

# Orbit 5.0 OML Quick Reference

## OML - Orbit Markup Language

Token	Description
<b>Token Definition</b>	
<b>&lt;token&gt;</b> <b>[A-Za-z0-9_+,:@\$/%]</b>  <b>. = [ ] are reserved</b>	<p>A Token Name consists of any combination of letters (A-Z, a-z), numbers (0-9), underscore (_), minus (-), plus (+), comma (,), colon (:), or special characters (@\$/%). The characters period (.), equals (=) and square brackets ([]) are reserved for Token Modifiers, Token Assignment and Token Functions.</p> <p>Global Tokens are created in the Token Maintenance Screen; Internal Tokens are defined within the Orbit Engine; Query Tokens take their names from the columns in your SQL Select Clause; Local Tokens may be created on the fly via Token Assignment.</p> <p>All tokens are referenced in the same way: by enclosing their names in pound signs (#). Tokens names are case-insensitive, so #Token_Name# is the same as #TOKEN_NAME# which is the same as #token_name#.</p>
<b>Token Modifiers</b>	
<b>#&lt;token&gt;.&lt;modifier&gt;#</b> <b>#&lt;token&gt;.&lt;modifier&gt;.&lt;modifier&gt;...#</b>	<p>A token modifier is an OML operator that is used to alter the way a token is displayed at a given reference. Token Modifiers have no effect on stored token contents; self-assignment can be used to permanently change the value of a token (e.g., #token=PARSE[#token.UPPER#]). You can string together more than one Token Modifier to affect a single token at once (e.g., #token.NBSP.UPPER.TRIM#).</p> <p>The following list of Token Modifiers is given in order of Modifier Precedence. For example, note that .TRIM comes before .LENGTH - this tells you that #token.TRIM.LENGTH# and #token.LENGTH.TRIM# are both parsed in the same manner, by first trimming the spaces off the ends of the #token# value and then counting the number of characters left in the string.</p>
<b>&lt;token&gt;.TRIM</b>	Displays <token> with any preceding and ending spaces removed.
<b>&lt;token&gt;.LTRIM</b>	Displays <token> with any preceding spaces removed.
<b>&lt;token&gt;.RTRIM</b>	Displays <token> with any ending spaces removed.
<b>&lt;token&gt;.LENGTH</b>	Displays the character length of the <token> value. This modifier is fired after the TRIM modifiers.
<b>&lt;token&gt;.0</b>	Displays 0 (zero) if the token is NULL. This is useful for comparing primary keys in a SQL Query so that if the token is NULL no SQL error will display.
<b>&lt;token&gt;.1</b>	Displays 1 (one) if the token is NULL. This is useful for OML Logic expressions where a NULL value should be interpreted as TRUE, or for numeric tokens that should default to 1.
<b>&lt;token&gt;.NBSP</b>	Displays &nbsp; if the token is NULL. This is useful in table data cells when you want to show the border and the token is null. Note: once the Orbit Engine finds this modifier, other modifiers are irrelevant and modifier processing is halted for the token.
<b>&lt;token&gt;.UPPER</b>	Displays <token> in all UPPER case.
<b>&lt;token&gt;.LOWER</b>	Displays <token> in all LOWER case.
<b>&lt;token&gt;.INITCAP</b>	Displays <token> with the initial letter of all words capitalized and the rest in lower case.
<b>&lt;token&gt;.NAMECAP</b> <b>&lt;token&gt;.INITNAME</b>	Same as INITCAP, but people's names are displayed correctly for suffixes and standard prefixes (e.g., III, IV, McName).
<b>&lt;token&gt;.RAW</b>	Escapes HTML and OML characters in <token> to display raw Orbit source code without parsing. Replaces the characters &lt; > # " with HTML Escape Sequences. Equivalent to the effect of token modifiers .NOTAGS, .NOPARSE and .NOQUOTE combined.
<b>&lt;token&gt;.TR</b>	Translates <token> for use in a URL address. The characters: % & # + < > " are translated to their hexadecimal equivalent. For example, if #P1# contained the value "Save & New", then in the URL: www.site.com/prod/Orbit.Show_Page?version=MY_VERSION&page=MY_PAGE&P1=#P1.TR# #P1.TR# would be replaced with: Save%20%26%20New
<b>&lt;token&gt;.NULL</b>	Displays the string "NULL" if the token is NULL. This is useful in SQL Queries. Note: once the Orbit Engine finds this modifier, other modifiers are irrelevant and modifier processing is halted for the token.
<b>&lt;token&gt;.SIGN</b>	Displays the output of the SQL SIGN function applied to the numeric value of the token: -1 if value < 0, 0 if value = 0, 1 if value > 0.
<b>&lt;token&gt;.CHECKED</b>	Used for displaying HTML checkboxes in their correct state. If the value of the token is '-1', '1', 'Y', 'T' or 'TRUE', the word CHECKED is displayed. If the value is '0', 'N', 'F', 'FALSE' or NULL, the value is set to NULL. This token modifier can go within the <input> tag for checkbox objects (e.g., <input type="checkbox" name="my_flag" #flag.CHECKED# value="1">). Note: once the Orbit Engine finds this modifier, other modifiers are irrelevant and modifier processing is halted for the token.
<b>&lt;token&gt;.CHECKBOX</b>	Used for displaying read-only checkbox images in their correct state. If the value of the token is '-1', '1', 'Y', 'T' or 'TRUE', the global token #CHECK_ON# will be displayed. If the value is '0', 'N', 'F', 'FALSE' or NULL, the global token #CHECK_OFF# will be displayed. CHECK_ON and CHECK_OFF should contain HTML image tags for displaying checked or unchecked checkboxes. Note: once the Orbit Engine finds this modifier, other modifiers are irrelevant and modifier processing is halted for the token.
<b>&lt;token&gt;.EXISTS</b> <b>&lt;token&gt;.ISNOTNULL</b>	Displays 1 (one) if the token value IS NOT NULL, 0 otherwise.
<b>&lt;token&gt;.NOTEXISTS</b> <b>&lt;token&gt;.ISNULL</b>	Displays 1 (one) if the token value IS NULL, 0 otherwise.
<b>&lt;token&gt;.ISNUMBER</b>	Displays 1 (one) if the token value is entirely numeric (with no whitespace, commas or other characters), 0 otherwise. Tests whether the SQL TO_NUMBER function returns a value without errors. NULL strings qualify as numeric.
<b>&lt;token&gt;.EVAL</b>	Evaluates the token contents according to the rules of OML Logic. Returns 1 (one) if the token evaluates to a logical TRUE, 0 otherwise.
<b>&lt;token&gt;.ROWS</b>	Returns the number of rows of text in the token, as defined by the "new line" (CHR(10)) character. If you want to specify a maximum character length for each row, use the #ROWS[#] token function instead. Useful for dynamically sizing the height of an HTML textarea (e.g. <textarea rows="#token.ROWS#">). Returns 1 (one) if the token is NULL.
<b>&lt;token&gt;.COLS</b>	Returns the maximum number of columns of text in the token between "new line" (CHR(10)) characters. Useful for dynamically sizing the width of an HTML textarea (e.g. <textarea cols="#token.COLS#">). Returns NULL if the token is NULL.

