Oracle® HTML DB

Release Notes

Release 1.6

Part No. B14499-01

November 2004

These *Release Notes* contain important last minute information not included in the Oracle HTML DB documentation. For the most current information, refer to updates of this document, which are located at the following Web site:

`http://www.oracle.com/technology/documentation/`

This document contains these topics:

- Documentation
- New Features
- Open Bugs and Known Issues
- Documentation Accessibility

1 Documentation

Documentation for Oracle HTML DB is available at

`http://otn.oracle.com/documentation/index.html`

2 New Features

This section describes new features of Oracle HTML DB that are not documented elsewhere.

This section contains these topics:

- Rendering Select Lists in a Report or Tabular Form
- Escaping Special Characters Rendered from Session State

2.1 Rendering Select Lists in a Report or Tabular Form

You can use the `HTMLDB_ITEM` package to render select lists in a report or tabular form by making direct calls to `HTMLDB_ITEM.SELECT_LIST` or by applying certain display types to report columns. Note that this type of select list cannot exceed 32K characters. Also note that selecting too many options may result in the select list not rendering correctly.

In Oracle HTML DB 1.5.1, select lists rendered using this approach did not include closing `OPTION` tags for each option. Closing `OPTION` tags for each option are now included in Oracle HTML DB 1.6, reducing the number of

Copyright © 2003, 2004, Oracle.
All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

options allowed in a select list. The actual change in the number of options depends on the length of the option text strings. This also affects pagination if the row ranges in select list style is chosen. If the number of allowed options is reached, the pagination style changes to X-Y of Z without using a select list.

2.2 Escaping Special Characters Rendered from Session State

Page and application items can have their session state values established using a URL or in `POSTDATA`. In release 1.6, the HTML DB engine escapes these values for two item types when it saves them in session state.

2.2.1 Item Types Escaped On Input

To accomplish this, the HTML DB engine converts commonly used characters to to entity references. For example, the HTML DB engine converts the less than sign (<) to `&lt;`. This rule applies to two types of items:

- Page items of type **Display as text (does not save state)**
- Application level items (that is, items that are not associated with a particular page)

The objective of this change is to protect your application from cross site scripting (XSS) security attacks for item types that are not escaped during rendering. Developers may continue to store escapable characters into any type of item by setting those item values internally within the page (for example, using page processes, computations, or queries). However, attempts to externally set the values of application items or page items of type **Display as text (does not save state)** will result in their values being escaped when they are saved in session state. Be aware that setting these types of items to text containing HTML or JavaScript from outside the application (for example, using the URL) is a dangerous practice.

Note that you may have problems with existing pages if they rely on the previous behavior (that is, no items values being escaped on input.) Possible examples include:

- When escapable input characters are passed into session state using a URL or `POSTDATA` for page items of type **Display as text (does not save state)** and are to be rendered as form items on the page without being escaped. In release 1.6 the values will be escaped on input and thus will not render correctly.
- When escapable input characters are passed into session state using a URL or `POSTDATA` for application items or for page items of type **Display as text (does not save state)** and are referenced in HTML regions using `&ITEM_NAME.` syntax and are expected to be rendered exactly as they were represented on input. For example, the string `A<B` passed into page item `P1_ITEM` in the URL would be stored as `A&lt;B` and would be rendered incorrectly as `A&lt;B`.

To fix these types of pages:

1. Set the content of page items of type **Display as text (does not save state)** using item default source values, page processes, or computations instead of setting them using the URL method. In this way values will not be escaped when stored in session state and the values will not be escaped during rendering as form items.

2. Set the content of application items or page items of type **Display as text (does not save state)** using page processes or computations instead of setting them using the URL method. In this way values will not be escaped when stored in session state. These values will, however, be escaped during rendering in the HTML region, just as they were in prior releases, so that the less than sign (<) will appear as the less than sign (<) and will not be interpreted as a special character by the browser.

2.2.2 Item Type Escaped On Output

Page items of type **Display as Text (saves state)** are escaped on output when rendered as form items on a page. The purpose of this change is to correct a bug that caused these items to not be escaped. The new behavior makes this form item display type consistent with other form item display types.

Note that you may also have problems with existing pages if they rely on the previous behavior (that is, where form items of type **Display as Text (saves state)** were not escaped during rendering.) For example, you may have problems with a page with items that containing escapable characters used for HTML markup.

To correct this type of situation:

1. Change the item type to **Display as text (does not save state)**.
2. If there is a requirement to submit the value in `POSTDATA,` adjust the page design to achieve the desired results without introducing XSS exposures.

