http://www.lansa.com

LANSA

Newsletter
July 2007

New VLF version

EPC804 provides Visual LANSA Framework (VLF) users with a host of new features.

Full details of EPC804:

1. Performance

This new version of the Framework is expected to perform better than all previous versions.

To accommodate the performance improvements the internal structure of the Framework has been heavily modified.

If you have created Visual LANSA code that accesses the internals of the Framework (which is a risk that some users have assumed), it may no longer function correctly and will need to be modified and/or recompiled.
Examples include non-standard code inclusions in UF_EXEC, UF_ADMIN or UF_DESGN entry points or equivalent, non-standard user-defined Code Assistants, etc.

If you have processes that read the Framework XML definition files, they may need to be modified to accommodate changes made to the XML structure.

WARNING: The internal changes required to accomplish this mean that for this version (unlike previous versions) *you cannot execute different VLF versions in different LANSA partitions within the same LANSA system*.
This means when you upgrade a Framework version in one partition to EPC804 level you must also upgrade any other Framework versions in any other partitions within the same LANSA system.

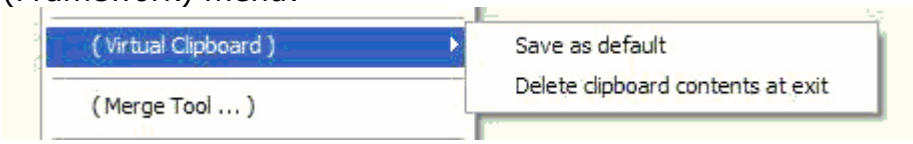
In This Issue

2. Instance List

Improved Application Tracing for Relationship Handlers
If you are using a Relationship Handler to dynamically expand nodes in an instance list tree and turn on application level tracing you will find a large amount of trace data is produced regarding the call to the relationship handler and what it returns. This makes finding problems in your relationship handlers easier.

3. Virtual Clipboard

New Virtual Clipboard Control Options
As a designer you will see a new (Virtual Clipboard) control options on the (Framework) menu:



The new Save as Default option allows you to save your virtual clipboard settings as a default file. Typically default file clipboard files are deployed to end users to establish basic system defaults.

The new Delete clipboard content at exit option causes the virtual clipboard to be deleted at Framework shutdown. This saves you from having to locate the files the clipboard is saved in and manually deleting them.

4. Custom Properties

Frameworks that share a User set no longer use the same custom property values for their users. This means that Frameworks with different Framework IDs but sharing the same User Set can store separate values of Custom Properties for the same user.

5. For non-English systems

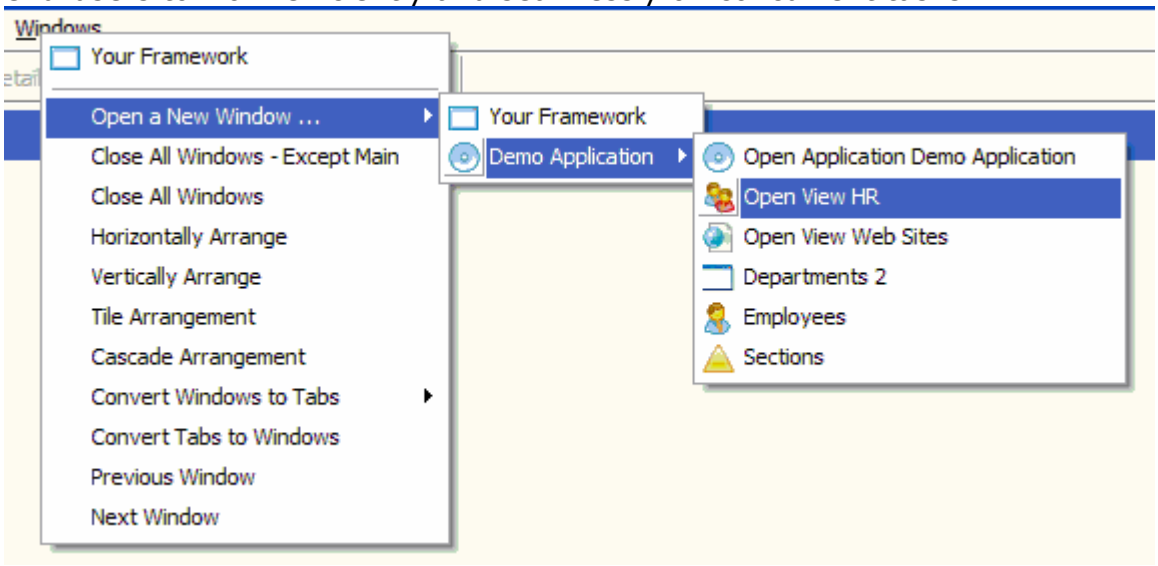
A. New default translation tables for Framework server definitions
The default translation tables used for Framework server definitions *using RDMLX partitions/connections* have been changed from QEBCDIC/QASCII to *JOB**/***JOB. Typically these values give better default translation results. If you have an existing VLF application using an RDMLX server definition *and are using the default translation table values*, then they will automatically change to use these new values.