# Orbit 5.0 OML Quick Reference

## OML - Orbit Markup Language

Token	Description
<b>&lt;token&gt;.NOT</b>	Returns 1 (one) if the token contains an OML Logic FALSE literal, such as 'N', 'F', 'FALSE', zero or NULL. Returns 0 (zero) if the token contains an OML Logic TRUE literal, such as 'Y', 'T', 'TRUE', or any positive or negative number. Returns NULL otherwise.
<b>&lt;token&gt;.CACHE</b>	Forces the caching of the token by setting the token's cache flag to 1. Returns the unchanged token value.
<b>&lt;token&gt;.UNCACHE</b>	Forces the un-caching of the token by setting the token's cache flag to 0. Returns the unchanged token value.
<b>&lt;token&gt;.NOTAGS</b>	Substitutes '<', '>', and '&' in <token> with '&lt;', '&gt;', and '&amp;', respectively, to prevent the characters from being confused with HTML tags (i.e. NOTAGS). This can be used to display preformatted or raw text within the <pre> </pre> tags. To also prevent token parsing, use NOPARSE as well.
<b>&lt;token&gt;.NOPARSE</b>	Escapes OML characters in <token> to display raw OML source code without parsing tokens. Replaces the character # with &#035; which is its HTML Escape Sequence.
<b>&lt;token&gt;.ONEPARSE</b>	Same as .NOPARSE, but replaces the character # with CHR(29), a single character. This is useful with token functions that depend on token length (e.g. #ROWS[#token.ONEPARSE#][10]#), since .NOPARSE increases the length of the token value. Note: this modifier should not be used for display purposes since CHR(29) is not represented.
<b>&lt;token&gt;.NOQUOTE</b>	Replaces the character " with &quot; which is its HTML Escape Sequence. This is required if the token is used within an HTML tag's value property (e.g. <input type="text" value="#token.NOQUOTE#">).
<b>&lt;token&gt;.UNTAG</b>	Performs the reverse functionality of .NOTAGS, replacing '&lt;' with '<', '&gt;' with '>' and '&amp;' with '&'.
<b>&lt;token&gt;.UNPARSE</b>	Performs the reverse functionality of .NOPARSE, replacing '&#035;' with '#'. Regular token substitution may follow if the .UNPARSE results contain OML syntax.
<b>&lt;token&gt;.UNRAW</b>	Performs the reverse functionality of .RAW, replacing '&lt;' with '<', '&gt;' with '>', '&amp;' with '&', '&#035;' with '#' and '&quot;' with '"'. Regular token substitution may follow if the .UNRAW results contain OML syntax.
<b>&lt;token&gt;.NOHTML</b>	Removes all text enclosed in HTML delimiters < and > from the token's contents. This is useful for displaying a token's contents in a context where HTML tags are unwanted or unnecessary (e.g. if #token# contained "The answer is <font color="blue">10</font>." then #token.NOHTML# would display "The answer is 10.").
<b>&lt;token&gt;.RAW2</b>	Same as .RAW except that it only replaces the characters < > " with HTML Escape Sequences, but doesn't replace & or #. This is used in multi-currency applications for displaying multi-byte characters correctly.
<b>&lt;token&gt;.QUOTE</b>	Escapes each single quote character (') by replacing it with two single quotes (") for use in SQL queries.
<b>&lt;token&gt;.JSQUOTE</b>	Escapes single quotes ('), double quotes (") and carriage returns with JavaScript style syntax (i.e., \', \", \r and \n). Supports UNIX, PC, and Mac formats for carriage returns.
<b>&lt;token&gt;.BR</b>	Displays <token> with all carriage returns converted to  . This is useful in table data cells when you want to show the line breaks in a text string. Supports UNIX, PC, and Mac formats for carriage returns. Use along with .NOWRAP to preserve token line-wrapping exactly.
<b>&lt;token&gt;.NOWRAP</b>	Displays <token> with all spaces converted to &nbsp;. This is useful in web pages to prevent the browser from wrapping lines at places where your token's contents do not already wrap. Use along with .BR to preserve token line-wrapping exactly.
<b>&lt;token&gt;.ROUND</b>	Displays the numeric equivalent of the token value, rounded to 0 decimal places.
<b>&lt;token&gt;.ROUND2</b>	Displays the numeric equivalent of the token value, rounded to 2 decimal places. This token is padded with zeros in order to fill in two decimal places.
<b>&lt;token&gt;.ROUND1</b>	Displays the numeric equivalent of the token value, rounded to 1 decimal place. This token is padded with zeros in order to fill in one decimal place.
<b>&lt;token&gt;.TRUNC</b>	Displays the numeric equivalent of the token value, truncated to zero decimal places.
<b>&lt;token&gt;.ABS</b>	Displays the absolute value of the numeric equivalent of the token value.
<b>&lt;token&gt;.CEIL</b>	Displays the numeric ceiling of token, the smallest integer greater than or equal to the value.
<b>&lt;token&gt;.FLOOR</b>	Displays the numeric floor of token, the largest integer equal to or less than the value.
<b>&lt;token&gt;.PERCENT</b>	Formats the numeric token as a percentage, i.e., multiplied by 100, rounded to no more than 4 decimal places, with trailing percentage sign (%).

