the <BODY> and </BODY> tags, which mark the body section of an HTML document.

Table 1–10 htp.bodyOpen, htp.bodyClose

Properties	Definitions

Syntax:	`htp.bodyOpen(`
	`  cbackground  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.bodyOpen(cbackground, cattributes) return varchar2;`
	
	`htp.bodyClose;`
	`htf.bodyClose return varchar2;`

Parameters:	cbackground - the value for the BACKGROUND attribute, which specifies a graphic file to use for the background of the document. cattributes - other attributes to be included as-is in the tag.
Generates:	<BODY background="***cbackground***" ***cattributes***> </BODY>

Example:	`htp.bodyOpen('/img/background.gif');`
	`generates:`
	`<BODY background="/img/background.gif">`

1.12 htp.bold

This generates the <B> and </B> tags, which direct the browser to display the text in boldface.

Table 1–11 htp.bold

Properties	Definitions

Syntax:	`htp.bold (`
	`  ctext  in  varchar2`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.bold (ctext, cattributes) return varchar2;`

Parameters:	ctext - the text that goes between the tags. cattributes - other attributes to be included as-is in the tag.
Generates:	<B *cattributes*>*ctext*</B>

1.13 htp.center

This generates the <CENTER> and </CENTER> tags, which center a section of text within a Web page.

Table 1–12 htp.center

Properties	Definitions
Syntax:	`htp.center(ctext in varchar2);` `htf.center(ctext in varchar2) return varchar2;`
Parameters:	ctext - the text to center.
Generates:	<CENTER>*ctext*</CENTER>

1.14 htp.centerOpen, htp.centerClose

This generates the <CENTER> and </CENTER> tags, which mark the section of text to center.

Table 1–13 htp.centerOpen, htp.centerClose

Properties	Definitions

Syntax:	`htp.centerOpen;`
	`htf.centerOpen return varchar2;`
	
	`htp.centerClose;`
	`htf.centerClose return varchar2;`

Parameters:	None
Generates:	<CENTER> </CENTER>

1.15 htp.cite

This generates the <CITE> and </CITE> tags, which direct the browser to render the text as citation.

Table 1–14 htp.cite

Properties	Definitions

Syntax:	`htp.cite (`
	`  ctext  in  varchar2`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.cite (ctext, cattributes) return varchar2;`

Parameters:	ctext - the text to render as citation. cattributes - other attributes to be included as-is in the tag.
Generates:	<CITE *cattributes*>*ctext*</CITE>

1.16 htp.code

This generates the <CODE> and </CODE> tags, which direct the browser to render the text in monospace font.

Table 1–15 htp.code

Properties	Definitions

Syntax:	`htp.code (`
	`  ctext  in  varchar2`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.code (ctext, cattributes) return varchar2;`

Parameters:	ctext - the text to render as code. cattributes - other attributes to be included as-is in the tag.
Generates:	<CODE *cattributes*>*ctext*</CODE>

1.17 htp.comment

This generates the comment tags.

Table 1–16 htp.comment

Properties	Definitions
Syntax:	`htp.comment (ctext in varchar2);` `htf.comment (ctext in varchar2) return varchar2;`
Parameters:	ctext - the comment.
Generates:	<!-- *ctext* -->

1.18 htp.dfn

This generates the <DFN> and </DFN> tags, which direct the browser to render the text in italics.

Table 1–17 htp.dfn

Properties	Definitions
Syntax:	`htp.dfn(ctext in varchar2);` `htf.dfn(ctext in varchar2) return varchar2;`
Parameters:	ctext - the text to render in italics.
Generates:	<DFN>*ctext*</DFN>

1.19 htp.dirlistOpen, htp.dirlistClose

This generates the <DIR> and </DIR> tags, which create a directory list section. A directory list presents a list of items that contains up to 20 characters. Items in this list are typically arranged in columns, 24 characters wide. The <LI> tag or htp.listItem must appear directly after this tag to define the items in the list.

Table 1–18 htp.dirlistOpen, htp.dirlistClose

Properties	Definitions

Syntax:	`htp.dirlistOpen;`
	`htf.dirlistOpen return varchar2;`
	
	`htp.dirlistClose;`
	`htf.dirlistClose return varchar2;`

Parameters:	None
Generates:	<DIR> </DIR>

1.20 htp.div

This generates the <DIV> tag, which creates document divisions.

Table 1–19 htp.div

Properties	Definitions

Syntax:	`htp.div(`
	`  calign  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.div(calign, cattributes) return varchar2;`

Parameters:	calign - the value for the ALIGN attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<DIV ALIGN="*calign*" *cattributes*>

1.21 htp.dlistOpen, htp.dlistClose

This generates the <DL> and </DL> tags, which create a definition list. A definition list looks like a glossary: it contains terms and definitions. Terms are inserted using htp.dlistTerm, and definitions are inserted using htp.dlistDef.

Table 1–20 htp.dlistOpen, htp.dlistClose

Properties	Definitions

Syntax:	`htp.dlistOpen (`
	`  cclear  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.dlistOpen (cclear, cattributes) return varchar2;`
	
	`htp.dlistClose;`
	`htf.dlistClose return varchar2;`

Parameters:	cclear - the value for the CLEAR attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<DL CLEAR="*cclear*" *cattributes*> </DL>

1.22 htp.dlistDef

This generates the <DD> tag, which inserts definitions of terms. Use this tag for a definition list <DL>. Terms are tagged <DT> and definitions are tagged <DD>.

Table 1–21 htp.dlistDef

Properties	Definitions

Syntax:	`htp.dlistDef(`
	`ctext in varchar2 DEFAULT NULL`
	`cclear in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.dlistDef(ctext, cclear, cattributes) return varchar2;`

Parameters:	ctext - the definition for the term. cclear - the value for the CLEAR attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<DD CLEAR="*cclear*" *cattributes*>*ctext*

1.23 htp.dlistTerm

This generates the <DT> tag, which defines a term in a definition list <DL>.

Table 1–22 htp.dlistTerm

Properties	Definitions

Syntax:	`htp.dlistTerm (`
	`ctext in varchar2 DEFAULT NULL`
	`cclear in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.dlistTerm (ctext, cclear, cattributes) return varchar2;`

Parameters:	ctext - the term. cclear - the value for the CLEAR attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<DT CLEAR="*cclear*" *cattributes*>*ctext*

1.24 htp.download_file

After uploading files to the database, you can download them, delete them from the database, and read and write their attributes.

Table 1–23 htp.download_file

Properties	Definitions

Syntax:	`htp.download_file(`
	`  sFileName  in  varchar2`
	`  bCompress  in  boolean  DEFAULT FALSE);`

Parameters:	sFileName - file to be downloaded. bCompress - to compress file or not.
Generates:	The downloaded file.

1.25 htp.get_download_files_list

After you have downloaded files to the database, you need to get the files.

Table 1–24 htp.get_download_files_list

Properties	Definitions

Syntax:	`htp.get_download_files_list(`
	`  sFileName  out varchar2`
	`  bCompress  out binary_integer);`

Parameters:	sFileName - file to get. bCompress - to compress file or not.
Generates:	The downloaded file.

1.26 htp.emphasis, htp.em

This generates the <EM> and </EM> tags, which define text to be emphasized.

Table 1–25 htp.emphasis, htp.em

Properties	Definitions

Syntax:	`htp.em (`
	`  ctext  in  varchar2`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.em (ctext, cattributes) return varchar2;`
	
	`htp.emphasis (`
	`  ctext  in  varchar2`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.emphasis (ctext, cattributes) return varchar2;`

Parameters:	ctext - the text to emphasize. cattributes - other attributes to be included as-is in the tag.
Generates:	<EM cattributes>*ctext*</EM>

1.27 htf.escape_sc

This replaces characters that have special meaning in HTML with their escape sequences. The following characters are converted:

& to &
" to &quot:
< to <
> to >

The procedure version of this subprogram does the same thing as htp.prints and htp.ps.

Table 1–26 htf.escape_sc

Properties	Definitions
Syntax:	`htf.escape_sc(ctext in varchar2) return varchar2;` `htp.escape_sc(ctext in varchar2);`
Parameters:	ctext - the string to convert.
Generates:	The converted string.

1.28 htf.escape_url

This replaces characters that have special meaning in HTML and HTTP with their escape sequences. The following characters are converted:

& to &
" to "
< to <
> to >
% to &25

Table 1–27 htf.escape_ur

Properties	Definitions
Syntax:	`htf.escape_url(p_url in varchar2) return varchar2;`
Parameters:	p_url - the string to convert.
Generates:	The converted string.

1.29 htp.fontOpen, htp.fontClose

This generates the <FONT> and </FONT> tags, which mark a section of text with the specified font characteristics.

Table 1–28 htp.fontOpen,htp.fontClose

Properties	Definitions

Syntax:	`htp.fontOpen(`
	`ccolor in varchar2 DEFAULT NULL`
	`cface in varchar2 DEFAULT NULL`
	`csize in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.fontOpen(ccolor, cface, csize, cattributes) return varchar2;`
	
	`htp.fontClose;`
	`htf.fontClose return varchar2;`

Parameters:	ccolor - the value for the COLOR attribute. cface - the value for the FACE attribute. csize - the value for the SIZE attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<FONT COLOR="*ccolor*" FACE="*cface*" SIZE="*csize*" *cattributes*> </FONT>

1.30 htf.format_cell

This formats column values inside an HTML table using htf.tableData. It allows better control over the HTML tables.

Table 1–29 htf.format_cell

Properties	Definitions
Syntax:	`htf.format_cell ( columnValue in varchar2 format_numbers in varchar2 DEFAULT NULL) return varchar2;`
Parameters:	columnValue - the value that needs to be formatted in an HTML table. format_numbers - the format that numeric data is displayed in. If the value of this parameter is not Null, number fields are right-justified and rounded to two decimal places.
Generates:	<TD>*columnValue*</TD>

1.31 htp.formCheckbox

This generates the <INPUT> tag with TYPE="checkbox", which inserts a checkbox element in a form. A checkbox element is a button that the user toggles on or off.

Table 1–30 htp.formCheckbox

Properties	Definitions
Syntax:	`htp.formCheckbox ( cname in varchar2 cvalue in varchar2 DEFAULT 'on' cchecked in varchar2 DEFAULT NULL cattributes in varchar2 DEFAULT NULL); htf.formCheckbox (cname, cvalue, cchecked, cattributes) return varchar2;`
Parameters:	cname - the value for the NAME attribute. cvalue - the value for the VALUE attribute. cchecked - if the value for this parameter is not NULL, the CHECKED attribute is added to the tag. cattributes - other attributes to be included as-is in the tag.
Generates:	<INPUT TYPE="checkbox" NAME="*cname*" VALUE="*cvalue*" CHECKED *cattributes*>

1.32 htp.formOpen, htp.formClose

This generates the <FORM> and </FORM> tags, which create a form section in an HTML document.

Table 1–31 htp.formOpen, htp.formClose

Properties	Definitions

Syntax:	`htp.formOpen (`
	`curl in varchar2`
	`cmethod in varchar2 DEFAULT 'POST'`
	`ctarget in varchar2 DEFAULT NULL`
	`cenctype in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.formOpen (curl, cmethod, ctarget, cenctype, cattributes) return varchar2;`
	
	`htp.formClose;`
	`htf.formClose return varchar2;`

Parameters:	curl - the URL of the WRB or CGI script where the contents of the form is sent. This parameter is required. cmethod - the value for the METHOD attribute. The value can be "GET" or "POST". ctarget - the value for the TARGET attribute. cenctype - the value for the ENCTYPE attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<FORM ACTION="*curl*" METHOD="*cmethod*" TARGET="*ctarget*" ENCTYPE="*cenctype*" *cattributes*> </FORM>

1.33 htp.formFile

This generates the <INPUT> tag with TYPE="file", which allows the user to select files so that their contents may be submitted with a form.

Table 1–32 htp.formFile

Properties	Definitions

Syntax:	`function formFile(`
	`  cname  in  varchar2`
	`  caccept  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.formFile (cname, caccept, cattributes) return varchar2;`

Parameters:	cname - the value for the NAME attribute. caccept - a comma seperated list of MIME types for upload cattributes - other attributes to be included as-is in the tag.
Generates:	<INPUT TYPE="file" NAME="*cname*" ACCEPT="*caccept*" *cattributes*>

1.34 htp.formHidden

This generates the <INPUT> tag with TYPE="hidden", which inserts a hidden form element. This element is not seen by the user. It submits additional values to the script.

Table 1–33 htp.formHidden

Properties	Definitions

Syntax:	`htp.formHidden (`
	`  cname  in  varchar2`
	`  cvalue  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.formHidden (cname, cvalue, cattributes) return varchar2;`

Parameters:	cname - the value for the NAME attribute. cvalue - the value for the VALUE attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<INPUT TYPE="hidden" NAME="*cname*" VALUE="*cvalue*" *cattributes*>

1.35 htp.formImage

This generates the <INPUT> tag with TYPE="image", which creates an image field that the user clicks to submit the form immediately. The coordinates of the selected point are measured in pixels, and returned (along with other contents of the form) in two name/value pairs. The x coordinate is submitted under the name of the field

with .x appended, and the y coordinate with .y appended. Any VALUE attribute is ignored.

Table 1–34 htp.formImage

Properties	Definitions

Syntax:	`htp.formImage (`
	`cname in varchar2`
	`csrc in varchar2`
	`calign in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.formImage (cname, csrc, calign, cattributes) return varchar2;`

Parameters:	cname - the VALUE for the NAME attribute. csrc - the value for the SRC attribute, which specifies the image file. calign - the value for the ALIGN attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<INPUT TYPE="*image*" NAME="*cname*" SRC="*csrc*" ALIGN="*calign*" *cattributes*>

1.36 htp.formPassword

This generates the <INPUT> tag with TYPE="password", which creates a single-line text entry field. When the user enters text in the field, each character is represented by one asterisk. This is used for entering passwords.

Table 1–35 htp.formPassword

Properties	Definitions

Syntax:	`htp.formPassword (`
	`  cname  in  varchar2`
	`  csize  in  varchar2`
	`  cmaxlength  in  varchar2  DEFAULT NULL`
	`  cvalue  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.formPassword (cname, csize, cmaxlength, cvalue, cattributes)`
	`return varchar2;`

Parameters:	cname - the value for the NAME attribute. csize - the value for the SIZE attribute. cmaxlength - the value for the MAXLENGTH attribute. cvalue - the value for the VALUE attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<INPUT TYPE="*password*" NAME="*cname*" SIZE="*csize*" MAXLENGTH="*cmaxlength*" VALUE="*cvalue*" *cattributes*>

1.37 htp.formRadio

This generates the <INPUT> tag with TYPE="radio", which creates a radio button on the HTML form. Within a set of radio buttons, the user selects only one. Each radio button in the same set has the same name, but different values. The selected radio button generates a name/value pair.

