Infor LN UI Difference Study

Publication information

Release: Infor LN UI 11.2.0

Publication Date: June 15, 2015

Document number: P3669F US

Contents

About this document

This document describes the differences between Infor LN UI and Web UI. LN UI is the HTML5-compliant browser-based user interface for Infor LN 10.3 and higher.

This document contains these sections:

- Deployment differences

  Describes the differences with respect to deployment and administration.
- End user differences

  Describes the differences which are relevant for end users.
- Developer differences

  Describes the differences which are relevant for LN developers.

For more information about LN UI, see these documents:

- *Infor LN UI Administration Guide (U9790)*
- *Infor LN UI Infor Ming.le-LN Plug-in User Guide (U9791)*

Contacting Infor

If you have questions about Infor products, go to the Infor Xtreme Support portal.

If we update this document after the product release, we will post the new version on this website. We recommend that you check this website periodically for updated documentation.

If you have comments about Infor documentation, contact documentation@infor.com.

Deployment differences

1

This list shows important deployment differences:

- LN UI does not require any client side java plugin. An HTML5-capable browser is sufficient. For supported browsers, see the *Infor LN UI Administration Guide (U9790)*.
- LN UI is supported on a wider range of client operating systems. For supported client operating systems, see the *Infor LN UI Administration Guide (U9790)*.
- LN UI only supports Infor LN 10.3 and higher. Baan IV, Baan 5, and older LN versions are not supported.
- LN UI is only supported in combination with Infor Ming.le™. Classic mode Web UI and standalone mode is not supported.
- LN UI does not support Fujitsu Workflow.
- LN UI does not support old style Web UI homepages.
- LN UI only supports report preview (print to display) in PDF format, not in HTML format.
- LN UI does not support integration with LN Report Viewer.
- LN UI does not support Java Web Start mode.
- LN UI does not support local printing (printers connected to the client) through BWPRINT. Local printing is only supported through the browser PDF viewer plugin. Bar code printing using BWPRINT is possible using a Windows Server Printer device. See "Windows Server Printer" in the "Device Management" chapter in the *Infor Enterprise Server Administration Guide (U8854)*.
- These installation modes are supported for LN UI:
  - A fresh installation. This requires a Tomcat instance which cannot be shared with an existing Web UI installation. After a fresh installation of LN UI, the administrator cannot import configuration settings from an existing Web UI installation.
  - An upgrade installation from an existing Web UI installation. This installation replaces the existing Web UI product. After an upgrade installation, the administrator can import Web UI configuration items from the previous Web UI installation. For details, see the *Infor LN UI Administration Guide (U9790)*.
- LN UI supports these authentication methods:
  - Single sign-on based on Integrated Windows Authentication (IWA). This authentication mode is only available when the LN UI Web server is deployed on a Windows OS.
  - Single sign-on based on IFS/ADFS authentication.
  - For testing purposes, only back-end authentication is supported.

- LN UI supports only the Infor UX3 UI skin.
- LN UI cannot start client side applications (such as Notepad or MS Word) from an LN session.
- LN UI does not support Infor LN Office Integration.
- LN UI does not support the ODM and ECM Infor LN document integration packages.
- LN UI does not support Silverlight-based Workbench sessions when using Chrome or the Safari browser.
- LN UI help does not work in Infor LN BW and Infor LN Worktop. For details, see Infor Xtreme KB 1499582.
- Online help information is stored on the Web server where LN UI is installed, not on the Infor Ming.le server. In case of Web UI, online help information is stored on the Infor Ming.le server. For details on how to install help packages for LN UI, see the *Infor LN UI Administration Guide (U9790)*.
- Because online Help is stored on the LN UI web server, the administrator must change the configuration for the documentation context application in Infor Ming.le. For details, see the *Infor LN UI Administration Guide (U9790)*.