# Orbit 5.0 OML Quick Reference

## OML - Orbit Markup Language

Token	Description
<b>Token Assignment</b>	
<b>#&lt;token&gt;=&lt;value&gt;]#</b>	OML Token Assignment is used to dynamically assign the contents of <value> to <token>. If <token> doesn't exist, it will be created as a Local Token and its value will persist throughout the remainder of this Orbit Page's parsing. If <token> does exist, its value will be overwritten. The effect of the assignment is wholly within the Orbit token cache; a token assignment statement itself parses to NULL (i.e., the empty string) and displays nothing on your Orbit Page. You can assign any text you like to tokens, including OML syntax such as tokens, token modifiers, token functions or other token assignments. Note, however, that token assignment doesn't parse the <value> that is assigned. You can force immediate parsing of <value> by using the #PARSE[]# token function.
<b>Token Functions</b>	
<b>#&lt;function&gt;&lt;arguments&gt;]#</b> <b>#&lt;token&gt;=&lt;function&gt;&lt;arguments&gt;]#</b>	Token Functions may appear on their own or as part of a Token Assignment. If on their own, function results are displayed within your page. If used as part of an assignment, function results are evaluated immediately (unlike Token Assignment without functions) and are dynamically assigned to the named token. The arguments of conditional functions are evaluated only as needed so, for example, the <then> and <else> arguments of the #IF[]# function are never both parsed in one function call. Token Function arguments can contain hardcoded text or OML syntax such as tokens, token modifiers, token assignments or nested token functions. Multiple arguments to a Token Function are specified by repeating the brackets: [arg1][arg2]...[argN]. Token Functions are listed in the order of their evaluation; the most commonly used functions are evaluated first for optimal performance.
<b>#&lt;token&gt;=PARSE&lt;expression&gt;]#</b>	Parses any OML syntax within <text> and assigns the results to <token>. This is in contrast to direct Token Assignment where the <text> to be assigned is unparsed. For example, #x=PARSE[#y]# parses the contents of #y# and assigns the result to #x#, whereas #x=#y]# merely assigns the literal text "#y#" to token #x#. Note that #PARSE[]# without assignment to a <token> has no effect (i.e., #PARSE[#y]# is the same thing as #y#).
<b>#CACHE&lt;name1&gt;[&lt;name1&gt;] [&lt;name1&gt;]...[&lt;nameN&gt;]#</b>	Forces the caching of the tokens with names <name1> through <nameN> by setting their cache flags to 1. The token names should not be enclosed with #-signs. These named tokens need not exist at the time of caching. #CACHE[]# returns NULL.
<b>#UNCACHE&lt;name1&gt;[&lt;name1&gt;] [&lt;name1&gt;]...[&lt;nameN&gt;]#</b>	Forces the UN-caching of the tokens with names <name1> through <nameN> by setting their cache flags to 1. The token names should not be enclosed with #-signs. These named tokens need not exist at the time of un-caching. #UNCACHE[]# returns NULL.
<b>#SEC&lt;section_code&gt; [&lt;SP1&gt;][&lt;SP2&gt;]...[&lt;SP10&gt;]#</b> <b>#SECTION&lt;section_code&gt; [&lt;SP1&gt;][&lt;SP2&gt;]...[&lt;SP10&gt;]#</b> <b>#FUNCTION&lt;section_code&gt; [&lt;SP1&gt;][&lt;SP2&gt;]...[&lt;SP10&gt;]#</b>	Parses and returns the output of the Orbit Section identified by <section_code>, which must be in the same Application Version as the object in which the function call appears. This may be useful if you wish to avoid breaking a unit of code down into multiple sections to accommodate a Repeating Section. As with other token functions, the output of #SECTION[]# may be assigned to a token. If any <SPx> arguments are defined, their text will be assigned (unparsed) to Local Tokens #SP1# through #SP10# for the duration of parsing this Orbit Section only. These tokens will revert to their former values when parsing of the section is complete. #FUNCTION[]# is the same as #SECTION[]# except that SPx tokens are parsed <i>before</i> assignment.
<b>#DROPDOWN&lt;section_code&gt; [&lt;option_value&gt;][&lt;option_text&gt;] [&lt;select_class&gt;][&lt;no_add_flag&gt;]#</b>	Same as the #SELECTED[]# function, except that the first argument is the name of an Orbit Section to be parsed into a list of options instead of the list of options itself. See #SELECTED[]# for more information. This is merely a shortcut for the equivalent syntax: #SELECTED[#SECTION<section_code>]#[<option_value>][<option_text>][<select_flag>][<no_add_flag>]#
<b>#STOP&lt;expression&gt;]#</b>	Stops the processing of the current page at the location of the #STOP[]# function if <expression> evaluates to TRUE according to the rules of OML Logic. Any remaining text in the section or page will NOT be displayed. This is useful to stop an application page from displaying for security reasons.
<b>#DEBUG&lt;raw_text_flag&gt;]#</b>	Displays the contents of the internal Token table for all currently defined tokens, and also shows the CGI environment variables. By default #DEBUG[]# shows the tokens and values in an HTML table format. If <raw_text_flag> parses to 1 (one), then the tokens and values are displayed without decoration in raw format. OML Logic expressions are <i>not</i> used in this argument.
<b>#IF&lt;expression&gt;][&lt;then&gt;][&lt;else&gt;]#</b>	Processes an IF-THEN-ELSE expression. If <expression> evaluates to TRUE according to the rules of OML Logic, then the <then> text is parsed and returned, otherwise the <else> text (which is optional) is parsed and returned. See the OML Logic section for more details on OML expressions.
<b>#EVAL&lt;expression&gt;][&lt;then&gt;][&lt;else&gt;]#</b>	If <expression> evaluates to TRUE according to the rules of OML Logic, then the <then> text is parsed and returned, otherwise the <else> text (which is optional) is parsed and returned. If neither <then> nor <else> arguments are specified, #EVAL[]# returns 1 (one) if <expression> evaluates to TRUE and 0 otherwise. See the OML Logic section for more details on OML expressions.
<b>#EQUAL&lt;val1&gt;][&lt;val2&gt;][&lt;then&gt;][&lt;else&gt;]#</b>	If <val1> is equal to <val2>, then the <then> text is parsed and returned, otherwise the <else> text (which is optional) is parsed and returned. If neither <then> nor <else> arguments are specified, #EQUAL[]# returns 1 (one) if <val1> is equal to <val2> and 0 otherwise.
<b>#NOTEQUAL&lt;val1&gt;][&lt;val2&gt;] [&lt;then&gt;][&lt;else&gt;]#</b>	If <val1> is NOT equal to <val2>, then the <then> text is parsed and returned, otherwise the <else> text (which is optional) is parsed and returned. If neither <then> nor <else> arguments are specified, #NOTEQUAL[]# returns 1 (one) if <val1> is NOT equal to <val2> and 0 otherwise.
<b>#LIKE&lt;val1&gt;][&lt;val2&gt;][&lt;then&gt;][&lt;else&gt;]#</b>	If <val1> is LIKE <val2> then the <then> text is parsed and returned, otherwise the <else> text (which is optional) is parsed and returned. If neither <then> nor <else> arguments are specified, #LIKE[]# returns 1 (one) if <val1> is LIKE <val2> and 0 otherwise. Note that the LIKE operator is tested in both directions, so you are free to use SQL wildcard characters % and _ in either the <val1> or the <val2> arguments.
<b>#EXISTS&lt;text&gt;][&lt;then&gt;][&lt;else&gt;]#</b> <b>#ISNOTNULL&lt;text&gt;][&lt;then&gt;][&lt;else&gt;]#</b> <b>#NOTNULL&lt;text&gt;][&lt;then&gt;][&lt;else&gt;]#</b>	If <text> does NOT parse to NULL, then the <then> text is parsed and returned, otherwise the <else> text (which is optional) is parsed and returned. If neither <then> nor <else> arguments are specified, #EXISTS[]# returns 1 (one) if <text> does NOT parse to NULL and 0 otherwise. #EXISTS[]#, #ISNOTNULL[]# and #NOTNULL[]# are alternative names for the same token function.
<b>#NOTEXISTS&lt;text&gt;][&lt;then&gt;][&lt;else&gt;]#</b> <b>#ISNULL&lt;text&gt;][&lt;then&gt;][&lt;else&gt;]#</b> <b>#NULL&lt;text&gt;][&lt;then&gt;][&lt;else&gt;]#</b>	If <text> parses to NULL, then the <then> text is parsed and returned, otherwise the <else> text (which is optional) is parsed and returned. If neither <then> nor <else> arguments are specified, #NOTEXISTS[]# returns 1 (one) if <text> parses to NULL and 0 otherwise. #NOTEXISTS[]#, #ISNULL[]# and #NULL[]# are alternative names for the same token function.
<b>#GT&lt;val1&gt;][&lt;val2&gt;][&lt;then&gt;][&lt;else&gt;]#</b> <b>#GREATERTHAN&lt;val1&gt;][&lt;val2&gt;] [&lt;then&gt;][&lt;else&gt;]#</b>	If <val1> is greater than <val2> then the <then> text is parsed and returned, otherwise the <else> text (which is optional) is parsed and returned. If neither <then> nor <else> arguments are specified, #GT[]# returns 1 (one) if <val1> is greater than <val2> and 0 otherwise. Note that if both arguments are numbers, a numeric comparison will be done; otherwise a text-based comparison will be done. #GT[]# and #GREATERTHAN[]# are alternative names for the same token function.