B. Translation of end-user visible text
Some text strings used by the Framework are *end-user* visible and may need to be translated into your language. This translation is carried out using LANSA function UFU003 (or your equivalent). A new version of UFU003 has been shipped.

Review UFU0003 (in process UF_SYSBR) to see what new end user strings are now available. The new strings relate to areas like the new multi-window facilities and hints for the new navigation pane selection icons.

6. Usability

A. Framework Windows
Programs and end-users can now open and control many Framework windows. Being able to have several objects open for editing at the same time allows the end-users to work efficiently and seamlessly on concurrent tasks.



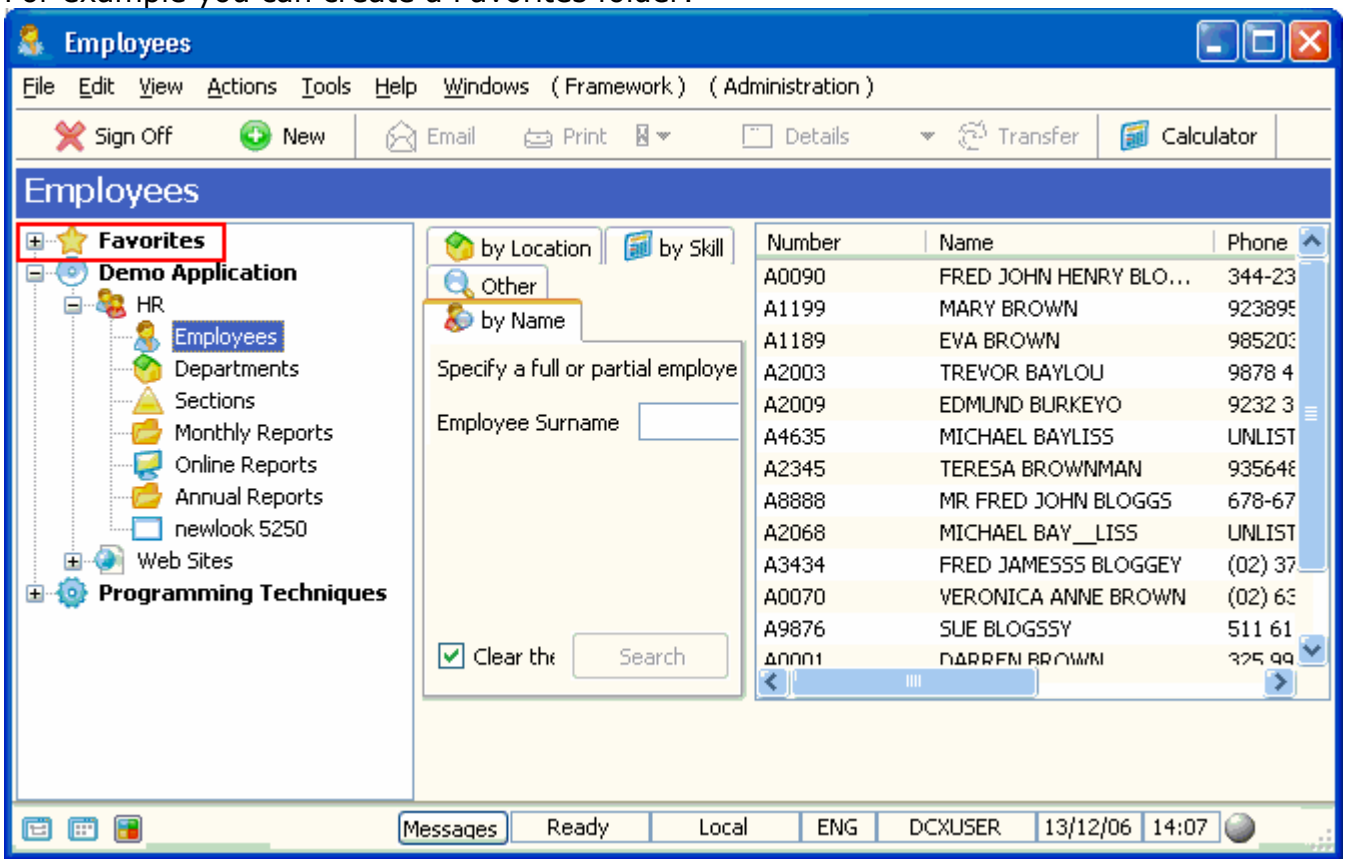
This feature is available in Windows only.
As a designer you can set a limit on how many windows that an end user can have concurrently open. You may control whether the whole Framework, individual applications, application views or business objects may, or may not, be opened in independent windows.
Full programmatic control of Framework windows is provided. Filters or command handlers may:

- Open or close windows
- Select what content is accessible in the window
- Enumerate all open windows
- Control the signaling of events to windows using the new parameter WindowScope in the avSignalEvent method.
- Switch to a new window to display a business object using the new optional parameter TargetWindow in the avSwitch method.
- Pass information into and out of windows

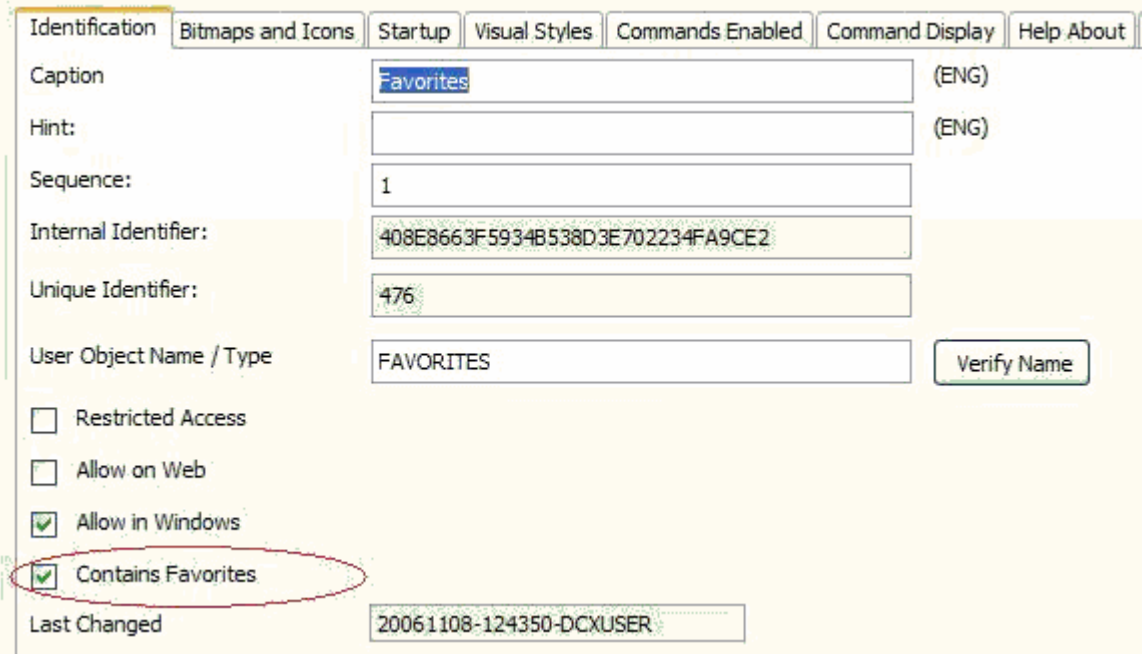
B. Favorites Folder

If an application designer permits it, end users can now create their own 'Favorites' application containing just the business objects they most commonly use. An application designer may choose to allow multiple favorites applications (eg: Normal Favorites, End of Month Favorites and Year End Favorites). Applications with the Contains Favorites property checked can store shortcuts to an end-user's favorite business objects. To do this end-users simply drag business objects they commonly use into the application where this property is checked. The actual business objects remain in the application they belong to.