End user differences

2

This list shows important differences which are visible to end users:

- During startup of Infor LN UI users cannot select another user profile. Users can change the default user profile and the user profile settings through the **Options > Settings** menu option.
- Floating sessions start differently compared to Web UI. When a floating session is started in LN UI, a new browser window is opened. By default, most browsers do not allow a new browser popup. To enable this feature, the user must explicitly configure the browser to always allow browser popup windows for the web site that hosts LN UI.

  You cannot start a floating session from a floating session. If a new session is started from a floating session, this new session is always opened in the same browser window as the original session, even if you hold down the Ctrl key.
- LN UI follows the zoom level settings of the browser. This setting was ignored by Web UI.
- When transferring files from the server to the client, or from the client to the server, the UI behavior differs compared to the Web UI. In LN UI the user must always explicitly acknowledge the client file access operations. This is caused by an HTML5 security restriction. An example of such an operation is the export to spreadsheet feature, which is available in most Infor LN sessions.
- LN print sessions do not show a list of locally installed printers on the **Device** Tab. This is caused by an HTML5 security restriction, which does not allow direct access to client side printers.
- The format of date, time, and numbers does not follow the client operating system settings, like in Web UI. In LN UI, these formats are directly related to the current locale of the user in Infor Ming.le. See the *Infor LN UI Infor Ming.le-LN Plug-in User Guide (U9791)*.
- Keyboard shortcuts for frequently used commands differ from the shortcuts used in Web UI and Worktop. This is because many existing shortcuts conflict with shortcuts used by browser(s) and platform. For details about keyboard shortcuts in LN UI, see the *Infor LN UI Infor Ming.le-LN Plug-in User Guide (U9791)*.
- LN UI does not support automatic hyperlink recognition in multi-line text fields and the text editor session.
- In the text editor session, LN UI does not support the **Import file** and **Export file** commands. This is caused by an HTML5 security restriction.
- When the application shows a non-interrupting message window, either because non-interrupting message mode is enabled or because the application autonomously generates a non-interrupting message, this message window does not popup automatically in LN UI. Instead, a flashing icon is displayed in the status bar. The user must click this icon to show the non-interrupting message window.

- When the LN server generates a system message while the user is already signed in, the system message window does not popup automatically in LN UI. Instead, a flashing icon is displayed in the status bar. The user must click this icon to show the system message window.
- In LN UI, the General Table Maintenance (ttaad4100) and Display General Table Information (ttaad4500) sessions look and behave like normal sessions. In Web UI, these sessions look like old style character mode sessions.
- LN UI does not support sending e-mails through a client side Outlook client or other e-mail client.
- In LN UI, the **Select Activity** top-level options menu item has been renamed to **Debug and Profile 4GL**.
- This table shows menu items which are **not** present in the LN UI top-level options menu:

Menu item	Remark
Cache Menu	No caching of menu items in LN UI.
Refresh Menu	No cache to be refreshed in LN UI.
Show Mandatory Indication	In LN UI this option is always on.
Auto Complete Settings	In LN UI this option is always on. The maximum number of entries is fixed to 7.
Show Excluded Business Processes	In LN UI this option is always on.
Show Only Main Processes	In LN UI this option is always off.

- On any field of a details session in LN UI, the **Hide field** and **Field Help** options are displayed when the user right-clicks the mouse while holding the Alt key, or the Ctrl key, pressed.
- LN UI does not support `file://` urls in the `open.url.local` 4GL function. Modern browsers prohibit local file access if the top-level window is fetched with `http://` or `https://`.

DEM

This list shows the differences between the DEM Process Viewer in LN UI and the Process Viewer in Web UI:

- A button was added to the toolbar to toggle the visibility of the icons between 'always visible' or 'visible on hover'. This is also reflected when printing the model.
- The icons are somewhat changed and positioned differently.
- Clicking an icon opens the most obvious action on the corresponding activity.
- In Web UI, if an activity contains a sub process, a drop-shadow is displayed around the activity. In LN UI this shadow is replaced by a zoom (play) button on the right side of the activity.
- Every (sub) process is opened in its own LN UI window. Therefore the browse buttons (left and right pointing play buttons) are removed from the toolbar.
- Manual and no-permission activities have an indicator that is always visible.
- The color that is given to annotations in the Process Modeler is now also displayed in the Process Viewer.
- More consistent feedback from the User Interface. For example, the cursor changes to show which elements are clickable and which are not.