# Orbit 5.0 OML Quick Reference

## OML - Orbit Markup Language

Token	Description
<b>#LT[&lt;val1&gt;][&lt;val2&gt;][&lt;then&gt;][&lt;else&gt;]#</b> <b>#LESSTHAN[&lt;val1&gt;][&lt;val2&gt;][&lt;then&gt;][&lt;else&gt;]#</b>	If <val1> is less than <val2> then the <then> text is parsed and returned, otherwise the <else> text (which is optional) is parsed and returned. If neither <then> nor <else> arguments are specified, #LT[# returns 1 (one) if <val1> is less than <val2> and 0 otherwise. Note that if both arguments are numbers, a numeric comparison will be done; otherwise a text-based comparison will be done. #LT[# and #LESSTHAN[# are alternative names for the same token function.
<b>#GE[&lt;val1&gt;][&lt;val2&gt;][&lt;then&gt;][&lt;else&gt;]#</b> <b>#GREATEREQUAL[&lt;val1&gt;][&lt;val2&gt;][&lt;then&gt;][&lt;else&gt;]#</b>	If <val1> is greater than or equal to <val2> then the <then> text is parsed and returned, otherwise the <else> text (which is optional) is parsed and returned. If neither <then> nor <else> arguments are specified, #GE[# returns 1 (one) if <val1> is greater than or equal to <val2> and 0 otherwise. Note that if both arguments are numbers, a numeric comparison will be done; otherwise a text-based comparison will be done. #GE[# and #GREATEREQUAL[# are alternative names for the same token function.
<b>#LE[&lt;val1&gt;][&lt;val2&gt;][&lt;then&gt;][&lt;else&gt;]#</b> <b>#LESSEQUAL[&lt;val1&gt;][&lt;val2&gt;][&lt;then&gt;][&lt;else&gt;]#</b>	If <val1> is less than or equal to <val2> then the <then> text is parsed and returned, otherwise the <else> text (which is optional) is parsed and returned. If neither <then> nor <else> arguments are specified, #LE[# returns 1 (one) if <val1> is less than or equal to <val2> and 0 otherwise. Note that if both arguments are numbers, a numeric comparison will be done; otherwise a text-based comparison will be done. #LE[# and #LESSEQUAL[# are alternative names for the same token function.
<b>#BETWEEN[&lt;text&gt;][&lt;fromVal&gt;][&lt;toVal&gt;][&lt;then&gt;][&lt;else&gt;]#</b>	If <text> is between <fromVal> and <toVal> then the <then> text is parsed and returned, otherwise the <else> text (which is optional) is parsed and returned. If neither <then> nor <else> arguments are specified, #BETWEEN[# returns 1 (one) if <text> is between <fromVal> and <toVal> and 0 otherwise. Note that if all arguments are numbers, a numeric comparison will be done; otherwise a text-based comparison will be done.
<b>#DECODE[&lt;text&gt;][&lt;if&gt;][&lt;then&gt;][&lt;if&gt;][&lt;then&gt;]...[&lt;else&gt;]#</b>	Tests if <text> is EQUAL to any of the <if> arguments, and parses and returns the associated <then> argument if a match is found. The <else> argument, if specified, is parsed and returned if all comparisons fail. This is similar to the SQL DECODE function.
<b>#DECODELIKE[&lt;text&gt;][&lt;if&gt;][&lt;then&gt;][&lt;if&gt;][&lt;then&gt;]...[&lt;else&gt;]#</b>	Tests if <text> is LIKE any of the <if> arguments, and parses and returns the associated <then> argument if a match is found. The <else> argument, if specified, is parsed and returned if all comparisons fail. Note that the LIKE operator is tested in both directions, so you are free to use SQL wildcard characters % and _ in either the <text> argument or in the <if> arguments.
<b>#AND[&lt;expr1&gt;][&lt;expr2&gt;]...[&lt;exprN&gt;]#</b>	Returns 1 if ALL of the expressions <expr1> through <exprN> evaluate to TRUE according to the rules of OML Logic, 0 otherwise. Evaluation of arguments will only continue until an argument that evaluates to FALSE is found or all arguments are evaluated to TRUE.
<b>#OR[&lt;expr1&gt;][&lt;expr2&gt;]...[&lt;exprN&gt;]#</b>	Returns 1 if ANY of the expressions <expr1> through <exprN> evaluate to TRUE according to the rules of OML Logic, 0 otherwise. Evaluation of arguments will only continue until an argument that evaluates to TRUE is found or all arguments are evaluated to FALSE.
<b>#IN[&lt;text&gt;][&lt;val1&gt;]...[&lt;valN&gt;]#</b>	Returns 1 (one) if <text> is EQUAL to any of the <valX> arguments, 0 otherwise. This is similar to the SQL IN operator; an unlimited number of arguments may be specified.
<b>#NOTIN[&lt;text&gt;][&lt;val1&gt;]...[&lt;valN&gt;]#</b>	Returns 0 if <text> is EQUAL to any of the <valX> arguments, 1 (one) otherwise. This is the logical opposite of the #IN[# token function.
<b>#INLIKE[&lt;text&gt;][&lt;val1&gt;]...[&lt;valN&gt;]#</b>	Returns 1 (one) if <text> is LIKE any of the <valX> arguments, 0 otherwise. Note that the LIKE operator is tested in both directions, so you are free to use SQL wildcard characters % and _ in either the <text> argument or in the <valX> arguments. This is a combination of the SQL IN and LIKE operators.
<b>#NOTINLIKE[&lt;text&gt;][&lt;val1&gt;]...[&lt;valN&gt;]#</b>	Returns 0 if <text> is LIKE any of the <valX> arguments, 1 (one) otherwise. This is the logical opposite of the #INLIKE[# token function.
<b>#SELECTED[&lt;select/option_text&gt;][&lt;option_value&gt;][&lt;option_text&gt;][&lt;select_class&gt;][&lt;no_add_flag&gt;]#</b>	Parses and returns the text of <select/option_text>, adjusted if possible to show the specified option as SELECTED. This function is used specifically for <SELECT> lists in HTML; it searches for "coption_value" in the <select/option_text> and replaces it with "<option_value> SELECTED". The <select/option_text> may contain a list of <OPTION> tags, the enclosing <SELECT> tags, or both, or neither. If the <option_value> cannot be found, a new <OPTION> will be added to the beginning of the list of options with the <option_text>, if specified, as its text. However, the missing option will not be added to the list if the <no_add_flag> is passed as 1 (one). If the optional <select_class> argument is specified, it will be added into the <SELECT> tag if it is present in the <select/option_text>; this can be useful for setting the style class of the dropdown list. The #SELECTED[# function allows you to define an entire dropdown list once, and use it on a multiple record update page without requerying the list from the database.
<b>#HELP[&lt;text&gt;][&lt;popup_text&gt;][&lt;required_flag&gt;]#</b>	Displays <text> with an underline and mouseover <popup_text>, as well as showing <popup_text> in the window status bar. Special characters are escaped correctly for JavaScript and HTML (" ' CR), so .JSQUOTE should not be used. If the <required_flag> argument is 1, the help_popup_req class is used (which is generally set to show the text in blue), otherwise the help_popup class is used. The return value looks like: <a href="javascript:void(0);" class="help_popup" onMouseOver="window.status='popup_text'; return true;" onMouseOut="window.status="; return true;" title="popup_text">text</a>
<b>#HELPOVER[&lt;popup_text&gt;]#</b>	Returns the HTML anchor code for displaying <popup_text> in a mouseover as well as in the window status bar. Special characters are escaped correctly for JavaScript and HTML (" ' CR), so .JSQUOTE should not be used. Use #HELP[# when you want mouseovers for plain text; use #HELPOVER[# when you want mouseovers for existing HTML links. The return value looks like: onMouseOver="window.status='popup_text'; return true;" onMouseOut="window.status="; return true;" title="popup_text"
<b>#SQL[&lt;expression&gt;][&lt;show_errors&gt;]#</b> <b>#MATH[&lt;expression&gt;][&lt;show_errors&gt;]#</b>	Executes a SQL query passing in <expression> to "SELECT <expression> FROM DUAL" and returns the result, converted to VARCHAR2 if necessary. If there is an error then NULL is returned, unless <show_errors> parses to an OML "TRUE" literal such as '1', '1', 'Y', 'T' or 'TRUE', in which case the error message text from SQLERRM is returned. #MATH[# and #SQL[# are alternative names for the same token function.
<b>#SELECT[&lt;sql_query&gt;][&lt;show_errors&gt;]#</b>	Executes a SQL query and returns the value of the first column from the first row, converted to VARCHAR2 if necessary (e.g. #SELECT[select code from dem_items where item_id = #P1.0##]). Other columns and rows are ignored. If there is an error then NULL is returned, unless <show_errors> parses to an OML "TRUE" literal such as '1', '1', 'Y', 'T' or 'TRUE', in which case the error message text from SQLERRM is returned.
<b>#EXEC[&lt;pl/sql_block&gt;]#</b>	Dynamically executes a PL/SQL block, which may contain procedures, functions, and any PL/SQL code. #EXEC[# returns NULL unless an error is encountered, in which case it returns the error message text from SQLERRM. #EXEC[# will enclose your PL/SQL block with BEGIN and END statements so these can be omitted from your code.
<b>#EXECIF[&lt;expression&gt;][&lt;pl/sql_block&gt;]#</b>	Same as #EXEC[# except that the <pl/sql_block> is only executed if <expression> evaluates to TRUE according to the rules of OML Logic. See the OML Logic section for more details on OML expressions.
<b>#CLOBTEXT[&lt;table_name&gt;][&lt;clob_column_name&gt;][&lt;rowid&gt;]#</b>	Returns the text of a CLOB (Character Large Object), given the database table and CLOB column name and the ROWID of the row from which it should be retrieved. This function performs a dynamic SQL call to the database. Note that you must have already retrieved the database ROWID to make use of this function.
<b>#SYSDATE[&lt;format&gt;]#</b>	Returns the system date in the specified format. Format masks are the same as used in the SQL TO_CHAR function. If <format> is NULL, the system date will be returned in default format (usually DD-MON-YY).