Table 1–36 htp.formRadio

Properties	Definitions

Syntax:	`htp.formRadio (`
	`cname in varchar2`
	`cvalue in varchar2`
	`cchecked in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.formRadio (cname, cvalue, cchecked, cattributes) return varchar2;`

Parameters:	cname - the value for the NAME attribute. cvalue - the value for the VALUE attribute. cchecked - if the value for this parameter is not NULL, the CHECKED attribute is added to the tag. cattributes - other attributes to be included as-is in the tag.
Generates:	<INPUT TYPE="*radio*" NAME="*cname*" VALUE="*cvalue*" CHECKED *cattributes*>

1.38 htp.formReset

This generates the <INPUT> tag with TYPE="reset", which creates a button that, when selected, resets the form fields to their initial values.

Table 1–37 htp.formReset

Properties	Definitions

Syntax:	`htp.formReset (`
	`cvalue in varchar2 DEFAULT 'Reset'`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.formReset (cvalue, cattributes) return varchar2;`

Parameters:	cvalue - the value for the VALUE attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<INPUT TYPE="*reset*" VALUE="*cvalue*" *cattributes*>

1.39 htp.formSelectOpen, htp.formSelectClose

This generates the <SELECT> and </SELECT> tags, which creates a Select form element. A Select form element is a listbox where the user selects one or more values. The values are inserted using htp.formSelectOption.

Table 1–38 htp.formSelectOpen,htp.formSelectClose

Properties	Definitions

Syntax:	`htp.formSelectOpen (`
	`  cname  in  varchar2`
	`  cprompt  in  varchar2  DEFAULT NULL`
	`  nsize  in  integer  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.formSelectOpen (cname, cprompt, nsize, cattributes) return varchar2;`
	
	`htp.formSelectClose;`
	`htf.formSelectClose return varchar2;`

Parameters:	cname - the value for the NAME attribute. cprompt - the string preceding the list box. nsize - the value for the SIZE attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	*cprompt* <SELECT NAME="*cname"* SIZE=*"nsize*" *cattributes*> </SELECT>

Example:	`htp.formSelectOpen('greatest_player',`
	`  'Pick the greatest player:');`
	`htp.formSelectOption('Messier');`
	`htp.formSelectOption('Howe');`
	`htp.formSelectOption('Gretzky');.`
	`htp.formSelectClose;`
	
	`Generates:`
	`Pick the greatest player:`
	`<SELECT NAME="greatest_player">`
	`<OPTION>Messier`
	`<OPTION>Howe`
	`<OPTION>Gretzky`
	`</SELECT>`

1.40 htp.formSelectOption

This generates the <OPTION> tag, which represents one choice in a Select element.

Table 1–39 htp.formSelectOption

Properties	Definitions

Syntax:	`htp.formSelectOption (`
	`cvalue in varchar2`
	`cselected in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.formSelectOption (cvalue, cselected, cattributes) return varchar2;`

Parameters:	cvalue - the text for the option. cselected - if the value for this parameter is not NULL, the SELECTED attribute is added to the tag. cattributes - other attributes to be included as-is in the tag.
Generates:	<OPTION SELECTED *cattributes*>*cvalue*
Example:	See htp.formSelectOpen, htp.formSelectClose

1.41 htp.formSubmit

This generates the <INPUT> tag with TYPE="submit", which creates a button that, when clicked, submits the form.

If the button has a NAME attribute, the button contributes a name/value pair to the submitted data.

Table 1–40 htp.formSubmit

Properties	Definitions

Syntax:	`htp.formSubmit (`
	`cname in varchar2 DEFAULT NULL`
	`cvalue in varchar2 DEFAULT 'Submit'`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.formSubmit (cname, cvalue, cattributes) return varchar2;`

Parameters:	cname - the value for the NAME attribute. cvalue - the value for the VALUE attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<INPUT TYPE="submit" NAME="*cname*" VALUE="*cvalue*" *cattributes*>

1.42 htp.formText

This generates the <INPUT> tag with TYPE="text", which creates a field for a single line of text.

Table 1–41 htp.formText

Properties	Definitions

Syntax:	`htp.formText (`
	`cname in varchar2`
	`csize in varchar2 DEFAULT NULL`
	`cmaxlength in varchar2 DEFAULT NULL`
	`cvalue in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.formText (cname, csize, cmaxlength, cvalue, cattributes) return varchar2;`

Parameters:	cname - the value for the NAME attribute. csize - the value for the SIZE attribute. cmaxlength - the value for the MAXLENGTH attribute. cvalue - the value for the VALUE attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<INPUT TYPE="text" NAME="*cname*" SIZE="*csize*" MAXLENGTH="*cmaxlength*" VALUE="*cvalue*" *cattributes*>

1.43 htp.formTextarea, htp.formTextarea2

This generates the <TEXTAREA> tag, which creates a text field that has no predefined text in the text area. This field enables entering several lines of text.

The difference between these subprograms is that htp.formTextarea2 has the cwrap parameter, which specifies a wrap style.

Table 1–42 htp.formTextarea, htp.formTextarea2

Properties	Definitions

Syntax:	`htp.formTextarea (`
	`  cname  in  varchar2`
	`  nrows  in  integer`
	`  ncolumns  in  integer`
	`  calign  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.formTextarea (cname, nrows, ncolumns, calign, cattributes)`
	`return varchar2;`
	
	`htp.formTextarea2 (`
	`  cname  in  varchar2`
	`  nrows  in  integer`
	`  ncolumns  in  integer`
	`  calign  in  varchar2  DEFAULT NULL`
	`  cwrap  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.formTextarea2 (cname, nrows, ncolumns, calign, cwrap,`
	`cattributes) return varchar2;`

Parameters:	cname - the value for the NAME attribute. nrows - the value for the ROWS attribute. This is an integer. ncolumns - the value for the COLS attribute. This is an integer. calign - the value for the ALIGN attribute. cwrap - the value for the WRAP attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<TEXTAREA NAME="*cname*" ROWS="*nrows"* COLS="*ncolumns"* ALIGN="*calign*" *cattributes*></TEXTAREA> <TEXTAREA NAME="*cname*" ROWS="*nrows"* COLS="*ncolumns"* ALIGN="*calign*" WRAP="*cwrap*" *cattributes*></TEXTAREA>

1.44 htp.formTextareaOpen, htp.formTextareaOpen2, htp.formTextareaClose

This generates the <TEXTAREA> and </TEXTAREA> tags, which creates a text area form element The difference between the two open subprograms is that htp.formTextareaOpen2 has the cwrap parameter, which specifies a wrap style.

Table 1–43 htp.formTextareaOpen, htp.formTextareaOpen2, htp.formTextareaClose

Properties	Definitions

Syntax:	`htp.formTextareaOpen (`
	`  cname  in  varchar2`
	`  nrows  in  integer`
	`  ncolumns  in  integer`
	`  calign  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.formTextareaOpen (cname, nrows, ncolumns, calign, cattributes)`
	`return varchar2;`
	
	`htp.formTextareaOpen2(`
	`  cname  in  varchar2`
	`  nrows  in  integer`
	`  ncolumns  in  integer`
	`  calign  in  varchar2  DEFAULT NULL`
	`  cwrap  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.formTextareaOpen2(cname, nrows, ncolumns, calign, cwrap,`
	`cattributes) return varchar2;`
	
	`htp.formTextareaClose;`
	`htf.formTextareaClose return varchar2;`

Parameters:	cname - the value for the NAME attribute. nrows - the value for the ROWS attribute. This is an integer. ncolumns - the value for the COLS attribute. This is an integer. calign - the value for the ALIGN attribute. cwrap - the value for the WRAP attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<TEXTAREA NAME="*cname*" ROWS="*nrows*" COLS="*ncolumns"* ALIGN="*calign*" *cattributes*> <TEXTAREA NAME="*cname*" ROWS="*nrows*" COLS="*ncolumns"* ALIGN="*calign*" WRAP = "*cwrap" cattributes*> </TEXTAREA>

1.45 htp.frame

This generates the <FRAME> tag, which defines the characteristics of a frame created by a <FRAMESET> tag.

Table 1–44 htp.frame

Properties	Definitions

Syntax:	`htp.frame(`
	`csrc in varchar2`
	`cname in varchar2 DEFAULT NULL`
	`cmarginwidth in varchar2 DEFAULT NULL`
	`cmarginheight in varchar2 DEFAULT NULL`
	`cscrolling in varchar2 DEFAULT NULL`
	`cnoresize in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.frame(csrc, cname, cmarginwidth, cmarginheight, cscrolling, cnoresize, cattributes) return varchar2;`

Parameters:	csrc - the URL to display in the frame. cname - the value for the NAME attribute. cmarginwidth - the value for the MARGINWIDTH attribute. cmarginheight - the value for the MARGINHEIGHT attribute. cscrolling - the value for the SCROLLING attribute. noresize - if the value for this parameter is not NULL, the NORESIZE attribute is added to the tag. cattributes - other attributes to be included as-is in the tag.
Generates:	<FRAME SRC="*csrc*" NAME="*cname*" MARGINWIDTH="*cmarginwidth*" MARGINHEIGHT="*cmarginheight*" SCROLLING="*cscrolling*" NORESIZE *cattributes*>

1.46 htp.framesetOpen, htp.framesetClose

This generates the <FRAMESET> and </FRAMESET> tags, which define a frameset section.

Table 1–45 htp.framesetOpen, htp.framesetClose

Properties	Definitions

Syntax:	`htp.framesetOpen(`
	`  crows  in  varchar2  DEFAULT NULL`
	`  ccols  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.framesetOpen(crows, ccols, cattributes) return varchar2;`
	
	`htp.framesetClose;`
	`htf.framesetClose return varchar2;`

Parameters:	crows - the value for the ROWS attribute. ccols - the value for the COLS attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<FRAMESET ROWS="*crows*" COLS="*ccols*" *cattributes*> </FRAMESET>

1.47 htp.headOpen, htp.headClose

This generates the <HEAD> and </HEAD> tags, which mark the HTML document head section

Table 1–46 htp.headOpen, htp.headClose

Properties	Definitions

Syntax:	`htp.headOpen;`
	`htf.headOpen return varchar2;`
	
	`htp.headClose;`
	`htf.headClose return varchar2;`

Parameters:	None
Generates:	<HEAD> </HEAD>

1.48 htp.header

This generates opening heading tags (<H1> to <H6>) and their corresponding closing tags (</H1> to </H6>).

Table 1–47 htp.header

Properties	Definitions
Syntax:	htp.header (nsize in integer cheader in varchar2 calign in varchar2 DEFAULT NULL cnowrap in varchar2 DEFAULT NULL cclear in varchar2 DEFAULT NULL cattributes in varchar2 DEFAULT NULL); htf.header (nsize, cheader, calign, cnowrap, cclear, cattributes) return varchar2;
Parameters:	nsize - the heading level. This is an integer between 1 and 6. calign - the value for the ALIGN attribute. cheader - the text to display in the heading. cnowrap - the value for the NOWRAP attribute. cclear - the value for the CLEAR attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<H*nsize* ALIGN="*calign*" NOWRAP CLEAR="*cclear*" *cattributes*>*cheader*</H*nsize*>
Example:	htp.header (1,'Overview'); produces: <H1>Overview</H1>

1.49 htp.htmlOpen, htp.htmlClose

This generates the <HTML> and </HTML> tags, which mark the beginning and the end of an HTML document.

Table 1–48 htp.htmlOpen, htp.htmlClose

Properties	Definitions

Syntax:	`htp.htmlOpen;`
	`htf.htmlOpen return varchar2;`
	
	`htp.htmlClose;`
	`htf.htmlClose return varchar2;`

Parameters:	None.
Generates:	<HTML> </HTML>

1.50 htp.img, htp.img2

This generates the <IMG> tag, which directs the browser to load an image onto the HTML page. The difference between these subprograms is that htp.img2 uses the *cusemap* parameter.

Table 1–49 htp.img, htp.img2

Properties	Definitions

Syntax:	`htp.img (`
	`curl in varchar2 DEFAULT NULL`
	`calign in varchar2 DEFAULT NULL`
	`calt in varchar2 DEFAULT NULL`
	`cismap in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.img (curl, calign, calt, cismap, cattributes) return varchar2;`
	
	`htp.img2(`
	`curl in varchar2 DEFAULT NULL`
	`calign in varchar2 DEFAULT NULL`
	`calt in varchar2 DEFAULT NULL`
	`cismap in varchar2 DEFAULT NULL`
	`cusemap in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.img2(curl, calign, calt, cismap, cusemap, cattributes) return varchar2;`

Parameters:	curl - the value for the SRC attribute. calign - the value for the ALIGN attribute. calt - the value for the ALT attribute, which specifies alternative text to display if the browser does not support images. cismap - if the value for this parameter is not NULL, the ISMAP attribute is added to the tag. The attribute indicates that the image is an imagemap. cusemap - the value for the USEMAP attribute, which specifies a client-side image map. cattributes - other attributes to be included as-is in the tag.
Generates:	<IMG SRC="*curl*" ALIGN="*calign*" ALT="*calt*" ISMAP *cattributes*> <IMG SRC="*curl*" ALIGN="*calign*" ALT="*calt*" ISMAP USEMAP="*cusemap*" *cattributes*>

1.51 htp.isindex

This creates a single entry field with a prompting text, such as "*enter value*," then sends that value to the URL of the page or program.

Table 1–50 htp.isindex

Properties	Definitions

Syntax:	`htp.isindex (`
	`  cprompt  in  varchar2  DEFAULT NULL`
	`  curl  in  varchar2  DEFAULT NULL);`
	`htf.isindex (cprompt, curl) return varchar2;`

Parameters:	cprompt - the value for the PROMPT attribute. curl - the value for the HREF attribute.
Generates:	`<ISINDEX PROMPT="`*cprompt*`" HREF="`*curl*`">`

1.52 htp.italic

This generates the <I> and </I> tags, which direct the browser to render the text in italics

Table 1–51 htp.italic

Properties	Definitions

Syntax	`htp.italic (`
	`  ctext  in  varchar2`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.italic (ctext, cattributes) return varchar2;`

Parameters	ctext - the text to be rendered in italics. cattributes - other attributes to be included as-is in the tag.
Generates	<I cattributes>*ctext*</I>

1.53 htp.keyboard, htp.kbd

This generates the <KBD> and </KBD> tags, which direct the browser to render the text in monospace. These subprograms do the same thing.