This list shows the differences between the DEM functionality in LN UI and the DEM functionality in Web UI:

- External applications cannot be started. This is caused by an HTML5 security restriction.
- A URL to open an Internet/Intranet site is supported in LN UI. A URL that refers to a UNC or another file based path is not supported.
- An alert is displayed if the DEM Process Viewer tries to open an invalid location. As a workaround, open the Windows Run Program dialog and copy the content of the alert to this dialog. Then click **OK** to start the external application or open the external document.

Gantt chart

This list shows the differences between the Gantt chart functionality in LN UI and the Gantt chart functionality in Web UI:

- LN UI has three levels of time information in the time bar at the top of the Gantt chart. Web UI has two.
- LN UI goes to seconds resolution instead of minutes.
- LN UI supports (Ctrl or Shift)+Mouse Wheel to zoom within the Gantt chart. Ctrl+Mouse Wheel does not work in Chrome.
- LN UI zooms in and out around the mouse cursor position. Web UI zooms about the center of the chart.
- LN UI does not have a horizontal scroll bar to pan the Gantt chart. Instead, you can drag the time bar, or zoom out and zoom in.
- LN UI does not support reordering (moving) of the property columns.
- LN UI implements different keyboard navigation in the left-hand tree:
  - The Right Arrow and Left Arrow buttons open and close nodes. In Web UI these keys move the 'focus' to different columns.
  - Pressing Enter does not move the selection to the next item, as it does in Web UI.

Developer differences

3

This section provides 4GL-developers guidelines for adding LN UI support to their sessions. The section basically consists of two parts:

1. A short explanation on the impact of LN UI on sessions and the available tools functions.
2. A set of descriptions on how to implement LN UI support.

Short explanation

LN UI does not support functionality that requires a higher privilege level on the client. Therefore some client access functions are not allowed anymore. You must replace calls to these functions with other client access functions, which are allowed to be used when running in LN UI mode.

Also, when a session has fields on its form to specify a client file or client folder, these fields must be hidden when running in LN UI mode.

Unsupported client access functions

These functions are not supported in LN UI:

- client2server()
- server2Client()
- create.local.file()
- create.local.directory()
- dir.select.dialog.local()
- get.client.directory()
- get.local.filename()
- remove.local.directory()
- remove.local.file()
- seq.fstat.local()

- seq.open.dialog.local()
- seq.open.dialog.next()
- seq.saveas.dialog.local()
- start.application.local()

If one of these functions is called while running in LN UI mode, this message is displayed:

```
This function is not supported in the LN UI:  [function name]
```

LN UI client functions

Instead of the unsupported functions, you must use these functions:

- Single file functions:
  - client.upload.file() - to upload a single file to the server.
  - client.download.file() - to download a single file from the server.
  - client.show.file() - to show a single file in a new tab (iFrame) or new browser.
- Multiple file functions:
  - client.prepare.download() - to prepare for a download for one or more files.
  - client.add.download.file() - to add a file to the list of files to download.
  - client.start.download() - to start the download of one or more files.
  - client.upload.files() - to upload one or more files to a server folder.
  - client.get.upload.file.count() - to get the number of files uploaded with client.upload.files().
  - client.get.upload.file() - to get details on an upload file.
  - client.delete.upload.file.object() - to delete the result object returned by client.upload.files().

If one of the LN UI client functions is used while NOT running in LN UI mode, this message is displayed:

```
This function is not supported in the WebUI: [function name]
```

Note: For details on these functions, see the latest *LN Programmer's Guide*.