# Orbit 5.0 OML Quick Reference

## OML - Orbit Markup Language

Token	Description
<b>#TRANSACTION_ID[]#</b>	Returns a unique Transaction ID for use in <FORM> sections by calling cor\$transaction_util.get_transaction_id. This function takes no arguments but the [] is required to identify it as a token function.
<b>#NVL[&lt;val1&gt;][&lt;val2&gt;]...[&lt;valN&gt;]#</b>	Considers each argument in turn and returns the parsed text of the first argument that does not parse to NULL. This is similar to the SQL NVL function but is extended in OML to an unlimited number of arguments.
<b>#COMMLIST[&lt;val1&gt;][&lt;val2&gt;]...[&lt;valN&gt;]#</b>	Returns a comma-separated list of all <valX> arguments which do NOT parse to NULL. An unlimited number of arguments may be specified, and all arguments are parsed.
<b>#REPEAT[&lt;times&gt;][&lt;text&gt;]#</b>	Repeats <text> by <times> number of times, and then parses the result and returns it. Note that this parsing order means that any context-sensitive OML inside <text> will be parsed <times> number of times, instead of being parsed once and then repeated. Care should be taken not to exceed 32K characters even before parsing, to avoid overrunning the #REPEAT[]# buffer.
<b>#WHILE[&lt;expression&gt;][&lt;text&gt;][&lt;maxtimes&gt;]#</b>	Repeats <text> by 0 to <maxtimes> times, as long as <expression> evaluates to TRUE according to the rules of OML Logic. Context-sensitive OML inside <text> will be re-parsed each time through the loop. The <maxtimes> argument is optional and defaults to 5000 if unspecified. See the OML Logic section for more details on OML expressions.
<b>#REPLACE[&lt;text&gt;][&lt;find&gt;][&lt;replace&gt;][&lt;all_flag&gt;]#</b>	Replaces any occurrence of <find> with <replace> in the given <text> input. If <replace> is not specified then occurrences of <find> will simply be removed (i.e., replaced with NULL). If <all_flag> is set to 1, the replace will be called iteratively until no more <find> occurrences are found, up to 50 iterations. This is useful for replacing instances of multiple spaces with single spaces.
<b>#IREPLACE[&lt;text&gt;][&lt;find&gt;][&lt;replace&gt;]#</b>	Similar to #REPLACE[]#, but performs case-insensitive comparisons on <find> and <replace>. Also, #IREPLACE[]# does not implement an <all_flag> in the manner of #REPLACE[]#.
<b>#TRANSLATE[&lt;text&gt;][&lt;search_set&gt;][&lt;replace_set&gt;]#</b>	Returns a character-by-character translation of <text>, with the nth character in the <search_set> translated to the nth character in the <replace_set>. This is the same as the SQL TRANSLATE function.
<b>#SUBSTR[&lt;text&gt;][&lt;begin&gt;][&lt;bytes&gt;]#</b>	Returns the substring of <text> beginning at position <begin> with a length of <bytes> characters. If <bytes> is not specified then the substring will run from the starting position to the end of the string. If <begin> is a negative number then it will be taken as an offset from the end of the string. This is the same as the SQL SUBSTR function.
<b>#INSTR[&lt;text&gt;][&lt;find&gt;][&lt;begin&gt;][&lt;occurrence&gt;]#</b>	Returns the position of the <occurrence> occurrence of <find> text in the <text> string, beginning at position <begin>. Arguments <begin> and <occurrence> both default to 1 if unspecified. If <begin> is a negative number then it will begin at this offset from the end of the string and search backwards towards the beginning. This is the same as the SQL INSTR function.
<b>#FORMAT[&lt;number&gt;][&lt;format&gt;]#</b> <b>#FORMAT[&lt;date&gt;][&lt;in_format&gt;][&lt;out_format&gt;]#</b>	Returns formatted numbers and dates based on the format masks provided. #FORMAT[]# with two arguments is interpreted as numeric formatting; if a third argument is present, #FORMAT[]# attempts date formatting instead. Format masks are the same as used in the SQL TO_CHAR function. Numeric formatting converts the <number> text to a number (with no format mask) and then back to a character using <out_format> mask. Date formatting converts the <date> text to a date using <in_format> mask and then back to a character using <out_format> mask.
<b>#CHR[&lt;n&gt;]#</b>	Returns the character with ASCII sequence number equal to <n>. For example, #CHR[9]# is tab and #CHR[10]# is new line. This is the same as the SQL CHR function.
<b>#RPAD[&lt;text&gt;][&lt;length&gt;][&lt;char&gt;]#</b>	Right pads <text> to length <length> with the character set <char>. Argument <length> is required; <char> is optional and defaults to space. If <length> is less than the length of <text> then it will be truncated. This is the same as the SQL RPAD function.
<b>#LPAD[&lt;text&gt;][&lt;length&gt;][&lt;char&gt;]#</b>	Left pads <text> to length <length> with the character set <char>. Argument <length> is required; <char> is optional and defaults to space. If <length> is less than the length of <text> then it will be truncated. This is the same as the SQL LPAD function.
<b>#RTRIM[&lt;text&gt;][&lt;char&gt;]#</b>	Removes any trailing <char> characters from the right end of <text>. Argument <char> is optional and defaults to space. This is the same as the SQL RTRIM function.
<b>#LTRIM[&lt;text&gt;][&lt;char&gt;]#</b>	Removes any leading <char> characters from the left end of <text>. Argument <char> is optional and defaults to space. This is the same as the SQL LTRIM function.
<b>#TRIM[&lt;text&gt;][&lt;char&gt;]#</b>	Removes both leading and trailing <char> characters from <text>. Argument <char> is optional and defaults to space. This is the same as the SQL TRIM function (Oracle8i and higher).
<b>#ROWS[&lt;text&gt;][&lt;max_linesize&gt;][&lt;min_rows&gt;][&lt;max_rows&gt;]#</b>	Returns the number of rows of text in <text>, as defined by the "new line" (CHR(10)) character. Useful for dynamically sizing the height of an HTML textarea (e.g. <textarea rows="#ROWS[#token#][80][5][50]#">). If <max_linesize> is specified, #ROWS[]# will wrap lines longer than <max_linesize>, thereby increasing the number of rows it returns. Orbit mimics IE-style textarea word-wrapping with its left-breaking (e.g., tab, left-parenthesis, etc.) and right-breaking (e.g., space, question mark, right-parenthesis, etc.) characters. If <min_rows> and <max_rows> are specified, #ROWS[]# will adjust its return value to be within these values.
<b>#WRAP[&lt;text&gt;][&lt;maxlength&gt;][&lt;wb_chars&gt;][&lt;lb_chars&gt;]#</b>	Returns <text> wrapped to no more than <maxlength> characters per line. Calls orb\$util.wrap_text, which attempts to wrap text at word boundaries -- after characters found in <wb_chars> -- but will break in the middle of words if necessary. Line-breaking characters are as found in <lb_chars>. Optional arguments: <maxlength> defaults to 80, <wb_chars> defaults to space and tab, and <lb_chars> defaults to CHR(10) and CHR(13).
<b>#WRAPTOKEN[&lt;name&gt;][&lt;maxlength&gt;][&lt;wb_chars&gt;][&lt;lb_chars&gt;]#</b>	Same as the #WRAP[]# function, except the first argument is the name of the token containing the text to be wrapped, not the text itself. The token name should not be enclosed with #-signs. Returns wrapped text as with #WRAP[]#.
<b>#STRIPTEXT[&lt;text&gt;][&lt;begin_text&gt;][&lt;end_text&gt;][&lt;replace_text&gt;][&lt;match_pairs&gt;]#</b>	Iteratively strips text strings out of <text> based on the <begin_text> and <end_text> (for example, removing comments such as <!-- OML Comment -->). Returns the resulting string without the text in between begin / end text strings. If <replace_text> is specified, replaces the match with this text. Use <match_pairs> = 1 to strip nested pairs, not just the first occurrence of the end text. All arguments but <text> are optional; begin / end default to <!-- / --> respectively, replace text to NULL, and match pairs to 0.
<b>#APPEND[&lt;text1&gt;][&lt;text2&gt;][&lt;linebreak&gt;]#</b>	Appends <text2> on the end of <text1>, separated by the <linebreak> text if specified and <text1> is not blank. Argument <linebreak> defaults to NULL. For example, #APPEND[#line_1#][#line_2#][ ]#.
<b>#&lt;token&gt;= [&lt;text&gt;][&lt;linebreak&gt;]#</b>	Self-referential version of #APPEND[]# -- appends <text> on the end of <token>, separated by the <linebreak> text if specified and <token> is not blank, and assigns the result to <token>. Argument <linebreak> defaults to NULL. If <token> is not specified, function returns <text>.
<b>#MIN[&lt;a&gt;][&lt;b&gt;]...[&lt;n&gt;]#</b>	Returns the minimum value from the list of arguments. NULL arguments are ignored, so consider using the .0 modifier on tokens if NULL values should be treated as 0 (zero). If all non-NULL arguments are numeric, a numeric comparison will be done; otherwise a text-based comparison will be done.
<b>#MAX[&lt;a&gt;][&lt;b&gt;]...[&lt;n&gt;]#</b>	Returns the maximum value from the list of arguments. NULL arguments are ignored, so consider using the .0 modifier on tokens if NULL values should be treated as 0 (zero). If all non-NULL arguments are numeric, a numeric comparison will be done; otherwise a text-based comparison will be done.
<b>#ADD[&lt;a&gt;][&lt;b&gt;]...[&lt;n&gt;]#</b>	Returns <a> with <b> through <n> added to it. An unlimited number of parameters may be specified. Non-numeric or NULL arguments are interpreted as 0.
<b>#&lt;token&gt;=+[&lt;a&gt;][&lt;b&gt;]...[&lt;n&gt;]#</b>	Self-referential version of #ADD[]# -- calculates <token> with <a> through <n> added to it, and assigns the result to <token>. If <token> is not specified, function returns the value of 0 with <a> through <n> added to it.