Table 1–52 htp.keyboard, htp.kbd

Properties	Definitions

Syntax:	`htp.keyboard (`
	`  ctext  in  varchar2`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.keyboard (ctext, cattributes) return varchar2;`
	
	`htp.kbd (`
	`  ctext  in  varchar2`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.kbd (ctext, cattributes) return varchar2;`

Parameters:	ctext - the text to render in monospace. cattributes - other attributes to be included as-is in the tag.
Generates:	<KBD *cattributes*>*ctext*</KBD>

1.54 htp.line, htp.hr

This generates the <HR> tag, which generates a line in the HTML document.

Table 1–53 htp.line, htp.hr

Properties	Definitions
Purpose	Generates the <HR> tag, which generates a line in the HTML document.

Syntax	`htp.line (`
	`  cclear  in  varchar2  DEFAULT NULL`
	`  csrc  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.line (cclear, csrc, cattributes) return varchar2;`
	
	`htp.hr (`
	`  cclear  in  varchar2  DEFAULT NULL`
	`  csrc  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.hr (cclear, csrc, cattributes) return varchar2;`

Parameters	cclear - the value for the CLEAR attribute. csrc - the value for the SRC attribute, which specifies a custom image as the source of the line. cattributes - other attributes to be included as-is in the tag.
Generates	<HR CLEAR="*cclear*" SRC="*csrc*" *cattributes*>

1.55 htp.linkRel

This generates the <LINK> tag with the REL attribute, which gives the relationship described by the hypertext link from the anchor to the target. This is only used when the HREF attribute is present. This tag indicates a relationship between documents, but does not create a link. To create a link, use htp.anchor, htp.anchor2.

Table 1–54 htp.linkRel

Properties	Directions

Syntax:	`htp.linkRel (`
	`  crel  in  varchar2`
	`  curl  in  varchar2`
	`  ctitle  in  varchar2  DEFAULT NULL);`
	`htf.linkRel (crel, curl, ctitle) return varchar2;`

Parameters:	crel - the value for the REL attribute. curl - the value for the HREF attribute. ctitle - the value for the TITLE attribute.
Generates:	<LINK REL="*crel*" HREF="*curl*" TITLE="*ctitle*">

1.56 htp.linkRev

This generates the <LINK> tag with the REV attribute, which gives the relationship described by the hypertext link from the target to the anchor. This is the opposite of htp.linkRel. This tag indicates a relationship between documents, but does not create a link. To create a link, use htp.anchor, htp.anchor2.

Table 1–55 htp.linkRev

Properties	Definitions

Syntax:	`htp.linkRev (`
	`  crev  in  varchar2`
	`  curl  in  varchar2`
	`  ctitle  in  varchar2  DEFAULT NULL);`
	`htf.linkRev (crev, curl, ctitle) return varchar2;`

Parameters:	crev - the value for the REV attribute. curl - the value for the HREF attribute ctitle - the value for the TITLE attribute.
Generates	<LINK REV="*crev*" HREF="*curl*" TITLE="*ctitle*">

1.57 htp.listHeader

This generates the <LH> and </LH> tags, which print an HTML tag at the beginning of the list.

Table 1–56 htp.listHeader

Properties	Definitions

Syntax:	`htp.listHeader (`
	`  ctext  in  varchar2`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.listHeader (ctext, cattributes) return varchar2;`

Parameters:	ctext - the text to place between <LH> and </LH>. cattributes - other attributes to be included as-is in the tag.
Generates:	<LH cattributes>*ctext*</LH>

1.58 htp.listingOpen, htp.listingClose

This generates the <LISTING> and </LISTING> tags, which mark a section of fixed-width text in the body of an HTML page.

Table 1–57 htp.listingOpen, htp.listingClose

Properties	Definitions

Syntax:	`htp.listingOpen;`
	`htf.listingOpen return varchar2;`
	
	`htp.listingClose;`
	`htf.listingClose return varchar2;`

Parameters:	None.
Generates:	<LISTING> </LISTING>

1.59 htp.listItem

This generates the <LI> tag, which indicates a list item.

Table 1–58 htp.listItem

Properties	Definitions

Syntax:	`htp.listItem (`
	`ctext in varchar2 DEFAULT NULL`
	`cclear in varchar2 DEFAULT NULL`
	`cdingbat in varchar2 DEFAULT NULL`
	`csrc in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.listItem (ctext, cclear, cdingbat, csrc, cattributes) return varchar2;`

Parameters:	ctext - the text for the list item. cclear - the value for the CLEAR attribute. cdingbat - the value for the DINGBAT attribute. csrc - the value for the SRC attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<LI CLEAR="*cclear*" DINGBAT="*cdingbat*" SRC="*csrc*" *cattributes*>*ctext*

1.60 htp.mailto

This generates the <A> tag with the HREF set to 'mailto' prepended to the mail address argument.

Table 1–59 htp.mailto

Properties	Definitions

Syntax:	`htp.mailto (`
	`caddress in varchar2`
	`ctext in varchar2`
	`cname in varchar2`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.mailto (caddress, ctext, cname, cattributes) return varchar2;`

Table 1–59 htp.mailto

Properties	Definitions
Parameters:	caddress - the email address of the recipient. ctext - the clickable portion of the link. cname - the value for the NAME attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<A HREF="mailto:*caddress*" NAME="*cname*" *cattributes*>*ctext*</A>
Example:	htp.mailto('pres@white_house.gov','Send Email to the President'); generates: <A HREF="mailto:pres@white_house.gov">Send Email to the President</A>

1.61 htp.mapOpen, htp.mapClose

This generates the <MAP> and </MAP> tags, which mark a set of regions in a client-side image map.

Table 1–60 htp.mapOpen, htp.mapClose

Properties	Definitions

Syntax:	`htp.mapOpen(`
	`  cname  in  varchar2`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.mapOpen(cname, cattributes) return varchar2;`
	`htp.mapClose;`
	`htf.mapClose return varchar2;`

Parameters:	cname - the value for the NAME attribute. cattributes - other attributes to be included as-is in the tag
Generates:	<MAP NAME="*cname*" *cattributes*> </MAP>

1.62 htp.menulistOpen, htp.menulistClose

This generates the <MENU> and </MENU> tags, which create a list that presents one line per item. The items in the list appear more compact than an unordered list. The htp.listItem defines the list items in a menu list.

Table 1–61 htp.menulistOpen, htp.menulistClose

Properties	Definitions

Syntax:	`htp.menulistOpen;`
	`htf.menulistOpen return varchar2;`
	
	`htp.menulistClose;`
	`htf.menulistClose return varchar2;`

Parameters:	None.
Generates:	<MENU> </MENU>

1.63 htp.meta

This generates the <META> tag, which embeds meta-information about the document and also specifies values for HTTP headers. For example, you specify the expiration date, keywords, and author name.

Table 1–62 htp.meta

Properties	Definitions

Syntax:	`htp.meta (`
	`chttp_equiv in varchar2`
	`cname in varchar2`
	`ccontent in varchar2);`
	`htf.meta (chttp_equiv, cname, ccontent) return varchar2;`

Parameters:	chttp_equiv - the value for the HTTP-EQUIV attribute. cname - the value for the NAME attribute. ccontent - the value for the CONTENT attribute.
Generates:	<META HTTP-EQUIV="*chttp_equiv*" NAME ="*cname"* CONTENT="*ccontent*">
Example:	htp.meta ('Refresh', NULL, 120); generates: <META HTTP-EQUIV="Refresh" CONTENT=120> On some Web browsers, this causes the current URL to be reloaded automatically every 120 seconds.

1.64 htp.nl, htp.br

This generates the
 tag, which begins a new line of text.

Table 1–63 htp.nl, htp.br

Properties	Definitions

Syntax:	`htp.nl (`
	`  cclear  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.nl (cclear, cattributes) return varchar2;`
	
	`htp.br (`
	`  cclear  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.br (cclear, cattributes) return varchar2;`

Parameters:	cclear - the value for the CLEAR attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<BR CLEAR="*cclear*" *cattributes*>

1.65 htp.nobr

This generates the <NOBR> and </NOBR> tags, which turn off line-breaking in a section of text.

Table 1–64 htp.nobr

Properties	Definitions
Syntax:	`htp.nobr(ctext in varchar2);` `htf.nobr(ctext) return varchar2;`
Parameters:	ctext - the text that is to be rendered on one line.
Generates:	<NOBR>*ctext*</NOBR>

1.66 htp.noframesOpen, htp.noframesClose

This generates the <NOFRAMES> and </NOFRAMES> tags, which mark a no-frames section.

Table 1–65 htp.noframesOpen, htp.noframesClose

Properties	Definitions

Syntax:	`htp.noframesOpen`
	`htf.noframesOpen return varchar2;`
	
	`htp.noframesClose`
	`htf.noframesClose return varchar2;`

Parameters:	None.
Generates:	<NOFRAMES> </NOFRAMES>
See Also:	htp.frame, htp.framesetOpen, htp.framesetClose

1.67 htp.olistOpen, htp.olistClose

This generates the <OL> and </OL> tags, which define an ordered list. An ordered list presents a list of numbered items. Numbered items are added using htp.listItem.

Table 1–66 htp.olistOpen, htp.olistClose

Properties	Definitions

Syntax:	`htp.olistOpen (`
	`cclear in varchar2 DEFAULT NULL`
	`cwrap in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.olistOpen (cclear, cwrap, cattributes) return varchar2;`
	
	`htp.olistClose;`
	`htf.olistClose return varchar2;`

Parameters:	cclear - the value for the CLEAR attribute. cwrap - the value for the WRAP attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<OL CLEAR="*cclear*" WRAP="*cwrap*" *cattributes*> </OL>

1.68 htp.para, htp.paragraph

This generates the <P> tag, which indicates that the text that comes after the tag is to be formatted as a paragraph. htp.paragraph enables you to add attributes to the tag.

Table 1–67 htp.para, htp.paragraph

Properties	Definitions

Syntax:	`htp.para;`
	`htf.para return varcahr2;`
	
	`htp.paragraph (`
	`calign in varchar2 DEFAULT NULL`
	`cnowrap in varchar2 DEFAULT NULL`
	`cclear in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.paragraph (calign, cnowrap, cclear, cattributes) return varchar2;`

Parameters:	calign - the value for the ALIGN attribute. cnowrap - if the value for this parameter is not NULL, the NOWRAP attribute is added to the tag. cclear - the value for the CLEAR attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<P> <P ALIGN="*calign*" NOWRAP CLEAR="*cclear*" *cattributes*>

1.69 htp.param

This generates the <PARAM> tag, which specifies parameter values for Java applets. The values can reference HTML variables. To invoke a Java applet from a Web page, use `htp.appletopen` to begin the invocation. Use one htp.param for each desired name-value pair, and use `htp.appletclose` to end the applet invocation.

Table 1–68 htp.param

Properties	Definitions

Syntax:	`htp.param(`
	`  cname  in  varchar2`
	`  cvalue  in  varchar2);`
	`htf.param(cname, cvalue) return varchar2;`

Parameters:	cname - the value for the NAME attribute. cvalue - the value for the VALUE attribute.
Generates:	<PARAM NAME=*cname* VALUE="*cvalue*">

1.70 htp.plaintext

This generates the <PLAINTEXT> and </PLAINTEXT> tags, which direct the browser to render the text they surround in fixed-width type.

Table 1–69 htp.plaintext

Properties	Definitions

Syntax:	`htp.plaintext(`
	`  ctext  in  varchar2`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.plaintext(ctext, cattributes) return varchar2;`

Parameters:	ctext - the text to be rendered in fixed-width font. cattributes - other attributes to be included as-is in the tag.
Generates:	<PLAINTEXT *cattributes*>*ctext*</PLAINTEXT>

1.71 htp.preOpen, htp.preClose

This generates the <PRE> and </PRE> tags, which mark a section of preformatted text in the body of the HTML page.

Table 1–70 htp.preOpen, htp.preClose

Properties	Definitions

Syntax:	`htp.preOpen (`
	`  cclear  in  varchar2  DEFAULT NULL`
	`  cwidth  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.preOpen (cclear, cwidth, cattributes) return varchar2;`
	
	`htp.preClose;`
	`htf.preClose return varchar2;`

Parameters:	cclear - the value for the CLEAR attribute. cwidth - the value for the WIDTH attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<PRE CLEAR="*cclear*" WIDTH="*cwidth*" *cattributes*> </PRE>

1.72 htp.print, htp.prn

`htp.print` generates the specified parameter as a string terminated with the `\n` newline character. The `\n` character is not the same as `<BR>`. The `\n` character formats the HTML source; it does not affect how the browser renders the HTML source. Use `<BR>` to control how the browser renders the HTML source.

`htp.prn` generates the specified parameter as a string. Unlike `htp.print`, the string is not terminated with the `\n` newline character. These subprograms are procedures only, they do not come as functions.

Table 1–71 htp.print, htp.prn

Properties	Definitions

Syntax	`htp.print (cbuf in varchar2);`
	`htp.print (dbuf in date);`
	`htp.print (nbuf in number);`
	
	`htp.prn (cbuf in varchar2);`
	`htp.prn (dbuf in date);`
	`htp.prn (nbuf in number);`

Parameters:	cbuf, dbuf, nbuf - the string to generate.
Generates:	htp.print - a string terminated with a newline. htp.prn - the specified string, not terminated with a newline.

1.73 htp.prints, htp.ps

Both these subprograms generate a string and replace the following characters with the corresponding escape sequence.

< to <
> to >
" to "
& to &

If not replaced, the special characters are interpreted as HTML control characters and produce garbled output. This procedure is the same as `htp.prn` but with the character substitution. These subprograms are procedures only, they are not available as functions. Use htf.escape_sc if you need a string conversion function.

Table 1–72 htp.prints, htp.ps

Properties	Definitions
Syntax:	`htp.prints(ctext in varchar2);` `htp.ps(ctext in varchar2);`
Parameters:	ctext - the string where to perform character substitution.
Generates:	A string.

1.74 htp.s

This generates the <S> and </S> tags, which direct the browser to render the text they surround in strikethrough type.