LN UI detection

You can use these functions to detect whether a session must run in LN UI mode:

- get.ui.mode() - returns the mode of the thin client. These are the possible return values:
  - NO_UI - without a UI
  - BW_UI - UI is BW

- WEBUI_CLASSIC - UI is Web UI stand-alone
- WEBUI_COMMONUI - UI is Web UI as Infor Ming.le plug-in
- HTML_UI - UI is LN UI

- tc.is.html.ui() - returns true if the thin client runs in LN UI mode.

The tc.is.thin.client() function is also still available to detect whether the thin client runs in NO_UI/BW_UI (false) or WEBUI_CLASSIC/WEBUI_COMMONUI/HTML_UI (true).

Implementing LN UI support

To support LN UI, the developer must complete these steps:

- Cleanup the UI of the session by hiding form fields that are used to specify a client file or a client folder.

  Remove any checks for these fields, in the check.input section and in the DAL is.valid hook.
- Implement an alternative path for accessing client files based on legitimate LN UI functions.

These restrictions apply to LN UI support:

- Starting a client application without a file is no longer supported.
- Whether a client application is opened strongly depends on the user, because the user is in control in LN UI.

Hiding form fields

Characteristics: the session allows users to specify a file on the client. The specified client path is used to access (create, open, etc.) a file on the client machine.

Before.program

To hide the fields, modify the before.program section of the UI-script. Add this code, with the appropriate field names, in the before.program section:

```
if tc.is.html.ui() then
 inputfield.invisible("<name of field 1>")
 inputfield.invisible("<name of field 2>”) … etc
endif
```

Skip validation of hidden fields

Although you have hidden the fields, validation checks might still be activated. Use the tc.is.html.ui() function to check the UI-mode and only validate these fields if the UI-mode is not LN UI.

For example:

```
field.curr.impf:
check.input:
 domain ttst255m impf.dir, impf.file
 long size

 if not tc.is.html.ui() then
  if isspace(curr.impf) then
     set.input.error("ttadv227006")
     |* Enter (correct) path.
  endif
 endif
```

Displaying File alternative

Characteristics: the session shows a file with server2client followed by start.application.local.

Usually this type of construction is used:

```
if server2client(aServerFile, aClientFile, textMode) = 0 then
 aClientApplication = "anApplication.exe"
 start.application.local(aClientApplication & " """ &
   aClientFile &""""", aWaitFlag, someExitCode)
endif
```

To show a file in LN UI mode, use one of these functions:

- client.show.file()
- client.download.file() - followed by the user that opens the downloaded file

Browsers typically support files with these file extensions:

- .txt - mime type: text/plain
- .pdf - mime type: application/pdf
- .htm(l) - mime type: text/html

Therefore we strongly recommend that you use client.show.file() for these file types. If you use client.show.file() for these file types, the file is streamed directly to the browser; no download question is asked.

For all other file types we recommend that you use client.download.file().

HTML support - for other file types than .txt, .pdf and .htm(l)

For commonly used extensions such as .xlsx and .docx, the client usually has an application associated with the file extension. For unknown file extensions, the user is asked which application must be associated with the file extension of the downloaded file. When the user decides to open the file, the associated application is automatically launched to open the file.

You can use this type of LN UI support code:

```
if tc.is.html.ui() then
 client.download.file(aServerFile, aMimeType)
else
 … original code as depicted in the box above
endif
```

To enable the client that runs in LN UI to use the correct application for opening the file, complete one of these steps:.

- Set the file extension of the file name correctly, for example to ".txt".
- Set the mime type accordingly, for example to "text/plain".

HTML support - for .txt, .pdf and .htm(l)

For files with a .txt, .pdf, or .htm(l) extension, the file can be streamed to the browser without actually downloading the file. This means that the user is not asked explicitly for the download. This function automatically shows the content of the file in a new tab or a new browser.

```
if tc.is.html.ui() then
client.show.file(aServerFile, aNewWindow, aTitle, aMimeType)
else
 … original code …
endif
```

This variant may open a separate window when aNewWindow is set to true.