# Orbit 5.0 OML Quick Reference

## OML - Orbit Markup Language

Token	Description
<b>#SUB[&lt;a&gt;][&lt;b&gt;]...[&lt;n&gt;]#</b> <b>#SUBTRACT[&lt;a&gt;][&lt;b&gt;]...[&lt;n&gt;]#</b>	Returns <a> with <b> through <n> subtracted from it. An unlimited number of parameters may be specified. Non-numeric or NULL arguments are interpreted as 0.
<b>#&lt;token&gt;=[&lt;a&gt;][&lt;b&gt;]...[&lt;n&gt;]#</b>	<i>Self-referential</i> version of #SUB[]# -- calculates <token> with <a> through <n> subtracted from it, and assigns the result to <token>. If <token> is not specified, function returns the value of 0 with <a> through <n> subtracted from it.
<b>#MULT[&lt;a&gt;][&lt;b&gt;]...[&lt;n&gt;]#</b> <b>#MULTIPLY[&lt;a&gt;][&lt;b&gt;]...[&lt;n&gt;]#</b>	Returns <a> multiplied by <b> through <n>. An unlimited number of parameters may be specified. Non-numeric or NULL arguments cause the function to return NULL.
<b>#&lt;token&gt;=[&lt;a&gt;][&lt;b&gt;]...[&lt;n&gt;]#</b>	<i>Self-referential</i> version of #MULT[]# -- calculates <token> multiplied by <a> through <n>, and assigns the result to <token>. If <token> is not specified, function returns NULL.
<b>#DIV[&lt;a&gt;][&lt;b&gt;]...[&lt;n&gt;]#</b> <b>#DIVIDE[&lt;a&gt;][&lt;b&gt;]...[&lt;n&gt;]#</b>	Returns <a> divided by <b> through <n>. Non-numeric or NULL arguments cause the function to return NULL. Division by 0 also returns NULL.
<b>#&lt;token&gt;=/[&lt;a&gt;][&lt;b&gt;]...[&lt;n&gt;]#</b>	<i>Self-referential</i> version of #DIV[]# -- calculates <token> divided by <a> through <n>, and assigns the result to <token>. If <token> is not specified, function returns NULL.
<b>#ROUND[&lt;number&gt;][&lt;decimals&gt;]#</b>	Returns the numeric equivalent of <number> rounded to <decimals> decimal places. If <decimals> isn't specified then it defaults to 0.
<b>#TRUNC[&lt;number&gt;][&lt;decimals&gt;]#</b>	Returns the numeric equivalent of <number> truncated to <decimals> decimal places. If <decimals> isn't specified then it defaults to 0.
<b>#SIGN[&lt;number&gt;]#</b>	Returns the sign of the numeric equivalent of <number>; i.e., returns -1 if <number> is less than zero, 0 if <number> is equal to 0, and 1 if <number> is greater than 0. This is the same as the SQL SIGN function.
<b>#ABS[&lt;number&gt;]#</b>	Returns the absolute value of the numeric equivalent of <number>. This is the same as the SQL ABS function.
<b>#MOD[&lt;m&gt;][&lt;n&gt;]#</b>	Returns the remainder of <m> divided by <n>. This is the same as the SQL MOD function, except that #MOD[]# returns NULL if <n> is equal to 0.
<b>#AVG[&lt;a&gt;][&lt;b&gt;]...[&lt;n&gt;]#</b>	Returns the average numeric value from the list of arguments. Non-numeric and NULL arguments are ignored, so #AVG[5][15]# is the same thing as #AVG[5][and][15]# (i.e., 10). Consider using the .0 modifier on tokens if NULL values should be treated as 0 (zero).
OML Comments	
<b>&lt;!-- OML Comment --&gt;</b>	OML Comments look similar to HTML Comments but contain three dashes instead of two. They are removed during parsing and aren't displayed on generated pages at all. They can be used for internal developer comments, or in contexts where HTML Comments aren't supported (such as generated PL/SQL or Plain Text pages). OML Comments may be nested, meaning you can use them to comment-out blocks of code which themselves contain OML Comments. A single OML Comment cannot be defined across more than one section or field.