Table 1–73 htp.s

Properties	Definitions

Syntax:	`htp.s(`
	`  ctext  in  varchar2`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.s(ctext, cattributes) return varchar2;`

Parameters:	ctext - the text to render in strikethrough type. cattributes - other attributes to be included as-is in the tag.
Generates:	<S *cattributes*>*ctext*</S>

1.75 htp.sample

This generates the <SAMP> and </SAMP> tags, which direct the browser to render the text they surround in monospace font.

Table 1–74 htp.sample

Properties	Definitions

Syntax:	`htp.sample (`
	`  ctext  in  varchar2`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.sample (ctext, cattributes) return varchar2;`

Parameters:	ctext - the text to render in monospace font. cattributes - other attributes to be included as-is in the tag.
Generates:	<SAMP *cattributes*>*ctext*</SAMP>

1.76 htp.script

This generates the <SCRIPT> and </SCRIPT> tags, which contain a script written in languages such as JavaScript and VBscript.

Table 1–75 htp.script

Properties	Definitions

Syntax:	`htp.script(`
	`  cscript  in  varchar2`
	`  clanguage  in  varchar2  DEFAULT NULL);`
	`htf.script(cscript, clanguage) return varchar2;`

Parameters:	cscript - the text of the script. This is the text that makes up the script itself, not the name of a file containing the script. clanguage - the langauge in which the script is written. If this parameter is omitted, the user's browser determines the scripting language.
Generates:	<SCRIPT LANGUAGE=*clanguage*>*cscript*</SCRIPT>

Example:	`htp.script ('script text here', 'Javascript');`
	
	`generates:`
	`<SCRIPT LANGUAGE=Javascript>`**`"script text here"`**
	`</SCRIPT>`
	
	`This causes the browser to run the script enclosed in the tags.`

1.77 htp.small

This generates the <SMALL> and </SMALL> tags, which direct the browser to render the text they surround using a small font.

Table 1–76 htp.small

Properties	Definitions

Syntax:	`htp.small(`
	`  ctext  in  varchar2`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.small(ctext, cattributes) return varchar2;`

Parameters:	ctext - the text to render in a small font. cattributes - other attributes to be included as-is in the tag.
Generates:	<SMALL *cattributes*>*ctext*</SMALL>

1.78 htp.strike

This generates the <STRIKE> and </STRIKE> tags, which direct the browser to render the text they surround in strikethrough type.

Table 1–77 htp.strike

Properties	Definitions

Syntax:	`htp.strike(`
	`  ctext  in  varchar2`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.strike(ctext, cattributes) return varchar2;`

Parameters:	ctext - the text to be rendered in strikethrough type. cattributes - other attributes to be included as-is in the tag.
Generates:	<STRIKE *cattributes*>*ctext*</STRIKE>

1.79 htp.strong

This generates the <STRONG> and </STRONG> tags, which direct the browser to render the text they surround in bold.

Table 1–78 htp.strong

Properties	Definitions

Syntax:	`htp.strong (`
	`  ctext  in  varchar2`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.strong (ctext, cattributes) return varchar2;`

Parameters:	ctext - the text to be emphasized. cattributes - other attributes to be included as-is in the tag.
Generates:	<STRONG *cattributes*>*ctext*</STRONG>

1.80 htp.style

This generates the <STYLE> and </STYLE> tags, which include a style sheet in your Web page. Style sheets are a feature of HTML 3.2. You can get more information about style sheets at **http://www.w3.org.** This feature is not compatible with browsers that support only HTML versions 2.0 or earlier. Such browsers will ignore this tag.

Table 1–79 htp.style

Properties	Definitions
Syntax:	htp.style(cstyle in varchar2); htf.style(cstyle) return varchar2;
Parameters:	cstyle - the style information to include.
Generates:	<STYLE>*cstyle*</STYLE>

1.81 htp.sub

This generates the _{and} tags, which direct the browser to render the text they surround as subscript.

Table 1–80 htp.sub

Properties	Definitions

Syntax:	`htp.sub(`
	`  ctext  in  varchar2`
	`  calign  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.sub(ctext, calign, cattributes) return varchar2;`

Parameters:	ctext - the text to render in subscript. calign - the value for the ALIGN attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	_{*ctext*}

1.82 htp.sup

This generates the ^{and} tags, which direct the browser to render the text they surround as superscript.

Table 1–81 htp.sup

Properties	Definitions

Syntax:	`htp.sup(`
	`  ctext  in  varchar2`
	`  calign  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.sup(ctext, calign, cattributes) return varchar2;`

Parameters:	ctext - the text to render in subscript. calign - the value for the ALIGN attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	^{*ctext*}

1.83 htp.tableCaption

This generates the <CAPTION> and </CAPTION> tags, which place a caption in an HTML table.

Table 1–82 htp.tableCaption

Properties	Definitions

Syntax:	`htp.tableCaption (`
	`  ccaption  in  varchar2`
	`  calign  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.tableCaption (ccaption, calign, cattributes) return varchar2;`

Parameters:	ccaption - the text for the caption. calign - the value for the ALIGN attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<CAPTION ALIGN="*calign*" *cattributes*>*ccaption*</CAPTION>

1.84 htp.tableData

This generates the <TD> and </TD> tags, which insert data into a cell of an HTML table.

Table 1–83 htp.tableData

Properties	Definitions

Syntax:	`htp.tableData (`
	`cvalue in varchar2 DEFAULT NULL`
	`calign in varchar2 DEFAULT NULL`
	`cdp in varchar2 DEFAULT NULL`
	`cnowrap in varchar2 DEFAULT NULL`
	`crowspan in varchar2 DEFAULT NULL`
	`ccolspan in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.tableData (cvalue, calign, cdp, cnowrap, crowspan, ccolspan, cattributes) return varchar2;`

Parameters:	cvalue - the data for the cell in the table. calign - the value for the ALIGN attribute. cdp - the value for the DP attribute. cnowrap - if the value of this parameter is not NULL, the NOWRAP attribute is added to the tag. crowspan - the value for the ROWSPAN attribute. ccolspan - the value for the COLSPAN attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<TD ALIGN="*calign*" DP="*cdp*" ROWSPAN="*crowspan*" COLSPAN="*ccolspan*" NOWRAP *cattributes*>*cvalue*</TD>

1.85 htp.tableHeader

This generates the <TH> and </TH> tags, which insert a header cell in an HTML table. The <TH> tag is similar to the <TD> tag, except that the text in the rows are usually rendered in bold type.

Table 1–84 htp.tableHeader

Properties	Definitions

Syntax:	`htp.tableHeader (`
	`cvalue in varchar2 DEFAULT NULL`
	`calign in varchar2 DEFAULT NULL`
	`cdp in varchar2 DEFAULT NULL`
	`cnowrap in varchar2 DEFAULT NULL`
	`crowspan in varchar2 DEFAULT NULL`
	`ccolspan in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.tableHeader (cvalue, calign, cdp, cnowrap, crowspan, ccolspan, cattributes) return varchar2;`

Parameters:	cvalue - the data for the cell in the table. calign - the value for the ALIGN attribute. cdp - the value for the DP attribute. cnowrap - if the value of this parameter is not NULL, the NOWRAP attribute is added to the tag. crowspan - the value for the ROWSPAN attribute. ccolspan - the value for the COLSPAN attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<TH ALIGN="*calign*" DP="*cdp*" ROWSPAN="*crowspan*" COLSPAN="*ccolspan*" NOWRAP *cattributes*>*cvalue*</TH>

1.86 htp.tableOpen, htp.tableClose

This generates the <TABLE> and </TABLE> tags, which define an HTML table.

Table 1–85 htp.tableOpen, htp.tableClose

Properties	Definitions

Syntax:	`htp.tableOpen (`
	`cborder in varchar2 DEFAULT NULL`
	`calign in varchar2 DEFAULT NULL`
	`cnowrap in varchar2 DEFAULT NULL`
	`cclear in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.tableOpen (cborder, calign, cnowrap, cclear, cattributes)`
	`return varchar2;`
	
	`htp.tableClose;`
	`htf.tableClose return varchar2;`

Parameters:	cborder - the value for the BORDER attribute. calign - the value for the ALIGN attribute. cnowrap - if the value of this parameter is not NULL, the NOWRAP attribute is added to the tag. cclear - the value for the CLEAR attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<TABLE "*cborder*" NOWRAP ALIGN="*calign*" CLEAR="*cclear*" *cattributes*> </TABLE>

1.87 htp.tableRowOpen, htp.tableRowClose

This generates the <TR> and </TR> tags, which inserts a new row in an HTML table.

Table 1–86 htp.tableRowOpen, htp.tableRowClose

Properties	Definitions

Syntax:	`htp.tableRowOpen (`
	`  calign  in  varchar2  DEFAULT NULL`
	`  cvalign  in  varchar2  DEFAULT NULL`
	`  cdp  in  varchar2  DEFAULT NULL`
	`  cnowrap  in  varchar2  DEFAULT NULL`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.tableRowOpen (calign, cvalign,cdp, cnowrap, cattributes) return varchar2;`
	
	`htp.tableRowClose;`
	`htp.tableRowClose return varchar2;`

Parameters:	calign - the value for the ALIGN attribute. cvalign - the value for the VALIGN attribute. cdp - the value for the DP attribute. cnowrap - if the value of this parameter is not NULL, the NOWRAP attribute is added to the tag. cattributes - other attributes to be included as-is in the tag.
Generates:	<TR ALIGN="*calign*" VALIGN="*cvalign*" DP="*cdp*" NOWRAP *catttributes*> </TR>

1.88 htp.teletype

This generates the <TT> and </TT> tags, which direct the browser to render the text they surround in a fixed width typewriter font, for example, the courier font.

Table 1–87 htp.teletype

Properties	Definitions

Syntax:	`htp.teletype (`
	`  ctext  in  varchar2`
	`  cattributes  in  varchar2  DEFAULT NULL);`
	`htf.teletype (ctext, cattributes) return varchar2;`

Parameters:	ctext - the text to render in a fixed width typewriter font. cattributes - other attributes to be included as-is in the tag.
Generates:	<TT *cattributes*>*ctext*</TT>

1.89 htp.title

This generates the <TITLE> and </TITLE> tags, which specify the text to display in the titlebar of the browser window.

Table 1–88 htp.title

Properties	Definitions
Syntax:	`htp.title (ctitle in varchar2);` `htf.title (ctitle) return varchar2;`
Parameters:	ctitle - the text to display in the titlebar of the browser window.
Generates:	<TITLE>*ctitle*</TITLE>

1.90 htp.ulistOpen, htp.ulistClose

This generates the <UL> and </UL> tags, which define an unordered list. An unordered list presents items with bullets. Add list items with htp.listItem.

Table 1–89 htp.ulistOpen, htp.ulistClose

Properties	Definitions

Syntax:	`htp.ulistOpen (`
	`cclear in varchar2 DEFAULT NULL`
	`cwrap in varchar2 DEFAULT NULL`
	`cdingbat in varchar2 DEFAULT NULL`
	`csrc in varchar2 DEFAULT NULL`
	`cattributes in varchar2 DEFAULT NULL`
	`htf.ulistOpen (cclear, cwrap, cdingbat, csrc, cattributes) return varchar2;`
	
	`htp.ulistClose;`
	`htf.ulistClose return varhar2;`

Parameters:	cclear - the value for the CLEAR attribute. cwrap - the value for the WRAP attribute. cdingbat - the value for the DINGBAT attribute. csrc - the value for the SRC attribute. cattributes - other attributes to be included as-is in the tag.
Generates:	<UL CLEAR="*cclear*" WRAP="*cwrap*" DINGBAT="*cdingbat*" SRC="*csrc*" *cattributes*> </UL>

1.91 htp.underline

This generates the <U> and </U> tags, which direct the browser to render the text they surround with an underline.

Table 1–90 htp.underline

Properties	Definitions

Syntax:	`htp.underline(`
	`ctext in varchar2`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.underline(ctext, cattributes) return varchar2;`

Parameters:	ctext - the text to render with an underline. cattributes - other attributes to be included as-is in the tag.
Generates:	<U *cattributes*>*ctext*</U>

1.92 htp.variable

This generates the <VAR> and </VAR> tags, which direct the browser to render the text they surround in italics.

Table 1–91 htp.variable

Properties	Definitions

Syntax:	`htp.variable (`
	`ctext in varchar2`
	`cattributes in varchar2 DEFAULT NULL);`
	`htf.variable (ctext, cattributes) return varchar2;`

Parameters:	ctext - the text to render in italics. cattributes - other attributes to be included as-is in the tag.
Generates:	<VAR *cattributes*>*ctext*</VAR>

1.93 htp.wbr

This generates the <WBR> tag, which inserts a soft line break within a section of NOBR text.

Table 1–92 htp.wbr

Properties	Definitions
Syntax:	`htp.wbr;` `htf.wbr return wbr;`
Parameters:	None.
Generates:	<WBR>

2

The owa_cache Package

The owa_cache package contains functions and procedures that enable the PL/SQL Gateway cache feature to improve the performance of your PL/SQL web application. This section describes the specification of these functions and procedures.

2.1 Summary

owa_cache.disable - disables the cache for this particular request.

owa_cache.set_expires - sets up the cache headers for expires model cache type.

owa_cache.set_cache - sets up the cache headers for validation model cache type.

owa_cache.set_not_modified - sets up the headers for a not modified cache hit (used in the Validation technique model only).

owa_cache.get_level - returns the caching level (used in the Validation technique model only).

owa_cache.get_etag - returns the tag associated with the cached content (used in the Validation technique model only).

2.2 owa_cache.disable

This disables the cache for this particular request.

Table 2–1 owa_cache.disable

Properties	Definitions
Parameters:	None
Exceptions:	None

2.3 owa_cache.set_expires

This sets up the cache headers for expires model cache type.

Table 2–2 owa_cache.set_expires

Properties	Definitions
Parameters:	p_expires IN - the number of minutes this content is valid. p_level IN - the caching level for it.
Exceptions:	VALUE_ERROR is thrown if p_expires is negative or zero, or p_level is not 'USER' or 'SYSTEM', or p_expires is > 525600 (1 year).

2.4 owa_cache.set_cache

This sets up the cache headers for validation model cache type.

Table 2–3 owa_cache.set_cache

Properties	Definitions
Parameters:	p_etag IN - the tag associated with this content. p_level IN - the caching level for it.
Exceptions:	VALUE_ERROR is thrown if p_etag is greater than 55, or p_level is not 'USER' or 'SYSTEM'.

2.5 owa_cache.set_not_modified

This sets up the headers for a not modified cache hit. It is used in the Validation technique only.