Editing File alternative

Characteristics: Use server2client to copy a server file to the client. Then use start.application.local to edit the file. Use client2server to copy the changed file back to the server.

Usually this type of construction is used:

```
if server2client(aServerFile, aClientFile, textMode) = 0 then
 aClientApplication = “anApplication.exe”
 start.application.local(aClientApplication & " """ &
   aClientFile &""""", aWaitFlag, someExitCode)
```

```
 client2server(aClientFile, aServerFile, textMode, true)
endif
```

You must replace the single action for editing with two actions:

1. An action to download the file from the server to the client. Afterwards the user can edit the file with the appropriate application.
2. An action to upload the changed file from the client to the server.

LN UI support - download/edit part

```
if tc.is.html.ui() then
 client.download.file(aServerFile, aMimeType)
else
 … original code as depicted in the box above
endif
```

LN UI support - download zipped archive

```
long handle

if tc.is.html.ui() then
 | Zip files and directories in file with .zip extension
 handle = zipinfo.new(aServerZipFile)
 zipinfo.add(handle, aServerFile1)
 zipinfo.add(handle, aServerFile2)
 zipinfo.add(handle, aServerDirectory1)
 zipinfo.add(handle, aServerDirectory2)
 zipfile.build(handle)
 zipinfo.delete(handle)

 client.download.file(aServerZipFile)
 | Delete intermediate zipfile
 file.rm(aServerZipFile)
else
 … original code as depicted in the box above
endif
```

HTML support - upload part

The upload part requires these changes:

1. A form command implementation that uploads the file from the client back to the server.
2. A command on the session/form to call this implementation. This command is only available if the client is running in LN UI mode.

For the form command implementation, you must use this type of construction:

```
Function extern upload.file()
{
 client.upload.file(aServerFile, theSelectedClientFile, aMimeType)
}
```

For a form command implementation, that uploads and unzips a zip archive, you must use this type of construction:

```
Function extern upload.and.unzip.archive()
{
 client.upload.file(aServerArchive, aServerZipFile)
 zipfile.extract(aServerZipFile, theRootSelectedArchive)
 | Delete intermediate extracted zipfile
 file.rm(aServerZipFile)
}
```