# Orbit 5.0 OML Quick Reference

## OML - Orbit Markup Language

Token	Description
<b>OML Logic v2</b>	
<p>OML implements a set of rules for determining whether a logical expression evaluates to TRUE or FALSE. This same OML Logic is used for the Section Display Logic fields, the Query Execute Logic field, and the following logical token functions: #IF[#], #EXECIF[#], #EVAL[#], #AND[#], #OR[#] and #STOP[#], as well as in the token modifier .EVAL.</p> <p>OML Logic can be optimized by coding for expressions that satisfy the earlier rules of its logic. For example, NULL strings or recognized strings (1 or 0, TRUE or FALSE, Y or N, etc.) are very quickly interpreted as TRUE or FALSE and so lead to faster parsing of your Orbit Page. PL/SQL expressions require a dynamic SQL call and therefore can take longer to evaluate.</p> <p>The full rules of OML Logic v2 are detailed here. See the Developer's Guide for information on OML Logic v1.</p> <p>* NULL literals default to TRUE. This refers to actual NULL arguments, not OML expressions that may parse to NULL.</p> <p>* OML expressions that parse to '1', '-1', 'Y', 'T' or 'TRUE' evaluate to TRUE. OML expressions that parse to '0', 'N', 'F' or 'FALSE' evaluate to FALSE. This logic is case-insensitive and single-quotes surrounding these values, if present, are ignored.</p> <p>* OML expressions that parse to NULL evaluate to FALSE. As mentioned above, this is not the same thing as NULL literal arguments, which evaluate to TRUE.</p> <p>* Parsed OML expressions that can successfully be interpreted as numbers are evaluated as follows: any positive or negative numbers evaluate to TRUE; zero evaluates to FALSE.</p> <p>* If none of the above rules apply, Orbit attempts to evaluate the OML expression as a PL/SQL expression which must return TRUE or FALSE.</p> <p>* OML Logic v2 assumes that any expression that failed to match all of the preceding rules must be a PL/SQL expression with invalid syntax. In this case, Orbit will write an error message to your Orbit Page and behave otherwise as though the OML expression had evaluated to FALSE.</p>	
<b>Version Control Panel: Security Tokens</b>	
<b>* Defined in Token Table for each Application Version *</b>	
<b>SECURITY_FLAG</b>	Security indicator - if set to 1, security is enabled for the version, 0 or NULL turns security off. If security is disabled with this parameter, this overrides any page level "use_security_flag". NOTE: The LOGON_PAGE token must also be defined for security to be active. The Use Security checkbox also needs to be set for each page that is under security.
<b>ACCESS_SECURITY_FLAG</b>	Defines whether Application and Page Access Security is enabled for a version, as defined in the User Maintenance page. If 1, Access to a page is checked for the User and their assigned Access Groups, and the user is either allowed or denied access accordingly. If 0, access security is not checked. Access Security is only enabled if the SECURITY_FLAG is 1.
<b>LOAD_ACCESS_GROUPS_FLAG</b>	Defines whether Access Groups are loaded for a user when a page is displayed. If 1, all access groups defined for a user will have tokens created in the form of SEC\$<Access_Group_Code>. These can then be used in a page to determine security access.
<b>LOGON_PAGE</b>	Default Logon Page Code for this version. This page is displayed if security is enabled and the user's session has expired or they are not logged on.
<b>ERROR_PAGE</b>	Default Error Page Code for this version. This page is displayed if there is an issue validating security.
<b>SESSION_TIMEOUT</b>	The number of minutes a session can remain active with no activity. This is only used if the SECURITY_FLAG = 1.
<b>Version Control Panel: General Tokens</b>	
<b>* Defined in Token Table for each Application Version *</b>	
<b>CACHE_FLAG</b>	Internal Page Caching Flag - if set to 1, global tokens will be pre-parsed on the page and the page will be stored in the cache. Setting the No Cache Flag on any Page, Section, Query, or Token will disable the cache for that object.
<b>LOG_PAGE_STATISTICS_FLAG</b>	Global flag for a version to indicate whether Page Statistics should be logged. If 1, 2 or 3, any page with the Log Page Statistics checkbox checked will be logged. If 0, no pages will have their statistics logged. Logging levels are 1 (BRIEF logging), 2 (AUDIT TRAIL logging), and 3 (VERBOSE logging) in increasing level of detail.
<b>LOG_QUERY_STATISTICS_FLAG</b>	Global flag for a version to indicate whether Query Statistics should be logged. If 1, 2 or 3, any query with the Log Query Statistics checkbox checked will be logged. If 0, no queries will have their statistics logged. Logging levels are 1 (BRIEF logging), 2 (AUDIT TRAIL logging), and 3 (VERBOSE logging) in increasing level of detail.
<b>SHOW_ENV_FLAG</b>	If set to 1, shows the value of all environment variables and cached tokens if an error occurs on a page.
<b>HOME_PAGE</b>	Default Home Page Code for this version. This page is displayed if Show_Page is called without a valid Page Code.
<b>SERVICE_PAGE</b>	Default Page Code when site is being serviced. This page is displayed if the site is being worked on or upgraded so the user does not see any pages with errors or interfere with the upgrade. No other page can be displayed when this token is enabled. Activate the token to enable. Inactivate the token to prevent this page from being displayed.
<b>COOKIE_NAME</b>	Security cookie name for an application version.
<b>CHECK_ON</b>	Image tag for a checked checkbox. This is used for the .CHECKBOX token modifier if the value of the token is TRUE.
<b>CHECK_OFF</b>	Image tag for an unchecked checkbox. This is used for the .CHECKBOX token modifier if the value of the token is FALSE.
<b>UTL_FILE</b>	The directory where generated HTML files are saved. This overrides the UTL_FILE_DIR parameter defined in the Orbit parameters. This must also be specified as the utl_file_dir parameter in init.ora.
<b>PROMPT_ERROR</b>	A string that appears at the beginning of any Error Message that is displayed (i.e., the contents of the ERROR_TEXT token).
<b>PROMPT_SUCCESS</b>	A string that appears at the beginning of any Success Message that is displayed (i.e., the contents of the SUCCESS_TEXT token).
<b>DATE_MASK</b>	Default Date Mask for the version.
<b>DATE_TIME_MASK</b>	Default Date Time Mask for the version.
<b>CURRENCY</b>	Default currency mask for the version.
<b>Version Control Panel: ROWX / ROWY Tokens</b>	
<b>* Defined in Token Table for each Application Version *</b>	
<b>ROWX1</b> <b>ROWX2</b> <b>ROWX3</b> <b>ROWX4</b> <b>ROWX5</b>	Row formatting tokens to set the ROWX Query Token based on the actual row number fetched from the query. The ROWX Query Token will be set to the value of ROWX1-5 depending on how many of these tokens are defined and the MOD of the actual row number. For example, if only ROWX1 and ROWX2 are defined, then odd-numbered rows will have ROWX set to ROWX1 and even-numbered rows will have ROWX set to ROWX2. ROWX1-5 can be defined as global tokens or in a prior section, such as #ROWX1=[class=lightblue]#.

# Orbit 5.0 OML Quick Reference

## OML - Orbit Markup Language

Token	Description
ROWY1 ROWY2 ROWY3 ROWY4 ROWY5	Secondary row formatting commands in addition to ROWX1-5.
<b>Internal Tokens</b>	
PLSQL_PATH	The URL path up to the Oracle stored procedure. This is the PL/SQL DAD (Database Access Descriptor) and is useful when migrating an application from one machine to another where the DAD may be different. e.g. if the URL is http://www.server.com/pls/dad/Orbit.Show_Page, #PLSQL_PATH# would be replaced by: /pls/dad/
COOKIE	A parameter in Show_Page used for cookie enabled security where the cookie contains the user_ID and session_ID.
P1 through P50	Flexible Page parameters passed in a call to Show_Page. These parameters are frequently passed by defining P1, P2, ... P50 as <input> tags within a form on an HTML page.
P_START_ROW	A parameter in Show_Page used for passing the next start row for Next / Previous type pages. <b>(OBSOLETE -- use START_RECORD instead)</b>
START_RECORD	A parameter in Show_Page used for passing the next start record for Next / Previous type pages.
MAX_RECORDS	A parameter in Show_Page used for passing the max number of records to display for a query.
NEXT_VERSION NEXT_PAGE	Parameters in Show_Page used for passing to an intermediate version / page.
SESSION_ID	A parameter in Show_Page for passing the session ID of the current web session (in absence of cookies).
PRIMARY_KEY_ID	A parameter in Show_Page for passing the unique primary key ID of a record.
ALT_PRIMARY_KEY_ID	A copy of PRIMARY_KEY_ID used when record processing results in errors.
MIME_TYPE	A parameter in Show_Page for displaying the output to a specific application (e.g. application/Excel).
PFROM / PTO PRANGE_FLAG PRANGE_COLUMN PDAYS_BACK PORDER_BY1 PORDER_BY2 PORDER_BY3	Parameters in Show_Page used for passing date ranges. PFrom and PTo contain date values; PRange_Flag is set if PFrom and PTo are both specified and define a date range; PRange_Column is the name of a query's date column to filter according to the specified date range(s); PDays_Back defines another date range that ends with SYSDATE, and may be used instead of or in addition to the PTo / PFrom logic.
PCOMPANY	A parameter in Show_Page for passing the Company ID.
PLANGUAGE	A parameter in Show_Page for passing the Language ID.
PFRAME_FLAG	A parameter in Show_Page used in drawing the Top Menu in "No Frames" mode in Orbit.
PPREVIOUS_URL	A parameter in Show_Page for passing the Previous URL. This is used when bouncing between databases within one application context.
ERROR_TEXT	Set by a call to Orbit.Register_Error or Orbit.Append_Error.
SUCCESS_TEXT	Set by a call to Orbit.Register_Success or Orbit.Append_Success.
MESSAGE_TEXT	Set by a call to Orbit.Register_Message or Orbit.Append_Message.
HEADER_MESSAGE	This is the composite ERROR_TEXT, SUCCESS_TEXT, and MESSAGE_TEXT for simple display in the header of a page. The Global Tokens PROMPT_ERROR and PROMPT_SUCCESS are displayed before the Error and Success messages, respectively.
SQLERRM	Contains the SQLERRM Oracle error text when OTHERS is raised in Orbit. This is used specifically for errors in SQL queries, especially if the ignore errors flag is checked.
SEC\$VIEW_ONLY	This token is set to 1 if the ACCESS_SECURITY_FLAG is set.
<b>Page Tokens</b>	
VERSION_CODE	The Version Code of the page currently being displayed.
PAGE_CODE	The Page Code of the page currently being displayed.
CURRENT_PAGE_CODE	The Page Code of the page that was being displayed before being redirected to the Logon Page. Use this for going to the original page the user clicked on prior to seeing the logon page. This creates continuity in the logon process.
VERSION_ID	The Version ID of the page currently being displayed. This can be passed to Show_Page instead of the Version Code.
PAGE_ID	The Page ID of the page currently being displayed. This can be passed to Show_Page instead of specifying both the Version Code and Page Code.
PAGE_TITLE	The Page Title as defined in the Page Maintenance screen.
PAGE_HEADING	The Page Heading as defined in the Page Maintenance screen.
PAGE_USE_SECURITY_FLAG	The value of the Use Security flag (1 or 0) for the page being displayed. This is useful for determining whether security checks need to be done for the page.
PAGE.TIME PAGE_CODE.TIME	Total Page Execution Time in seconds, up to the point where this token is called. Calling #PAGE.TIME# at different positions in the page will show the time offset from when the page started generating.
<b>Section Tokens</b>	