Table 2–4 owa_cache.set_not_modified

Properties	Definitions
Exceptions:	VALUE_ERROR is thrown if **ETag** was not passed in.

2.6 owa_cache.get_level

This returns the caching level. It is used in the Validation technique model only.

Table 2–5 owa_cache.get_level

Properties	Definitions
Returns:	The caching level string ('USER' or 'SYSTEM') for cache hit; null otherwise.

2.7 owa_cache.get_etag

This returns the tag associated with the cached content. It is used in the Validation technique only.

Table 2–6 owa_cache.get_etag

Properties	Definitions
Returns:	The tag for cache hit; null otherwise.

3

The owa_cookie Package

The owa_cookie package contains subprograms that send and retrieve HTTP cookies from the client's browser. Cookies are opaque strings sent to the browser to maintain state between HTTP calls. State can be maintained throughout the client's sessions, or longer if an expiration date is included. Your system date is calculated with reference to the information specified in the owa_custom package.

3.1 Summary

owa_cookie.cookie data type - data type to contain cookie name-value pairs.

owa_cookie.get function - gets the value of the specified cookie.

owa_cookie.get_all procedure - gets all cookie name-value pairs.

owa_cookie.remove procedure - removes the specified cookie.

owa_cookie.send procedure - generates a ""Set-Cookie" line in the HTTP header.

Note: All HTTP headers must be in English. If the headers are generated from the database, verify they are created in the English language.

3.2 owa_cookie.cookie data type

Since the HTTP standard allows cookie names to be overloaded (that is, multiple values can be associated with the same cookie name), this is a PL/SQL RECORD holding all values associated with a given cookie name.

Type vc_arr is table of varchar2(4000) index by binary_integer.

Table 3–1 owa_cookie.cookie data type

Properties	Definitions

Syntax:	`type cookie is RECORD (`
	`  name  varchar2(4000),`
	`  vals  vc_arr,`
	`  num_vals  integer);`

Returns:	Not applicable.

3.3 owa_cookie.get function

This function returns the values associated with the specified cookie. The values are returned in a owa_cookie.cookie data type.

Table 3–2 owa_cookie.get function

Properties	Definitions
Syntax:	`owa_cookie.get(name in varchar2) return cookie;`
Parameter:	name - the name of the cookie.
Returns:	An owa_cookie.cookie data type.

3.4 owa_cookie.get_all procedure

This procedure returns all cookie names and their values from the client's browser. The values appear in the order in which they were sent from the browser.

Table 3–3 owa_cookie.get_all procedures

Properties	Definitions

Syntax:	`owa_cookie.get_all(`
	`  names  out  vc_arr,`
	`  vals  out  vc_arr,`
	`  num_vals  out  integer);`

Parameters:	names - the names of the cookies. vals - the values of the cookies. num_vals - the number of cookie-value pairs.
Generates:	Arrays of the names and values in the order received, and the count of the combinations.

3.5 owa_cookie.remove procedure

This procedure forces a cookie to expire immediately by setting the "expires" field of a Set-Cookie line in the HTTP header to "01-Jan-1990". This procedure must be called within the context of an HTTP header.

Table 3–4 owa_cookie.remove procedure

Properties	Definitions

Syntax:	`owa_cookie.remove(`
	`name in varchar2,`
	`val in varchar2,`
	`path in varchar2 DEFAULT NULL);`

Parameters:	name - the name of the cookie to expire. value - the value of the cookie. path - currently unused.
Generates:	Set-Cookie: <*name*>=<*value*> expires=01-JAN-1990

3.6 owa_cookie.send procedure

This procedure generates a Set-Cookie line, which transmits a cookie to the client. This procedure must occur in the context of an HTTP header.

Table 3–5 owa_cookie.send procedure

Properties	Definitions

Syntax	`owa_cookie.send(`
	`name in varchar2,`
	`value in varchar2,`
	`expires in date DEFAULT NULL,`
	`path in varchar2 DEFAULT NULL,`
	`domain in varchar2 DEFAULT NULL,`
	`secure in varchar2 DEFAULT NULL);`

Parameters:	name -the name of the cookie. value - the value of the cookie. expires - the date at which the cookie will expire. path - the value for the path field. domain - the value for the domain field. secure - if the value of this parameter is not NULL, the "secure" field is added to the line.
Generates:	Set-Cookie: <*name*>=<*value*> expires=<*expires*> path=<*path*> domain=<*domain*> secure

4

The owa_image Package

The owa_image package contains subprograms that get the coordinates of where the user clicked on an image. Use this package when you have any image map whose destination links invoke the PL/SQL Gateway. Your procedure would look something like:

```
create or replace procedure process_image
    (my_img in owa_image.point)
    as
    x integer := owa_image.get_x(my_img);
    y integer := owa_image.get_y(my_img);
begin
    /* process the coordinate */
end
```

4.1 Summary

owa_image.NULL_POINT package variable - variable of type point whose X and Y values are NULL.

owa_image.point data type - data type to contain the X and Y values of a coordinate.

owa_image.get_x function - gets the X value of a point type.

owa_image.get_y function - gets the Y value of a point type.

4.2 owa_image.NULL_POINT package variable

This package variable of type `point` is used to default point parameters. Both the X and the Y fields of this variable are NULL.

Table 4–1 owa_image.NULL_POINT package variable

Properties	Definitions
Syntax:	`null_point - package variable`
Returns:	Not applicable.

4.3 owa_image.point data type

This data type provides the x and y coordinates of a user's click on an imagemap. It is defined as:

```
type point is table of varchar2(32767) index by binary_integer.
```

Table 4–2 owa_image.point data type

Properties	Definitions
Syntax:	`point - data type`
Returns:	Not applicable.

4.4 owa_image.get_x function

This function returns the X coordinate of the point where the user clicked on an image map.

Table 4–3 owa_image.get_x function

Properties	Definitions
Syntax:	`owa_image.get_x(p in point) return integer;`
Parameter:	p - the point where the user clicked.
Returns:	The X coordinate as an integer.

4.5 owa_image.get_y function

This function returns the Y coordinate of the point where the user clicked on an image map.

Table 4–4 owa_image.get_y function

Properties	Definitions
Syntax:	`owa_image.get_y(p in point) return integer;`
Parameters:	p - the point where the user clicked.
Returns:	The Y coordinate as an integer.

5

The owa_opt_lock Package

The owa_opt_lock package contains subprograms that impose optimistic locking strategies, so as to prevent lost updates. It checks if the row that the user is interested in updating has been changed by someone else in the meantime.

The PL/SQL Gateway cannot use conventional database locking schemes because HTTP is a stateless protocol. The owa_opt_lock package gives you two ways of dealing with the lost update problem:

- The hidden fields method stores the previous values in hidden fields in the HTML page. When the user requests an update, the PL/SQL Gateway checks these values against the current state of the database. The update operation is performed only if the values match. To use this method, call the owa_opt_lock.store_values procedure.
- The checksum method stores a checksum rather than the values themselves. To use this method, call the owa_opt_lock.checksum function.

These methods are optimistic. They do not prevent other users from performing updates, but they do reject the current update if an intervening update has occurred.

5.1 Summary

owa_opt_lock.vcArray data type - data type to contain ROWIDs.

owa_opt_lock.checksum function - returns the checksum value.

owa_opt_lock.get_rowid function - returns the ROWID value.

owa_opt_lock.store_values procedure - stores unmodified values in hidden fields for later verification.

owa_opt_lock.verify_values function - verifies the stored values against modified values.

5.2 owa_opt_lock.vcArray data type

This data type is a PL/SQL table intended to hold ROWIDs.

Type vcArray is table of varchar2(2000) index by binary_integer.

Note that this is different from the owa_text.vc_arr data type.

Table 5–1 owa_opt_lock.vcArray data type

Properties	Definitions
Syntax:	owa_opt_lock.vcArray - data type
Returns:	Not applicable.

5.3 owa_opt_lock.checksum function

This function returns a checksum value for a specified string, or for a row in a table. For a row in a table, the function calculates the checksum value based on the values of the columns in the row. This function comes in two versions.

The first version returns a checksum based on the specified string. This is a "pure" 32-bit checksum executed by the database and based on the Internet 1 protocol.

The second version returns a checksum based on the values of a row in a table. This is a "impure" 32-bit checksum based on the Internet 1 protocol.

Table 5–2 owa_opt_lock.checksum function

Properties	Definitions

Syntax:	`owa_opt_lock.checksum(p_buff in varchar2) return number;`
	`owa_opt_lock.checksum(`
	`p_owner in varchar2`
	`p_tname in varchar2`
	`p_rowid in rowid) return number;`

Parameters:	p_buff - the string where you want to calculate the checksum. p_owner - the owner of the table. p_tname - the table name. p_rowid - the row in *p_tname* where you want to calculate the checksum value. Use the owa_opt_lock.get_rowid function to convert vcArray values to proper rowids.
Returns:	A checksum value.

5.4 owa_opt_lock.get_rowid function

This function returns the ROWID data type from the specified owa_opt_lock.vcArray data type.

Table 5–3 owa_opt_lock.get_rowid function

Properties	Definitions
Syntax:	`owa_opt_lock.get_rowid(p_old_values in vcArray) return rowid;`
Parameters:	p_old_values - this parameter is usually passed in from an HTML form.
Returns:	A ROWID.

5.5 owa_opt_lock.store_values procedure

This procedure stores the column values of the row that you want to update later. The values are stored in hidden HTML form elements. Before updating the row, compare these values with the current row values to ensure that the values in the row have not been changed. If the values have changed, you can warn the users and let them decide if the update should take place.

Table 5–4 owa_opt_lock.store_values procedure

Properties	Definitions

Syntax:	`owa_opt_lock.store_values(`
	`p_owner in varchar2`
	`p_tname in varchar2`
	`p_rowid in rowid);`

Parameters:	p_owner - the owner of the table. p_tname - the name of the table. p_rowid - the row where you want to store values.
Generates:	A series of hidden form elements: One hidden form element is created for the table owner. The name of the element is "old_*p_tname*", where *p_tname* is the name of the table. The value of the element is the owner name. One hidden form element is created for the table name. The name of the element is "old_*p_tname*", where *p_tname* is the name of the table. The value of the element is the table name. One element is created for each column in the row. The name of the element is "old_*p_tname*", where *p_tname* is the name of the table. The value of the element is the column value.
See also:	owa_opt_lock.verify_values function.

5.6 owa_opt_lock.verify_values function

This function verifies whether values in the specified row have been updated since the last query. Use this function with the owa_opt_lock.store_values procedure.

Table 5–5 owa_opt_lock.verify_values function

Properties	Definitions
Syntax:	`owa_opt_lock.verify_values(p_old_values in vcArray) return boolean;`
Parameters:	p_old_values - a PL/SQL table containing the following information: p_old_values(1) specifies the owner of the table. p_old_values(2) specifies the table. p_old_values(3) specifies the rowid of the row to verify. The remaining indexes contain values for the columns in the table. Typically, this parameter is passed in from the HTML form, where you have previously called the owa_opt_lock.store_values procedure to store the row values on hidden form elements.
Returns:	TRUE if no other update has been performed; otherwise, FALSE.
See also:	owa_opt_lock.store_values procedure.

6

The owa_pattern Package

The owa_pattern package in the PL/SQL Web Toolkit locates text patterns within strings and replaces the matched string with another string. Use regular expressions with the subprograms in this package.

6.1 Subprograms

Regular Expressions - this section describes the special characters, quantifiers, and flags used in forming regular expressions.

owa_pattern.amatch function - determines if a string contains the specified pattern. It lets you specify where in the string the match has to occur.

owa_pattern.change function and procedure - replaces a pattern within a string. If you call it as a function it returns the number of times the regular expression was found and replaced.

owa_pattern.getpat procedure - generates a pattern data type from a VARCHAR2 type.

owa_pattern.match function - determines if a string contains the specified pattern.

owa_pattern.pattern data type - data type to store regular expressions.

These subprograms are overloaded. That is, there are several versions of each, distinguished by the parameters they use. Specifically, there are six versions of MATCH, and four each of AMATCH and CHANGE. The subprograms use the following parameters:

- line - This is the target to be examined for a match. It can be more than one line of text or a owa_text.multi_line data type.

- pat - This is the pattern that the subprograms attempt to locate in line. The pattern can contain regular expressions. In the owa_pattern.change function and procedure, this parameter is called from_str.
- flags - This specifies whether the search is case-sensitive or if substitutions are done globally.

6.2 Regular Expressions

Specify a regular expression by creating the string you want to match interspersed with various wildcard tokens and quantifiers.

6.2.1 Wildcard Tokens

Wildcard tokens match something other than themselves:

Table 6–1 Wildcard tokens recognized by owa_pattern package

Token	Description
^	Matches newline or the beginning of the target
$	Matches newline or the end of the target
\n	Matches newline
.	Matches any character except newline
\t	Matches tab
\d	Matches digits [0-9]
\D	Matches non-digits [not 0-9]
\w	Matches word characters (0-9, a-z, A-Z, or _)
\W	Matches non-word characters (not 0-9, a-z, A-Z, or _)
\s	Matches whitespace characters (blank, tab, or newline).
\S	Matches non-whitespace characters (not blank, tab, or newline)
\b	Matches "word" boundaries (between \w and \W)
\x<HEX>	Matches the value in the current character set of the two hexadecimal digits
\<OCT>	Matches the value in the current character set of the two or three octal digits
\	Followed by any character not covered by another case matches that character

Table 6–1 Wildcard tokens recognized by owa_pattern package

Token	Description
&	Applies only to CHANGE. This causes the string that matched the regular expression to be included in the string that replaces it. This differs from the other tokens in that it specifies how a target is changed rather than how it is matched. This is explained further under CHANGE.

6.2.2 Quantifiers

Any tokens except & can have their meaning extended by any of the following quantifiers. You can also apply these quantifiers to literals:

Table 6–2 Quantifiers

Quantifier	Description
?	0 or 1 occurrence(s)
*	0 or more occurrences
+	1 or more occurrence(s)
{n}	Exactly *n* occurrences
(n,}	At least *n* occurrences
{n,m}	At least *n*, but not more than *m*, occurrences

6.2.3 Flags

In addition to targets and regular expressions, the owa_pattern functions and procedures use flags to affect how they are interpreted.

Table 6–3 Flags

Flag	Description
i	This indicates a case-insensitive search.
g	This applies only to CHANGE. It indicates a global replace. That is, all portions of the target that match the regular expression are replaced.