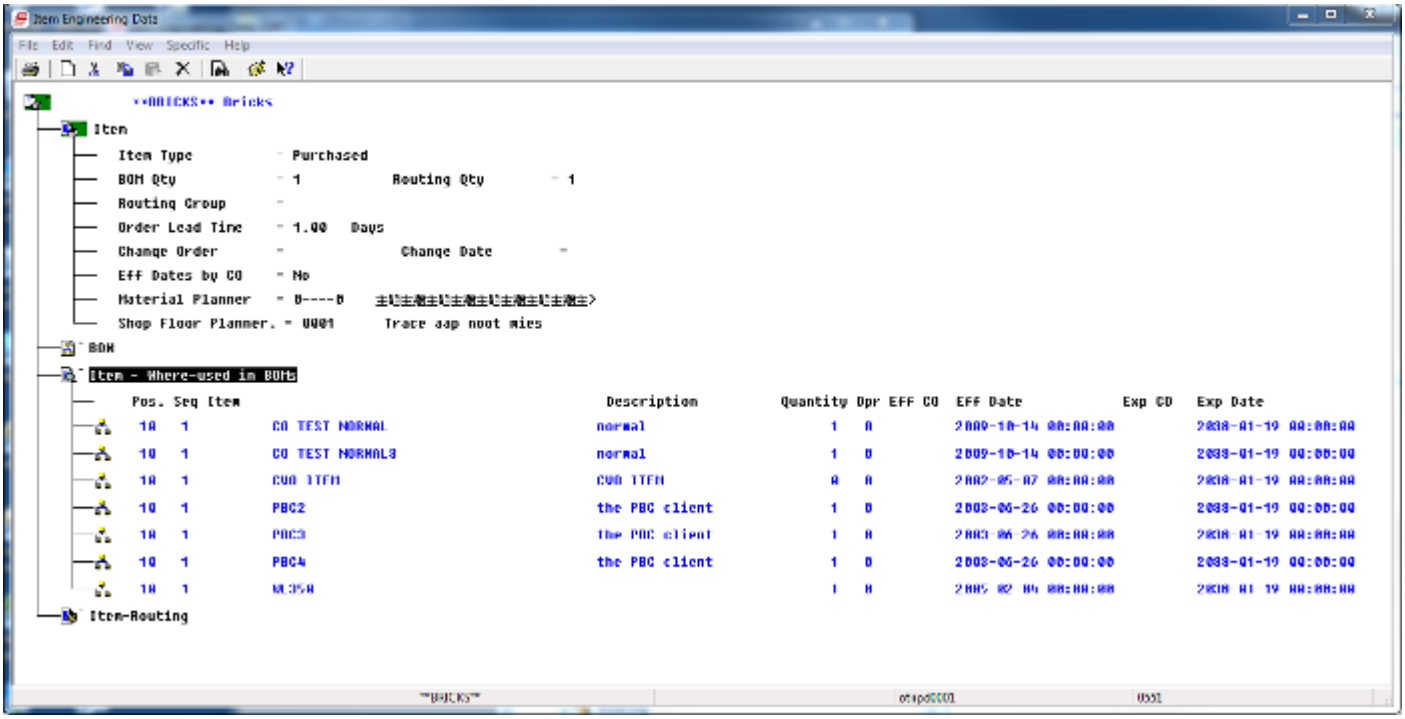
Note: There is a risk that the server file was changed in the meantime. Uploading an old version of the file overwrites these changes. When implementing this two phased download/upload approach for editing a file, you must add a mechanism to determine whether the server file was changed in the meantime. If a changed server file is detected, the user must be warned and asked to continue before performing the actual upload.

To remove the form command from the session, use this code:

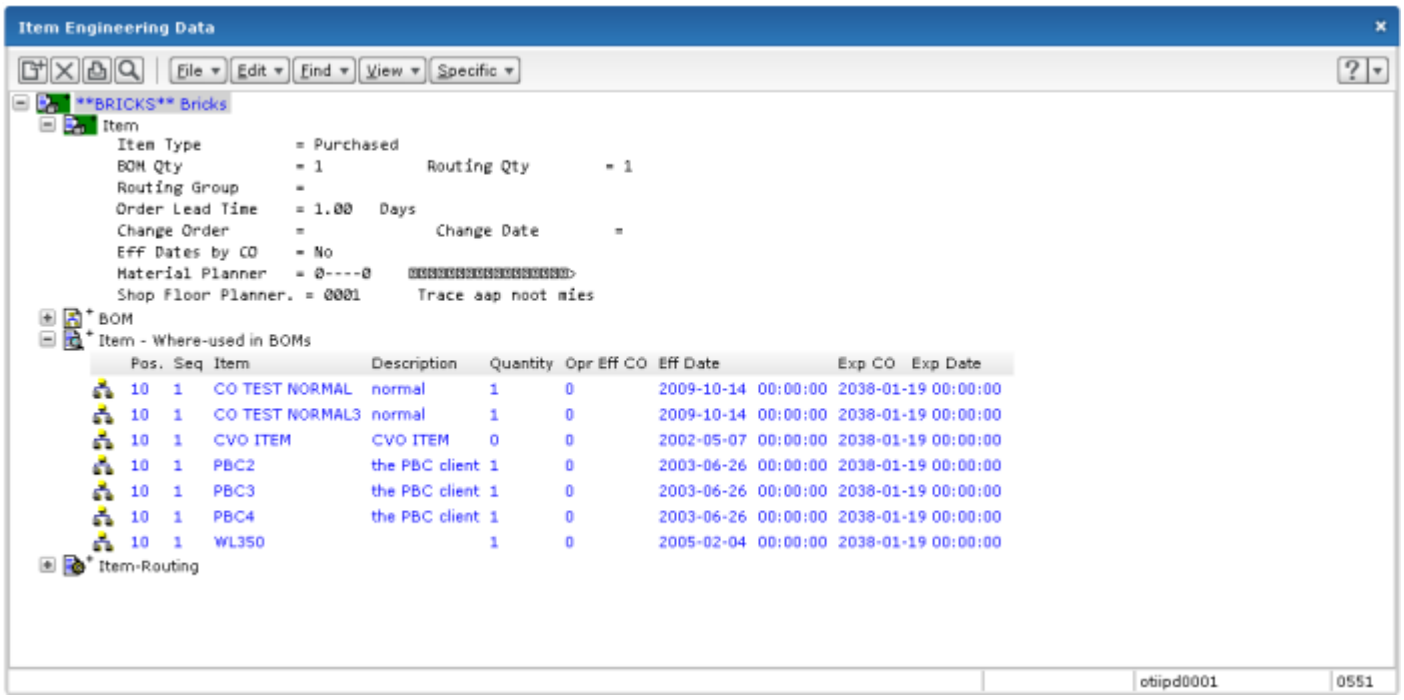
```
after.form.read:
 if not tc.is.html.ui() then
  remove.form.commands("upload.file")
 endif
```