3 Open Bugs and Known Issues

This section describes bugs and known issues for Oracle HTML DB:

- Restoring Indented Link Values in a Report Column
- Implementing Themes in an Application Migrated from Earlier Releases
- Column Attribute Format in Japanese
- Creating an Item with a Japanese Item Name
- Runtime Errors in an Application Imported from a Previous Release
- Creating a Web Reference on a WSDL that Has Input Parameters Defined as Arrays

3.1 Restoring Indented Link Values in a Report Column

Earlier releases of Oracle HTML DB stripped all HTML tags from query column values when those columns where used in HTML expressions and links. In Oracle HTML DB 1.6, you can disable this behavior by turning off the Strip HTML attribute on the Report Attribute page. By default, this attribute is enabled so that applications developed in earlier releases will behave the same way after upgrading.

In release 1.6, stripping of HTML tags also includes `NBSC` escape characters. This type of character is used in some applications to indent link values in a report column. You can restore the indent after upgrading, by editing your report and setting the Strip HTML attribute on the Report Attribute page to **No**.

3.2 Implementing Themes in an Application Migrated from Earlier Releases

Oracle HTML DB release 1.6 introduces the concept of themes. A theme is a logical group and classification of templates within an Oracle HTML DB application. The HTML DB engine maps templates between different themes using classes. Because this functionality did not exist in previous releases, be aware of the following issues before trying to utilize themes in an application created in a previous release.

When you import an application into release 1.6 from an earlier release, Oracle HTML DB groups all the templates in the application into a theme named *Application Theme*.

Before switching the theme of an application developed in an earlier release you must:

- Edit each template in your application and assign it the appropriate class.

  Both the theme you are switching from and the theme you are switching to must contain the same classes. If you have a template that does not fall within a predefined class, you can use a custom class. Each template type has eight custom classes.

- Review the default templates (and template classes) in the theme you are switching from and the theme you are switching to. You can review the default templates defined in a theme on the Define Theme page.

  Note that failing to perform this step may result in unexpected results. For example, you might run into problems if one theme has a default page template that is one-level template while another has a default page template which is a two-level template.

If you switch an application theme and it does not render correctly:

1. Verify that the default templates for the active theme are set correctly.
2. Check that the templates within the active theme have the correct template classifications.

3.3 Column Attribute Format in Japanese

When you open the number or date format select popup dialog on the Column Attribute of a Page Definition in Application Builder, it always displays `'backslash'+ 5,234.10` in the dialog. It is expected that the symbol of 'yen' displays accurately in a Japanese environment.

Note that backslash and yen are the same character code point, but display differently depending on the selected font. Backslash is also displayed when applying the data format on the page in the application.

3.4 Creating an Item with a Japanese Item Name

If you create a form on a table or view based on an included column whose name is in Japanese using a wizard, the name of the new item will be included in Japanese.

To correct this problem, when you create new items on the Page Definition use alphanumeric characters A_Z, 0-9 and '_' for the item names. You may also need to changes item names to alphanumeric before you apply changes to the item.

3.5 Runtime Errors in an Application Imported from a Previous Release

If you export an application from an earlier Oracle HTML DB release and then import and install it using the installation pages in Application Builder, in rare situations you may encounter runtime errors after the application installs.

These errors often manifest themselves as PL/SQL parser or execution errors pertaining to blocks of PL/SQL code embedded within application components. The installation process sometimes splits strings greater than 200 characters into multiple lines. For example, lines may split between PL/SQL keywords, or at other places that cause parsing errors.

If you encounter these types of errors and suspect the installation process has split large strings:

1. Isolate the failing component containing the suspect PL/SQL within the application by editing the failing page in Application Builder.
2. Locate the blocks of code that appear to split incorrectly.
3. Attempt to split the blocks of code in more appropriate places, or insert whitespace with the lines until no runtime errors are observed.
4. Export the application, import the export file, and then reinstall it.
5. Retain the new export file as a permanent backup copy.

3.6 Creating a Web Reference on a WSDL that Has Input Parameters Defined as Arrays

If you create a Web service reference in Oracle HTML DB on a WSDL document that has input parameters defined as arrays, you will not be able to use built-in wizards to create a form or a form and report on that Web reference. Oracle HTML DB does not provide a user interface for input parameters that are arrays. Output parameters that are defined as arrays are handled properly if you use the Form and Report on Web Service Wizard.

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 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 does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.