6.3 owa_pattern.amatch function

This function specifies if a pattern occurs in a particular location in a string. There are four versions to this function:

- The first and second versions of the function do not save the matched tokens (these are saved in the *backrefs* parameters in the third and fourth versions). The difference between the first and second versions is the *pat* parameter, which can be a VARCHAR2 or a pattern data type.
- The third and fourth versions of the function save the matched tokens in the *backrefs* parameter. The difference between the third and fourth versions is the *pat* parameter, which can be a VARCHAR2 or a pattern data type.

Note: If multiple overlapping strings match the regular expression, this function takes the longest match.

Table 6–4 owa_pattern.amatch function

Properties	Definitions

Syntax:	`owa_pattern.amatch(`
	`  line  in  varchar2`
	`  from_loc  in  integer`
	`  pat  in  varchar2`
	`  flags  in  varchar2  DEFAULT NULL) return integer;`
	
	`owa_pattern.amatch(`
	`  line  in  varchar2`
	`  from_loc  in  integer`
	`  pat  in out  pattern`
	`  flags  in  varchar2  DEFAULT NULL) return integer;`
	
	`owa_pattern.amatch(`
	`  line  in  varchar2`
	`  from_loc  in  integer`
	`  pat  in  varchar2`
	`  backrefs  out  owa_text.vc_arr`
	`  flags  in  varchar2  DEFAULT NULL) return integer;`

Table 6–4 owa_pattern.amatch function

Properties	Definitions

	`owa_pattern.amatch(`
	`line in varchar2`
	`from_loc in integer`
	`pat in out pattern`
	`backrefs out owa_text.vc_arr`
	`flags in varchar2 DEFAULT NULL) return integer;`

Parameters:	line - the text to search in. from_loc - the location (in number of characters) in *line* where the search is to begin. pat - the string to match. It can contain regular expressions. This can be either a VARCHAR2 or a pattern. If it is a pattern, the output value of this parameter is the pattern matched. backrefs - the text that is matched. Each token that is matched is placed in a cell in the owa_text.vc_arr data type PL/SQL table. flags - whether or not the search is case-sensitive. If the value of this parameter is "i", the search is case-insensitive. Otherwise the search is case-sensitive.
Returns:	The index of the character after the end of the match, counting from the beginning of *line*. If there was no match, the function returns `0`.

6.4 owa_pattern.change function and procedure

This function or procedure searches and replaces a string or multi_line data type. If multiple overlapping strings match the regular expression, this function takes the longest match.

Table 6–5 owa_pattern.change function and procedure

Properties	Definitions

Syntax:	`/* function */`
	`owa_pattern.change(`
	`line in out varchar2`
	`from_str in varchar2`
	`to_str in varchar2`
	`flags in varchar2 DEFAULT NULL) return integer;`
	
	`/* procedure */`
	`owa_pattern.change(`
	`line in out varchar2`
	`from_str in varchar2`
	`to_str in varchar2`
	`flags in varchar2 DEFAULT NULL);`
	
	`/* function */`
	`owa_pattern.change(`
	`mline in out owa_text.multi_line`
	`from_str in varchar2`
	`to_str in varchar2`
	`flags in varchar2 DEFAULT NULL) return integer;`
	
	`/* procedure */`
	`owa_pattern.change(`
	`mline in out owa_text.multi_line`
	`from_str in varchar2`
	`to_str in varchar2`
	`flags in varchar2 DEFAULT NULL);`

Table 6–5 owa_pattern.change function and procedure

Properties	Definitions
Parameters:	line - the text to search in. The output value of this parameter is the altered string. mline - the text to search in. This is a owa_text.multi_line data type data type. The output value of this parameter is the altered string. from_str - the regular expression to replace. to_str - the substitution pattern. flags - whether or not the search is case-sensitive, and whether or not changes are to be made globally. If "i" is specified, the search is case-insensitive. If "g" is specified, changes are made to all matches. Otherwise, the function stops after the first substitution is made.
Returns:	As a function, it returns the number of substitutions made. If the flag `g' is not used, this number can only be 0 or 1 and only the first match is replaced. The flag `g' specifies to replace all matches with the regular expression.

Example:	Example 1:
	`owa_pattern.change('Cats in pajamas', 'C.+in', '& red ')`
	The regular expression matches the substring `Cats in'. It then replaces this string with `& red'. The ampersand character (&) indicates `Cats in', since that's what matched the regular expression. Thus, this procedure replaces the string `Cats in pajamas' with 'Cats in red'. If you called this as a function instead of a procedure, the value it returns is 1, indicating that a single substitution has been made.
	Example 2:
	`create or replace procedure test_pattern as theline VARCHAR2(256);`
	` num_found integer;`
	`begin`
	`theline := 'what is the goal?';`
	` num_found := owa_pattern.change(theline, 'goal', 'idea', 'g');`
	`htp.print(num_found); -- num_found is 1`
	`htp.print(theline); -- theline is 'what is the idea?'`
	`end;`
	`/`
	`show errors`

6.5 owa_pattern.getpat procedure

This procedure converts a VARCHAR2 string into a owa_pattern.pattern data type.

Table 6–6 owa_pattern.getpat procedure

Properties	Definitions
Syntax:	`owa_pattern.getpat(arg in VARCHAR2, pat in out pattern);`
Parameters:	arg - the string to convert. pat - the owa_pattern.pattern data type initialized with *arg*.
Returns:	None.

6.6 owa_pattern.match function

This function determines if a string contains the specified pattern. The pattern can contain regular expressions. If multiple overlapping strings can match the regular expression, this function takes the longest match.

The regular expression in this function can be either a VARCHAR2 or a owa_pattern.pattern data type. Create a owa_pattern.pattern data type from a string using the owa_pattern.getpat procedure.

Create a multi_line data type from a long string using the owa_text.stream2multi procedure. If a multi_line is used, the rlist parameter specifies a list of chunks where matches were found.

If the line is a string and not a multi_line, you can add an optional output parameter called backrefs. This parameter is a row_list that holds each string in the target that was matched by a sequence of tokens in the regular expression.

Table 6–7 owa_pattern.match function

Properties	Definitions

Syntax:	`owa_pattern.match(`
	`  line  in  varchar2`
	`  pat  in  varchar2`
	`  flags  in  varchar2  DEFAULT NULL) return boolean;`
	
	`owa_pattern.match(`
	`  line  in  varchar2`
	`  pat  in out  pattern`
	`  flags  in  varchar2  DEFAULT NULL) return boolean;`
	
	`owa_pattern.match(`
	`  line  in  varchar2`
	`  pat  in  varchar2`
	`  backrefs  out  owa_text.vc_arr`
	`  flags  in  varchar2  DEFAULT NULL) return boolean;`
	
	`owa_pattern.match(`
	`  line  in  varchar2`
	`  pat  in out  pattern`
	`  backrefs  out  owa_text.vc_arr`
	`  flags  in  varchar2  DEFAULT NULL) return boolean;`
	
	`owa_pattern.match(`
	`  mline  in  owa_text.multi_line`
	`  pat  in  varchar2`
	`  rlist  out  owa_text.row_list`
	`  flags  in  varchar2  DEFAULT NULL) return boolean;`
	
	`owa_pattern.match(`
	`  mline  in  owa_text.multi_line`
	`  pat  in out  pattern`
	`  rlist  out  owa_text.row_list`
	`  flags  in  varchar2  DEFAULT NULL) return boolean;`

Table 6–7 owa_pattern.match function

Properties	Definitions
Parameters:	line - the text to search in. mline - the text to search in. This is a owa_text.multi_line data type data type. pat - the pattern to match. This is either a VARCHAR2 or a owa_pattern.pattern data type data type. It it is a pattern, the output value of this parameter is the pattern matched. backrefs - the text that is matched. Each token that is matched is placed in a cell in the owa_text.vc_arr data type PL/SQL table. This parameter is a row_list that holds each string in the target that was matched by a sequence of tokens in the regular expression. rlist - an output parameter containing a list of matches. flags - whether or not the search is case-sensitive. If the value of this parameter is "i", the search is case-insensitive. Otherwise the search is case-sensitive.
Returns:	TRUE if a match was found, FALSE otherwise.

Examples:	KAZOO is the target where it is searching for the `zoo.*` regular expression. The period indicates any character other than newline, and the asterisk matches 0 or more of the preceding characters. In this case, it matches any character other than the newline.
	Therefore, this regular expression specifies that a matching target consists of `zoo`, followed by any set of characters neither ending in nor including a newline (which does not match the period). The i flag indicates to ignore case in the search. In this case, the function returns TRUE, which indicates that a match had been found.
	`boolean foundMatch;`
	`foundMatch := owa_pattern.match('KAZOO', 'zoo.*', 'i');`
	The following example searches for the string "goal" followed by any number of characters in *sometext*. If found,
	`sometext VARCHAR2(256);`
	`pat VARCHAR2(256);`
	
	`sometext := 'what is the goal?'`
	`pat := 'goal.*';`
	`if owa.pattern.match(sometext, pat) then`
	`htp.print('Match found');`
	`else`
	`htp.print('Match not found');`
	`end if;`

6.7 owa_pattern.pattern data type

You can use a pattern as both an input and output parameter. Thus, you can pass the same regular expression to OWA_PATTERN function calls, and it only has to be parsed once.

Table 6–8 owa_pattern.pattern data type

Properties	Definitions
Syntax:	`owa_pattern.pattern - data type`
Returns:	Not applicable.

7

The owa_sec Package

This chapter describes the functions, procedures, and data types in the `owa_sec` package in the PL/SQL Web Toolkit.

> **Note:** Use the procedures and functions in the `owa_sec` package for custom authentication.

Parameters that have default values are optional.

7.1 Summary

owa_sec.get_client_hostname function - returns the client's hostname.

owa_sec.get_client_ip function - returns the client's IP address.

owa_sec.get_password function - returns the password that the user entered.

owa_sec.get_user_id function - returns the username that the user entered.

owa_sec.set_authorization procedure - enables the PL/SQL application to use custom authentication.

owa_sec.set_protection_realm procedure - defines the realm that the page is in.

7.2 owa_sec.get_client_hostname function

This function returns the hostname of the client.

Table 7–1 owa_sec.get_client_hostname function

Properties	Definitions
Syntax:	`owa_sec.get_client_hostname return varchar2;`
Parameters:	None.
Returns:	The hostname.

7.3 owa_sec.get_client_ip function

This function returns the IP address of the client.

Table 7–2 owa_sec.get_client_ip function

Properties	Definitions
Syntax:	`owa_sec.get_client_ip return owa_util.ip_address;`
Parameters:	None.

Returns:	The IP address. The owa_util.ip_address data type is a PL/SQL table where the first four elements contain the four numbers of the IP address. For example, if the IP address is `123.45.67.89` and the variable *ipaddr* is of the owa_util.ip_address data type, the variable would contain the following values:
	`ipaddr(1) = 123`
	`ipaddr(2) = 45`
	`ipaddr(3) = 67`
	`ipaddr(4) = 89`

7.4 owa_sec.get_password function

This function returns the password that the user used to log in. For security reasons, this function returns a true value only when custom authentication is used. If you call this function when you are not using custom authentication, the function returns an undefined value. Thus, the database passwords are not exposed.

Table 7–3 owa_sec.get_password function

Properties	Definitions
Syntax:	`owa_sec.get_password return varchar2;`
Parameters:	None.
Returns:	The password.

7.5 owa_sec.get_user_id function

This function returns the username that the user used to log in.

Table 7–4 owa_sec.get_user_id function

Properties	Definitions
Syntax:	`owa_sec.get_user_id return varchar2;`
Parameters:	None.
Returns:	The username.

7.6 owa_sec.set_authorization procedure

This procedure, called in the initialization portion of the owa_custom package, sets the authorization scheme for the PL/SQL Gateway. This implements your `authorize` function, which authorizes the user before his requested procedure is run. The placement of the `authorize` function depends on the scheme you selected.

Table 7–5 owa_sec.set_authorization procedure

Properties	Definitions
Syntax:	owa_sec.set_authorization(scheme in integer);
Parameters:	scheme - the authorization scheme. It is one of the following schemes for set_ authorization: OWA_SEC.NO_CHECK - Specifies that the PL/SQL application is not to do any custom authentication. This is the default. OWA_SEC.GLOBAL - Defines an authorize function that is called for all users and all procedures. The function is **owa_custom.authorize** function in the "oas_public" schema. OWA_SEC.PER_PACKAGE - Define an `authorize` function that is called when procedures in a package or anonymous procedures are called. If the procedures are in a package, the *package*.**authorize** function in the user's schema is called to authorize the user. If the procedures are not in a package, then the anonymous `authorize` function in the user's schema is called. OWA_SEC.CUSTOM - Implements different authorize functions for each user. The function **owa_custom.authorize** in the user's schema is called to authorize the user. If the user's schema does not contain an **owa_custom.authorize** function, the PL/SQL Gateway looks for it in the "oas_public" schema. The custom `authorize` function has the following signature: `function authorize return boolean;` If the function returns TRUE, authentication succeeded. If it returns FALSE, authentication failed. If the `authorize` function is not defined, the Gateway returns an error and fails.
Returns:	Not applicable

7.7 owa_sec.set_protection_realm procedure

This procedure sets the realm of the page that is returned to the user. The user enters a username and login that already exist in the realm.

Table 7–6 owa_sec.set_protection_realm procedure

Properties	Definitions
Syntax:	`owa_sec.set_protection_realm(realm in varchar2);`
Parameters:	realm - the realm where the page belongs. This string is displayed to the user.
Returns:	Not applicable.

8

The owa_text Package

The owa_text package contains subprograms used by owa_pattern for manipulating strings. They are externalized so you can use them directly.

8.1 Summary

owa_text.add2multi procedure - adds text to an existing multi_line type.

owa_text.multi_line data type - data type for holding large amounts of text.

owa_text.new_row_list - creates a new row_list.

owa_text.print_multi procedure - prints out the contents of a multi_list.

owa_text.print_row_list procedure - prints out the contents of a row_list.

owa_text.row_list data type - data type for holding data to be processed.

owa_text.stream2multi procedure - converts a varchar2 to a multi_line type.

owa_text.vc_arr data type - data type for holding large amounts of text.

8.2 owa_text.add2multi procedure

This procedure adds content to an existing owa_text.multi_line data type.