# Orbit 5.0 OML Quick Reference

## OML - Orbit Markup Language

Token	Description
<b>SECTION_CODE</b>	The Section Code currently being displayed.
<b>LAST_SECTION_CODE</b>	The Section Code that was displayed prior to the current one. Sections displayed with the #SECTION[]# Token Function are not reflected in LAST_SECTION_CODE.
<b>SP1 through SP10</b>	Flexible Section Parameter tokens passed to a section by arguments of the #SECTION[]# Token Function. These tokens exist only during parsing and display of the Orbit Section in question, and are restored to their prior contents (if any) once section parsing is complete.
Query Tokens	
<b>QUERY_CODE</b>	The Query Code of the current or most recently executed Orbit Query.
<b>LAST_QUERY_CODE</b>	The Query Code of the Orbit Query executed before the one identified in QUERY_CODE.
<b>ROWX</b>	Row formatting token set according to the values of the ROWX1-5 tokens, based on the actual row number fetched from the query. See Version Control Panel: ROWX / ROWY Tokens (above) for details.
<b>ROWY</b>	Secondary row formatting token in addition to ROWX. See Version Control Panel: ROWX / ROWY Tokens (above) for details.
<b>ROWNUM</b>	The current sequential row number for a repeating Query.
<b>&lt;query_code&gt;.ROWNUM</b>	The current sequential row number for the specified Orbit Query. This is useful in master / detail queries.
<b>&lt;query_code&gt;.COLCOUNT</b>	Returns the number of columns that were in the specified Orbit Query.
<b>&lt;query_code&gt;.COUNT</b>	The total number of records retrieved by the last execution of the specified Orbit Query.
<b>&lt;query_code&gt;.MAX_ROWS</b>	The Maximum amount of rows to be displayed for the specified Orbit Query. This is defined in the Query Maintenance screen.
<b>&lt;query_code&gt;.START_WITH</b>	The Start With record for the specified Orbit Query. This is defined in the Query Maintenance screen.
<b>&lt;query_code&gt;.NEXT</b>	The next Start With record for the specified Orbit Query. This is set if <query_code>.MAX_ROWS is reached.
<b>&lt;query_code&gt;.PREV</b>	The previous Start With record for the specified Orbit Query. This is set if <query_code>.MAX_ROWS is reached.
<b>&lt;query_code&gt;.TIME</b>	Query Execution Time in seconds for the specified Orbit Query. This includes OML Parsing time as well as Fetch time.
<b>&lt;query_token&gt;.SUM</b>	Sum of the numeric values of the specified Query Token, which must be included in the Sum Field Calculations field of the Query Maintenance Screen.
<b>&lt;query_token&gt;.AVG</b>	Average of the numeric values of the specified Query Token, which must be included in the Average Field Calculations field of the Query Maintenance Screen.
<b>&lt;query_token&gt;.MIN</b>	Minimum value of the specified Query Token, which must be included in the Min Value field of the Query Maintenance Screen. If all values are numeric, a numeric minimum will be used; otherwise a text-valued minimum will be used.
<b>&lt;query_token&gt;.MAX</b>	Maximum value of the specified Query Token, which must be included in the Max Value field of the Query Maintenance Screen. If all values are numeric, a numeric maximum will be used; otherwise a text-valued maximum will be used.
<b>&lt;query_token&gt;.LAST</b>	Previous record's value of the specified Query Token, which is used for determining break points. This is only defined when the token is included in the Breaks Field Calculations field of the Query Maintenance Screen.
<b>&lt;query_token&gt;.CURRENT</b>	Current record's value of the specified Query Token, which is used for determining break points. This is only defined when the token is included in the Breaks Field Calculations field of the Query Maintenance Screen.
User Tokens	
<b>USER\$ID</b>	USER_ID of the Orbit User currently logged in.
<b>USER\$USERNAME</b>	USERNAME of the Orbit User currently logged in.
<b>USER\$FULL_NAME</b>	Full Name of the Orbit User currently logged in. This is specified in the User Maintenance Screen.
<b>USER\$EMPLOYEE_NUMBER</b>	Employee Number of the Orbit User currently logged in. This is specified in the User Maintenance Screen.
<b>USER\$LANGUAGE_ID</b>	LANGUAGE_ID of the Orbit User currently logged in. This is specified in the User Maintenance Screen.
<b>USER\$COMPANY_ID</b>	Default COMPANY_ID of the Orbit User currently logged in.
<b>USER\$DATE_MASK</b>	Date Format of the Orbit User currently logged in. This is set to #DATE_MASK# if none is specified in the User Maintenance Screen. Used in TO_CHAR (e.g. TO_CHAR (created_date, #USER\$DATE_MASK#)).
<b>USER\$DATE_TIME_MASK</b>	Date Time Format of the Orbit User currently logged in. This is set to #DATE_TIME_MASK# if none is specified in the User Maintenance Screen. Used in TO_CHAR (e.g. TO_CHAR (created_date, #USER\$DATE_TIME_MASK#)).
<b>USER\$CURRENCY_MASK</b>	Currency Format of the Orbit User currently logged in. This is set to #CURRENCY# if none is specified in the User Maintenance Screen. Used in TO_CHAR (e.g. TO_CHAR (234.3, #USER\$CURRENCY_MASK#)).
<b>USER\$SESSION_ID</b>	Current unique SESSION_ID of the Orbit User currently logged in.
Environment Tokens	
<b>ENV\$DATABASE</b>	Returns the database name from v\$database, or from global_name if not found.
<b>ENV\$SYSDATE</b>	Returns the current system date formatted as specified in token USER\$DATE_MASK.
<b>ENV\$SYSDATE_TIME</b>	Returns the current system date and time formatted as specified in token USER\$DATE_TIME_MASK.
<b>ENV\$REMOTE_HOST</b>	The hostname of the machine requesting the page.

# Orbit 5.0 OML Quick Reference

## OML - Orbit Markup Language

Token	Description
<b>ENV\$REMOTE_ADDR</b>	The IP Address of the machine requesting the page.
<b>ENV\$REMOTE_USER</b>	The Username on the machine requesting the page.
<b>ENV\$REQUEST_METHOD</b>	The request method (GET / PUT).
<b>ENV\$HTTP_USER_AGENT</b>	The browser requesting the page (e.g. "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0;)").
<b>ENV\$HTTP_HOST</b>	The hostname identified in the URL (e.g. "www.hostname.com").
<b>ENV\$SERVER_NAME</b>	The server's internal hostname (e.g. "hostname.com").
<b>ENV\$SERVER_PORT</b>	The Port Number on the request (e.g. "80").
<b>ENV\$REQUEST_PROTOCOL</b>	The HTTP protocol used (HTTP / HTTPS).
<b>ENV\$HOST</b>	The fully qualified host prefix on the URL (e.g. "http://www.hostname.com" or "http://www.hostname.com:88").
<b>ENV\$IE</b>	Flag indicating whether the browser is Internet Explorer. Set to 1 (one) if it's an IE compatible browser, 0 otherwise.
<b>ENV\$NS</b>	Flag indicating whether the browser is Netscape Navigator. Set to 1 (one) if it's a Netscape compatible browser, 0 otherwise.
<b>ENV\$VERSION</b>	Browser Version from HTTP_USER_AGENT string (e.g. "4.75", "5.5", etc.).
<b>ENV\$EXCEL</b>	Flag indicating whether the current mime type contains Excel. Set to 1 (one) if it contains "excel", 0 otherwise.
<b>ENV\$NOTEXCEL</b>	Flag indicating whether the current mime type does NOT contain Excel. This is the inverse of the ENV\$EXCEL token. Set to 1 (one) if it does NOT contain "excel", 0 otherwise.
<b>ENV\$DOC_TABLE</b>	The database table used by the PL/SQL DAD (Database Access Descriptor) for upload of documents. This is based on the "Document Table" defined in the DAD configuration files.
<b>ENV\$DOC_PATH</b>	The URL path used by the PL/SQL DAD (Database Access Descriptor) for download of documents. This is based on the "Document Access Path" defined in the DAD configuration files.