For example you can create a Favorites folder:



By creating an application with the caption Favorites with the *Contains Favorites* property selected:

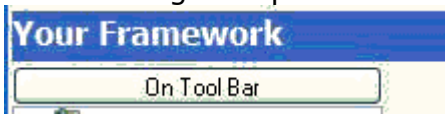


The favorites information is stored in the Framework Virtual Clipboard in the user's temporary directory.

This feature is available in Windows only.

C. Navigation pane view buttons have changed

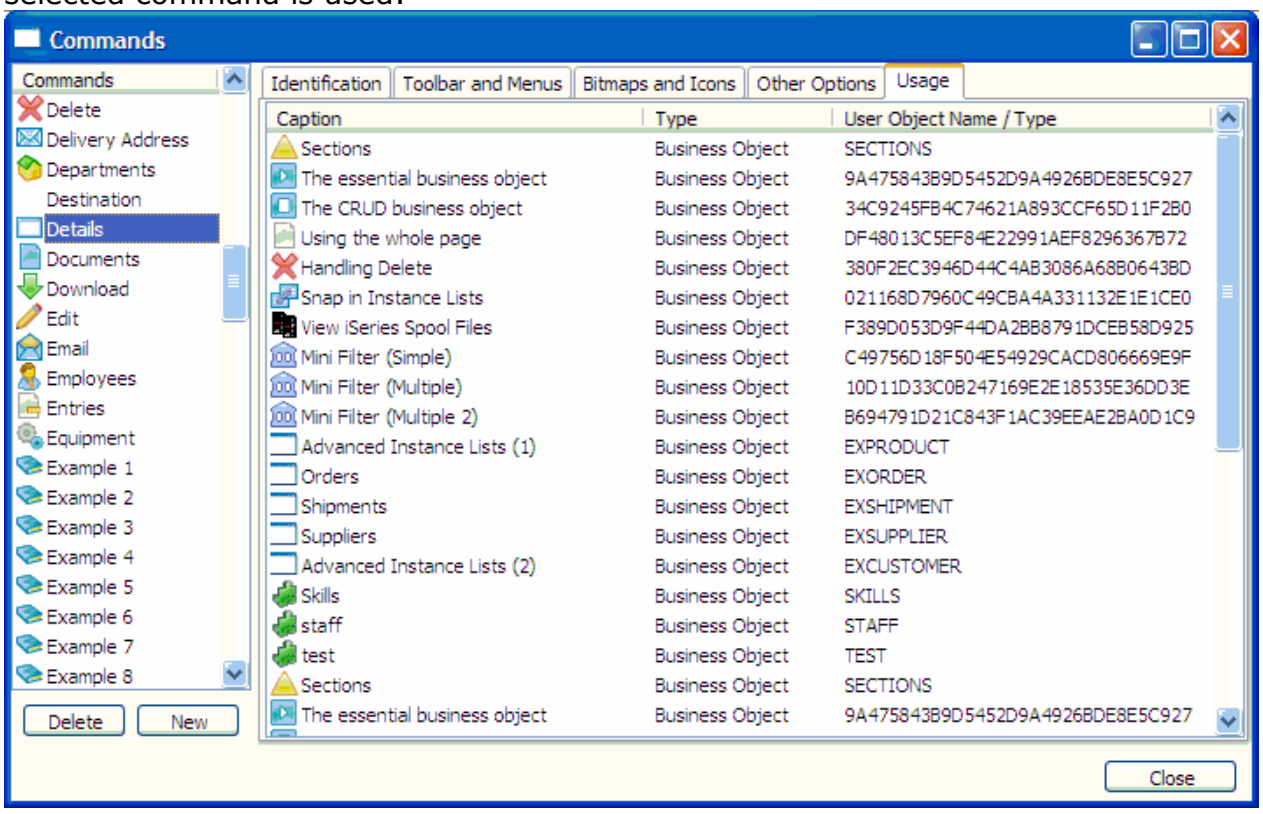
The navigation pane buttons that could be used to cycle through the list, tree and toolbar navigation pane views have been removed:



And the buttons have been replaced by three small icons on the left the status bar like this:

The icons represent the tree, list and drop-down button options. The user can now navigate to the required view in a single click.

D. Command Usage tab
The tab lists the business objects, applications and Frameworks in which a selected command is used:



7. Deployment
New sections in the VLF Guides fully describe all forms of application deployment. Planning sheets and check lists are provided.