Table 8–1 owa_text.add2multi procedure

Properties	Definitions

Syntax:	`owa_text.add2multi(`
	`  stream  in  varchar2`
	`  mline  in out  multi_line`
	`  continue  in  boolean  DEFAULT TRUE);`

Parameters:	stream - the text to add. mline - the owa_text.multi_line data type. The output of this parameter contains *stream*. continue - if TRUE, the procedure appends *stream* within the previous final row (assuming it is less than 32K). If FALSE, the procedure places *stream* in a new row.
Returns:	None.

8.3 owa_text.multi_line data type

This data type is a PL/SQL record that holds large amounts of text. The rows field, of type owa_text.vc_arr data type, contains the text data in the record.

Table 8–2 owa_text.multi_line data type

Properties	Definitions

Syntax:	`type multi_line is record (`
	`  rows  vc_arr,`
	`  num_rows  integer,`
	`  partial_row  boolean);`

Returns:	Not applicable.

8.4 owa_text.new_row_list

This function or procedure creates a new owa_text.row_list data type. The function version uses no parameters and returns a new empty row_list. The procedure version creates the row_list data type as an output parameter.

Table 8–3 owa_text.new_row_list

Properties	Definitions

Syntax:	`/* procedure */`
	`owa_text.new_row_list(rlist out row_list);`
	`/* function */`
	`owa_text.new_row_list return row_list;`

Parameters:	rlist - this is an output parameter containing the new row_list data type.
Returns:	The function version returns the new row_list data type.

8.5 owa_text.print_multi procedure

This procedure uses htp.print, htp.prn to print the "rows" field of the owa_text.multi_line data type.

Table 8–4 owa_text.print_multi procedure

Properties	Definitions
Syntax:	`owa_text.print_multi(mline in multi_line);`
Parameters:	mline - the multi_line data type to print.
Returns:	The contents of the multi_line.

8.6 owa_text.print_row_list procedure

This procedure uses htp.print, htp.prn to print the "rows" field of the owa_text.row_list data type.

Table 8–5 owa_text.print_row_list procedure

Properties	Definitions
Syntax:	`owa_text.print_row_list(rlist in row_list);`
Parameters:	rlist - the row_list data type to print.
Returns:	The contents of the row_list.

8.7 owa_text.row_list data type

This is the data type for holding data to be processed.

Table 8–6 owa_text.row_list data type

Properties	Definitions

Syntax:	`type row_list is record (`
	`  rows  int_arr,`
	`  num_rows  integer);`
	`int_arr is defined as:`
	`type int_arr is table of integer index by binary_integer;`

Returns:	Not applicable.

8.8 owa_text.stream2multi procedure

This procedure converts a string to a multi_line data type.

Table 8–7 owa_text.stream2multi procedure

Properties	Definitions
Syntax:	owa_text.stream2multi(stream in varchar2 mline out multi_line);
Parameters:	stream - the string to convert. mline - the stream in owa_text.multi_line data type format.
Returns:	None.

8.9 owa_text.vc_arr data type

This is a component of the owa_text.multi_line data type.

Table 8–8 owa_text.vc_arr data type

Properties	Definitions
Syntax:	type vc_arr is table of varchar2(32767) index by binary_integer;
Returns:	Not applicable.

9

The owa_util Package

The owa_util package contains utility subprograms for performing operations such as getting the value of CGI environment variables, printing the data that is returned to the client, and printing the results of a query in an HTML table.

The owa_util contains three types of utility subprograms.

- Dynamic SQL Utilities enable you to produce pages with dynamically generated SQL code.
- HTML utilities enable you to retrieve the values of CGI environment variables and perform URL redirects.
- Date utilities enable correct date-handling. Date values are simple strings in HTML, but are treated as a data type by the Oracle database.

9.1 Summary

owa_util.bind_variables function - prepares a SQL query and binds variables to it.

owa_util.calendarprint procedure - prints a calendar.

owa_util.cellsprint procedure - prints the contents of a query in an HTML table.

owa_util.choose_date procedure - generates HTML form elements that allow the user to select a date.

owa_util.dateType data type - data type to hold date information.

owa_util.get_cgi_env function - returns the value of the specified CGI environment variable.

owa_util.get_owa_service_path function - returns the full virtual path for the PL/SQL Gateway.

owa_util.get_procedure function - returns the name of the procedure that is invoked by the PL/SQL Gateway.

owa_util.http_header_close procedure - closes the HTTP header.

owa_util.ident_arr data type - a data type

owa_util.ip_address data type - used by the owa_sec.get_client_ip function.

owa_util.listprint procedure - generates a HTML form element that contains data from a query.

owa_util.mime_header procedure - generates the Content-type line in the HTTP header.

owa_util.print_cgi_env procedure - generates a list of all CGI environment variables and their values.

owa_util.redirect_url procedure - generates the Location line in the HTTP header.

owa_util.showpage procedure - prints a page generated by the htp and htf packages in SQL*Plus.

owa_util.showsource procedure - prints the source for the specified subprogram.

owa_util.signature procedure - prints a line that says that the page is generated by the PL/SQL Agent.

owa_util.status_line procedure - generates the Status line in the HTTP header.

owa_util.tablePrint function - prints the data from a table in the database as an HTML table.

owa_util.todate function - converts dateType data to the standard PL/SQL date type.

owa_util.who_called_me procedure - returns information on the caller of the procedure.

9.2 owa_util.bind_variables function

This function prepares a SQL query by binding variables to it, and stores the output in an opened cursor. Use this function as a parameter to a procedure sending a dynamically generated query. Specify up to 25 bind variables.

Table 9–1 owa_util.bind_variables function

Properties	Definitions

Syntax:	`owa_util.bind_variables(`
	`theQuery in varchar2 DEFAULT NULL`
	`bv1Name in varchar2 DEFAULT NULL`
	`bv1Value in varchar2 DEFAULT NULL`
	`bv2Name in varchar2 DEFAULT NULL`
	`bv2Value in varchar2 DEFAULT NULL`
	`bv3Name in varchar2 DEFAULT NULL`
	`bv3Value in varchar2 DEFAULT NULL`
	`...`
	`bv25Name in varchar2 DEFAULT NULL`
	`bv25Value in varchar2 DEFAULT NULL) return integer;`

Parameters:	theQuery - the SQL query statement. This must be a SELECT statement. bv1Name - the name of the variable. bv2Value - the value of the variable.
Returns:	An integer identifying the opened cursor.

9.3 owa_util.calendarprint procedure

This procedure creates a calendar in HTML. Each date in the calendar can contain any number of hypertext links. Design your query as follows:

- The first column is a DATE. This correlates the information produced by the query with the calendar output generated by the procedure.
- The query output must be sorted on this column using ORDER BY.
- The second column contains the text, if any, that you want printed for that date.
- The third column contains the destination for generated links. Each item in the second column becomes a hypertext link to the destination given in this column. If this column is omitted, the items in the second column are simple text, not links.

This procedure has 2 versions. Version 1 uses a hard-coded query stored in a varchar2 string. Version 2 uses a dynamic query prepared with the owa_util.bind_variables function.

Table 9–2 owa_util.calendarprint procedure

Properties	Definitions

Syntax:	`owa_util.calendarprint(`
	`  p_query  in  varchar2`
	`  p_mf_only  in  varchar2  DEFAULT 'N');`
	
	`owa_util.calendarprint(`
	`  p_cursor  in  integer`
	`  p_mf_only  in  varchar2  DEFAULT 'N');`

Parameters:	p_query - a PL/SQL query. See the description above on what the query returns. p_cursor - a PL/SQL cursor containing the same format as *p_query.* p_mf_only - if "N" (the default), the generated calendar includes Sunday through Saturday. Otherwise, it includes Monday through Friday only.
Generates:	A calendar in the form of an HTML table with a visible border.

9.4 owa_util.cellsprint procedure

This procedure generates an HTML table from the output of a SQL query. SQL atomic data items are mapped to HTML cells and SQL rows to HTML rows. You must write the code to begin and end the HTML table. There are nine versions of this procedure:

- The first version passes the results of a query into an index table. Perform the query and cellsprint does the formatting. To have more control in generating an HTML table from the output of an SQL query, use the htf.format_cell function.
- The second and third versions display rows (up to the specified maximum) returned by the query or cursor.
- The fourth and fifth versions exclude aspecified number of rows from the HTML table. Use the fourth and fifth versions to scroll through result sets by saving the last row seen in a hidden form element.

The sixth through ninth versions are the same as the first four versions, except that they return a row count output parameter.

Table 9–3 owa_util.cellsprint procedure

Properties	Definitions
Syntax:	owa_util.cellsprint(p_colCnt in integer p_resultTbl in vc_arr p_format_numbers in varchar2 DEFAULT NULL); owa_util.cellsprint(p_theQuery in varchar2 p_max_rows in number DEFAULT 100 p_format_numbers in varchar2 DEFAULT NULL); owa_util.cellsprint(p_theCursor in integer p_max_rows in number DEFAULT 100 p_format_numbers in varchar2 DEFAULT NULL); owa_util.cellsprint(p_theQuery in varchar2 p_max_rows in number DEFAULT 100 p_format_numbers in varchar2 DEFAULT NULL p_skip_rec in number DEFAULT 0 p_more_data out boolean); owa_util.cellsprint(p_theCursor in integer p_max_rows in number DEFAULT 100 p_format_numbers in varchar2 DEFAULT NULL p_skip_rec in number DEFAULT 0 p_more_data out boolean); owa_util.cellsprint(p_theQuery in varchar2 p_max_rows in number DEFAULT 100 p_format_numbers in varchar2 DEFAULT NULL p_reccnt out number); owa_util.cellsprint(p_theCursor in integer p_max_rows in number DEFAULT 100 p_format_numbers in varchar2 DEFAULT NULL p_reccnt out number);

Table 9–3 owa_util.cellsprint procedure

Properties	Definitions
	owa_util.cellsprint(p_theQuery in varchar2 p_max_rows in number DEFAULT 100 p_format_numbers in varchar2 DEFAULT NULL p_skip_rec in number DEFAULT 0 p_more_data out boolean p_reccnt out number); owa_util.cellsprint(p_theCursor in integer p_max_rows in number DEFAULT 100 p_format_numbers in varchar2 DEFAULT NULL p_skip_rec in number DEFAULT 0 p_more_data out boolean p_reccnt out number);
Parameters:	p_colCnt - the number of columns in the table. p_theQuery - a SQL SELECT statement. p_theCursor - a cursor ID. This can be the return value from the owa_util.bind_variables function. p_max_rows - the maximum number of rows to print. p_format_numbers - if the value of this parameter is not NULL, number fields are right-justified and rounded to two decimal places. p_skip_rec - the number of rows to exclude from the HTML table. p_more_data - TRUE if there are more rows in the query or cursor, FALSE otherwise. p_reccnt - the number of rows that have been returned by the query. This value does not include skipped rows (if any). p_resultTbl - the index table which will contain the result of the query. Each entry in the query will correspond to one column value.
Generates:	<tr><td>**QueryResultItem**</td><td>**QueryResultItem**</td></tr> <tr><td>**QueryResultItem**</td><td>**QueryResultItem**</td></tr>

9.5 owa_util.choose_date procedure

This procedure generates three HTML form elements that allow the user to select the day, the month, and the year. The parameter in the procedure that receives the data from these elements must be a owa_util.dateType data type. Use the owa_

util.todate function to convert the owa_util.dateType data type value to the standard Oracle7 DATE data type.

Table 9–4 owa_util.choose_date procedure

Properties	Definitions

Syntax:	`owa_util.choose_date(`
	`  p_name  in  varchar2,`
	`  p_date  in  date  DEFAULT SYSDATE);`

Parameters:	p_name - the name of the form elements. p_date - the initial date that is selected when the HTML page is displayed.

Generates:	`<SELECT NAME="p_name" SIZE="1">`
	`<OPTION value="01">1`
	`  ...`
	`<OPTION value="31">31`
	`</SELECT>`
	`-`
	`<SELECT NAME="p_name" SIZE="1">`
	`<OPTION value="01">JAN`
	`  ...`
	`<OPTION value="12">DEC`
	`</SELECT>`
	`-`
	`<SELECT NAME="p_name" SIZE="1">`
	`<OPTION value="1992">1992`
	`  ...`
	`<OPTION value="2002">2002`
	`</SELECT>`

9.6 owa_util.dateType data type

The owa_util.todate function converts an item of this type to the type DATE, which is understood and properly handled as data by the database. The procedure owa_util.choose_date procedure enables the user to select the desired date.

Table 9–5 owa_util.dateType data type

Properties	Definitions
Syntax:	`type dateType is table of varchar2(10) index by binary_integer;`
Returns:	Not applicable.

9.7 owa_util.get_cgi_env function

This function returns the value of the specified CGI environment variable. Although the WRB is not operated through CGI, many WRB cartridges, including the PL/SQL Gateway, can make use of CGI environment variables.

Table 9–6 owa_util.get_cgi_env function

Properties	Definitions
Syntax:	`owa_util.get_cgi_env(param_name in varchar2) return varchar2;`
Parameters:	param_name - the name of the CGI environment variable. It is case-insensitive. Get the values of all CGI environment variables except for QUERY_STRING because the PL/SQL Gateway parses the value of QUERY_STRING to determine the parameters to pass to the stored procedure.
Returns:	The value of the specified CGI environment variable. If the variable is not defined, the function returns NULL.

9.8 owa_util.get_owa_service_path function

This function returns the full virtual path of the PL/SQL Gateway that is handling the request.

Table 9–7 owa_util.get_owa_service_path function

Properties	Definitions
Syntax:	`owa_util.get_owa_service_path return varchar2;`
Parameters:	None.
Returns:	A virtual path of the PL/SQL Gateway that is handling the request.

9.9 owa_util.get_procedure function

This function returns the name of the procedure that is being invoked by the PL/SQL Gateway.

Table 9–8 owa_util.get_procedure function

Properties	Definitions
Syntax:	`owa_util.get_procedure return varchar2;`
Parameters:	None.
Returns:	The name of a procedure, including the package name if the procedure is defined in a package.

9.10 owa_util.http_header_close procedure

This procedure generates a newline character to close the HTTP header. Use this procedure if you have not closed the header by using the *bclose_header* parameter in calls such as owa_util.mime_header procedure, owa_util.redirect_url procedure, or owa_util.status_line procedure. The HTTP header must be closed before any `htp.print` or `htp.prn` calls.

Table 9–9 owa_util.http_header_close procedure

Properties	Definitions
Syntax:	`owa_util.http_header_close;`
Parameters:	None.
Generates:	A newline character, which closes the HTTP header.

9.11 owa_util.ident_arr data type

This data type is used for an array.