GBF

GBF is fully supported, but there is one usage pattern which is difficult to support in LN UI. This figure shows this usage pattern in a sample session in Worktop/BW:



This figure shows the same session in Web UI:



These examples show node descriptions in a tabular manner. In BW, this only works when using fixed-sized fonts; all characters have the same width.

In Worktop/BW, a fixed-sized font can be used because the application passes the GBF.OPT.FONT.FIXED value in the gbf.init() call.

Web UI has difficulty in maintaining logic to determine the column boundaries.

LN UI support - use fixed fonts (not recommended)

By using fixed width fonts in LN UI for GBF descriptions, the alignment of the characters in the description remains correct. To achieve this, perform a gbf.init() call with the GBF.OPT.FONT.FIXED passed. However, fixed width fonts are not that pretty and might even harm the overall look and feel of the UI.

LN UI support - use variable width fonts (recommended)

To show descriptions in a tabular manner with variable width fonts, use the gbf.desc.to.column() function. See the *LN Programmer's Guide* for a detailed description of this function.

If gbf.add.object is called with a 'formatted' description, you must change this call. See the examples below.

This example shows a gbf.add.object call with a 'formatted' description:

```
func.ret = gbf.add.object(where.used.key,
  sprintf$("%4d %3d %47s %15s %13.4g %3d %6s "&
  "%u002 %U001 %6s %u002 %U001 ",
  tibom010.pono,tibom010.seqn,
  tibom010.mitm, tcibd001.dsca(1;15),
  tibom010.qana, tibom010.opno, tibom010.efco,
  tibom010.indt, tibom010.indt,
  tibom010.exco, tibom010.exdt, tibom010.exdt),
  WHERE.VAL, GBF.LEAF, par.leaf.val, 0, 0, 0,
  MASK.TOOLS, line.col, line.style)
```

You must change the above call as follows:

```
func.ret = gbf.add.object(where.used.key,
  gbf.desc.to.column(
  tibom010.pono,tibom010.seqn,
  tibom010.mitm, tcibd001.dsca(1;15),
  tibom010.qana, tibom010.opno, tibom010.efco,
  tibom010.indt, tibom010.indt,
  tibom010.exco, tibom010.exdt, tibom010.exdt),
  WHERE.VAL, GBF.LEAF, par.leaf.val, 0, 0, 0,
  MASK.TOOLS, line.col, line.style)
```

Note: gbf.desc.to.column also works for other UIs such as. BW, Web UI, and Infor Ming.le. Therefore, you do not have to use tc.is.html.ui() to check the client mode.