Table 9–10 owa_util.ident_arr data type

Properties	Definitions
Syntax:	`type ident_arr is table of varchar2(30) index by binary_integer;`
Returns:	Not applicable.

9.12 owa_util.ip_address data type

This data type is used by the owa_sec.get_client_ip function.

Table 9–11 owa_util.ip_address data type

Properties	Definitions
Syntax:	`type ip_address is table of integer index by binary_integer;`
Returns:	Not applicable.

9.13 owa_util.listprint procedure

This procedure generates an HTML selection list form element from the output of a SQL query. The columns in the output of the query are handled in the following manner:

- The first column specifies the values that are sent back. These values are for the VALUE attribute of the OPTION tag.
- The second column specifies the values that the user sees.
- The third column specifies whether or not the row is marked as SELECTED in the OPTION tag. If the value is not NULL, the row is selected.

There are two versions of this procedure. The first version contains a hard-coded SQL query, and the second version uses a dynamic query prepared with the owa_util.bind_variables function.

Table 9–12 owa_util.listprint procedure

Properties	Definitions

Syntax:	`owa_util.listprint(`
	`  p_theQuery  in  varchar2`
	`  p_cname  in  varchar2`
	`  p_nsize  in  number`
	`  p_multiple  in  boolean  DEFAULT FALSE);`
	
	`owa_util.listprint(`
	`  p_theCursor  in  integer`
	`  p_cname  in  varchar2`
	`  p_nsize  in  number`
	`  p_multiple  in  boolean  DEFAULT FALSE);`

Parameters:	p_theQuery - the SQL query. p_theCursor - the cursor ID. This can be the return value from the owa_util.bind_variables function. p_cname - the name of the HTML form element. p_nsize - the size of the form element (this controls how many items the user can see without scrolling). p_multiple - whether multiple selection is permitted.

Generates:	`<SELECT NAME="p_cname" SIZE="p_nsize">`
	`<OPTION SELECTED value='value_from_the_first_column'>value_from_the_second_column`
	`<OPTION SELECTED value='value_from_the_first_column'>value_from_the_second_column`
	`  ...`
	`</SELECT>`

9.14 owa_util.mime_header procedure

This procedure changes the default MIME header that the script returns. This procedure must come before any `htp.print` or `htp.prn` calls to direct the script not to use the default MIME header.

Table 9–13 owa_util.mime_header procedure

Properties	Definitions

Syntax:	`owa_util.mime_header(`
	`  ccontent_type in varchar2 DEFAULT 'text/html',`
	`  bclose_header in boolean DEFAULT TRUE,`
	`  ccharset in varchar2 DEFAULT NULL);`

Parameters:	ccontent_type - the MIME type to generate. bclose_header - whether or not to close the HTTP header. If TRUE, two newlines are sent, which closes the HTTP header. Otherwise, one newline is sent, and the HTTP header remains open. ccharset - the character set to use.
Generates:	Content-type: <ccontent_type>; charset=<ccharset>

Example:	`owa_util.mime_header('text/plain', false, 'ISO-8859-4')`
	`generates:`
	`Content-type: text/plain; charset=ISO-8859-4\n`

9.15 owa_util.print_cgi_env procedure

This procedure generates all the CGI environment variables and their values made available by the PL/SQL Gateway to the stored procedure.

Table 9–14 owa_util.print_cgi_env procedure

Properties	Definitions
Syntax:	`owa_util.print_cgi_env;`
Parameters:	None.
Generates:	A list in the following format: `cgi_env_var_name = value\n`

9.16 owa_util.redirect_url procedure

This procedure specifies that the application server is to visit the specified URL. The URL may specify either a web page to return or a program to execute. This procedure must come before any htp or htf procedure or function call.

Table 9–15 owa_util.redirect_url procedure

Properties	Definitions

Syntax:	`owa_util.redirect_url(`
	`curl in varchar2`
	`bclose_header in boolean DEFAULT TRUE);`

Parameters:	curl - the URL to visit. bclose_header - whether or not to close the HTTP header. If TRUE, two newlines are sent, which closes the HTTP header. Otherwise, one newline is sent, and the HTTP header is still open.
Generates:	Location: <*curl*>\n\n

9.17 owa_util.showpage procedure

This procedure prints out the HTML output of a procedure in SQL*Plus, SQL*DBA, or Oracle Server Manager. The procedure must use the `htp` or `htf` packages to generate the HTML page, and this procedure must be issued after the procedure has been called and before any other HTP or HTF subprograms are directly or indirectly called. This method is useful for generating pages filled with static data. This procedure uses **dbms_output** and is limited to 255 characters per line and an overall buffer size of 1,000,000 bytes.

Table 9–16 owa_util.showpage procedure

Properties	Definitions
Syntax:	`owa_util.showpage;`
Parameters:	None.

Generates:	The output of htp procedure is displayed in SQL*Plus, SQL*DBA, or Oracle Server Manager. For example:
	`SQL> set serveroutput on`
	`SQL> spool gretzky.html`
	`SQL> execute hockey.pass("Gretzky")`
	`SQL> execute owa_util.showpage`
	`SQL> exit`
	This would generate an HTML page that could be accessed from Web browsers.

9.18 owa_util.showsource procedure

This procedure prints the source of the specified procedure, function, or package. If a procedure or function which belongs to a package is specified, then the entire package is displayed.

Table 9–17 owa_util.showsource procedure

Properties	Definitions
Syntax:	`owa_util.showsource (cname in varchar2);`
Parameters:	cname - name of the procedure or function.
Generates:	The source code of the specified function, procedure, or package.

9.19 owa_util.signature procedure

This procedure generates an HTML line followed by a signature line on the HTML document. If a parameter is specified, the procedure also generates a hypertext link to view the PL/SQL source for that procedure. The link calls the owa_util.showsource procedure.

Table 9–18 owa_util.signature procedure

Properties	Definitions
Syntax:	`owa_util.signature;` `owa_util.signature (cname in varchar2);`
Parameters:	cname - the function or procedure whose source you want to show.

Generates:	Without a parameter, the procedure generates a line that looks like the following:
	`This page was produced by the **PL/SQL Agent** on August 9, 2001 09:30.`
	With a parameter, the procedure generates a signature line in the HTML document that looks like the following:
	`This page was produced by the **PL/SQL Agent** on 8/09/01 09:30`
	`**View PL/SQL Source**`

9.20 owa_util.status_line procedure

This procedure sends a standard HTTP status code to the client. This procedure must come before any `htp.print` or `htp.prn` calls so that the status code is returned as part of the header, rather than as "content data".

Table 9–19 owa_util.status_line procedure

Properties	Definitions

Syntax:	`owa_util.status_line(`
	`nstatus in integer,`
	`creason in varchar2 DEFAULT NULL`
	`bclose_header in boolean DEFAULT TRUE);`

Parameters:	nstatus - the status code. creason - the string for the status code. bclose_header - whether or not to close the HTTP header. If TRUE, two newlines are sent, which closes the HTTP header. Otherwise, one newline is sent, and the HTTP header is still open.
Generates:	Status: <*nstatus*> <*creason*>\n\n

9.21 owa_util.tablePrint function

This function generates either preformatted tables or HTML tables (depending on the capabilities of the user's browser) from database tables. RAW columns are supported, but LONG RAW columns are not. References to LONG RAW columns will print the result 'Not Printable'. In this function, *cattributes* is the second, rather than the last, parameter.

Table 9–20 owa_util.tablePrint function

Properties	Definitions

Syntax:	`owa_util.tablePrint(`
	`ctable in varchar2`
	`cattributes in varchar2 DEFAULT NULL`
	`ntable_type in integer DEFAULT HTML_TABLE`
	`ccolumns in varchar2 DEFAULT '*'`
	`cclauses in varchar2 DEFAULT NULL`
	`ccol_aliases in varchar2 DEFAULT NULL`
	`nrow_min in number DEFAULT 0`
	`nrow_max in number DEFAULT NULL) return boolean;`

Parameters:	ctable - the database table. cattributes - other attributes to be included as-is in the tag. ntable_type - how to generate the table. Specify "`HTML_TABLE`" to generate the table using <TABLE> tags or "`PRE_TABLE`" to generate the table using the <PRE> tags. ccolumns - a comma-delimited list of columns from *ctable* to include in the generated table. cclauses - WHERE or ORDER BY clauses, which specify which rows to retrieve from the database table, and how to order them. ccol_aliases - a comma-delimited list of headings for the generated table. nrow_min - the first row, of those retrieved, to display. nrow_max - the last row, of those retrieved, to display.
Generates:	A preformatted or HTML table. Returns TRUE if there are more rows beyond the nrow_max requested, FALSE otherwise.

Table 9–20 owa_util.tablePrint function

Properties	Definitions
Examples:	For browsers that do not support HTML tables, create the following procedure:

```
create or replace procedure showemps is
ignore_more boolean;
begin
ignore_more := owa_util.tablePrint('emp', 'BORDER', OWA_UTIL.PRE_
TABLE);
end;
```

Requesting a URL such as:
`http://myhost:8080/ows-bin/hr/plsql/showemps` returns the following to the client:

```
<PRE>
---------------------------------------------------------------------
| EMPNO |ENAME |JOB      |MGR   |HIREDATE   | SAL | COMM | DEPTNO |
---------------------------------------------------------------------
| 7369| SMITH  | CLERK   | 7902 | 17-DEC-80 | 800  |      | 20     |
| 7499| ALLEN  | SALESMAN| 7698 | 20-FEB-81 | 1600 | 300  | 30     |
| 7521| WARD   | SALESMAN| 7698 | 22-FEB-81 | 1250 | 500  | 30     |
| 7566| JONES  | MANAGER | 7839 | 02-APR-81 | 2975 |      | 20     |
| 7654| MARTIN | SALESMAN| 7698 | 28-SEP-81 | 1250 | 1400 | 30     |
| 7698| BLAKE  | MANAGER | 7839 | 01-MAY-81 | 2850 |      | 30     |
| 7782| CLARK  | MANAGER | 7839 | 09-JUN-81 | 2450 |      | 10     |
| 7788| SCOTT  | ANALYST | 7566 | 09-DEC-82 | 3000 |      | 20     |
| 7839| KING   |PRESIDENT|      | 17-NOV-81 | 5000 |      | 10     |
| 7844| TURNER | SALESMAN| 7698 | 08-SEP-81 | 1500 | 0    | 30     |
| 7876| ADAMS  | CLERK   | 7788 | 12-JAN-83 | 1100 |      | 20     |
| 7900| JAMES  | CLERK   | 7698 | 03-DEC-81 | 950  |      | 30     |
| 7902| FORD   | ANALYST | 7566 | 03-DEC-81 | 3000 |      | 20     |
| 7934| MILLER | CLERK   | 7782 | 23-JAN-82 | 1300 |      | 10     |
---------------------------------------------------------------------
</PRE>
```

To view the employees in department 10, and only their employee ids, names, and salaries, create the following procedure:

```
create or replace procedure showemps_10 is
 ignore_more boolean;
begin
 ignore_more := owa_util.tablePrint
    ('EMP', 'BORDER', OWA_UTIL.PRE_TABLE,
 'empno, ename, sal',
 'where deptno=10 order by empno',
 'Employee Number, Name, Salary');
end;
```

Table 9–20 owa_util.tablePrint function

Properties	Definitions

A request for a URL like **http://myhost:8080/ows-bin/hr/plsql/showemps_10** would return the following to the client:

```
<PRE>
------------------------------------
| Employee Number |Name     | Salary |
------------------------------------
| 7782            | CLARK   | 2450   |
| 7839            | KING    | 5000   |
| 7934            | MILLER  | 1300   |
------------------------------------
</PRE>
```

For browsers that support HTML tables, to view the department table in an HTML table, create the following procedure:

```
create or replace procedure showdept is
 ignore_more boolean;
begin
 ignore_more := owa_util.tablePrint('dept', 'BORDER');
end;
```

Table 9–20 owa_util.tablePrint function

Properties	Definitions
	A request for a URL like **http://myhost:8080/ows-bin/hr/plsql/showdept** would return the following to the client:

```
<TABLE BORDER>
<TR>
<TH>DEPTNO</TH>
<TH>DNAME</TH>
<TH>LOC</TH>
</TR>
<TR>
<TD ALIGN="LEFT">10</TD>
<TD ALIGN="LEFT">ACCOUNTING</TD>
<TD ALIGN="LEFT">NEW YORK</TD>
</TR>
<TR>
<TD ALIGN="LEFT">20</TD>
<TD ALIGN="LEFT">RESEARCH</TD>
<TD ALIGN="LEFT">DALLAS</TD>
</TR>
<TR>
<TD ALIGN="LEFT">30</TD>
<TD ALIGN="LEFT">SALES</TD>
<TD ALIGN="LEFT">CHICAGO</TD>
</TR>
<TR>
<TD ALIGN="LEFT">40</TD>
<TD ALIGN="LEFT">OPERATIONS</TD>
<TD ALIGN="LEFT">BOSTON</TD>
</TR>
</TABLE>
```

A Web browser would format this to look like the following table:

```
--------------------------------------
| DEPTNO  | DNAME         | LOC      |
|-------------------------------------|
| 10      | ACCOUNTING    | NEW YORK |
| 20      | RESEARCH      | DALLAS   |
| 30      | SALES         | CHICAGO  |
--------------------------------------
```

9.22 owa_util.todate function

This function converts the owa_util.dateType data type to the standard Oracle database DATE type.

Table 9–21 owa_util.todate function

Properties	Definitions
Syntax:	`owa_util.todate(p_dateArray in dateType) return date;`
Parameters:	p_dateArray - the value to convert.
Generates	A standard DATE.

9.23 owa_util.who_called_me procedure

This procedure returns information (in the form of output parameters) about the PL/SQL code unit that invoked it.

Table 9–22 owa_util.who_called_me procedure

Properties	Definitions

Syntax:	`owa_util.who_called_me(`
	`owner out varchar2`
	`name out varchar2`
	`lineno out number`
	`caller_t out varchar2);`

Parameters:	owner - the owner of the program unit. name - the name of the program unit. This is the name of the package, if the calling program unit is wrapped in a package, and the name of the procedure or function if the calling program unit is a stand-alone procedure or function. If the calling program unit is part of an anonymous block, this is NULL. lineno - the line number within the program unit where the call was made. caller_t - the type of program unit that made the call. The possibilities are: package body, anonymous block, procedure, and function. Procedure and function are only for stand-alone procedures and functions.
Generates	Not applicable.

Index

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W