8. Imbedded Interface Points
New IIP methods in UF_SYSTM (or equivalent) are now symmetrically invoked when the main framework window or a secondary framework window are:
a) opened and ready for work or
b) they are closing.

The new methods are called avMAINWindowReady, avSECONDWindowReady, avCloseMAINWindow, avCloseSECONDWindow.
The new avXXXXXWindowReady methods are invoked when a) the window has been opened, b) the user has been logged on and connected to any server and c) all Framework setup operations are completed. Their purpose is to provide a single point at which you might perform operations such as:

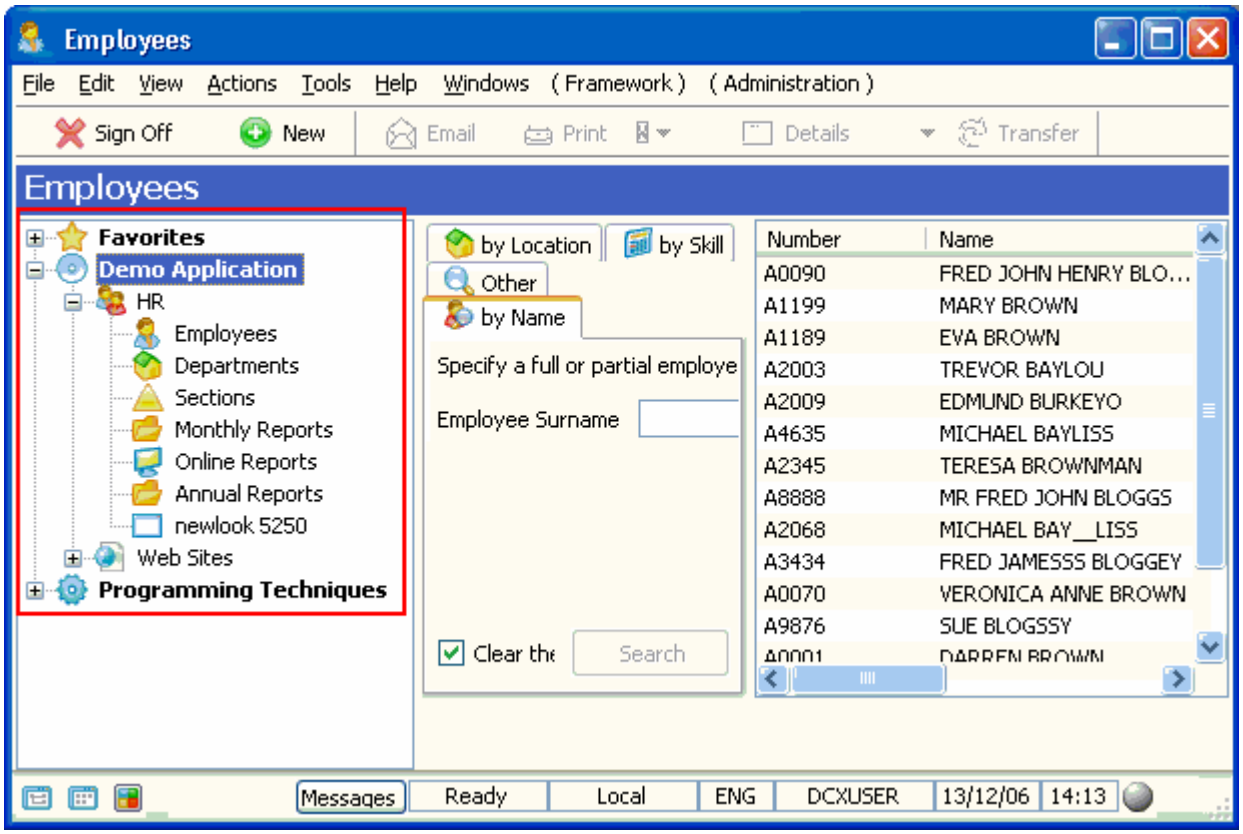
* Ask the user for additional logon details (eg: select a company).
* Map setup information into the virtual clipboard.
* Switch to an initial application, business object and/or command.

In all cases, a parameter #Continue is provided to allow you to control whether the window should continue to be opened or closed.

9. Look and Feel

A. Applications shown in Bold

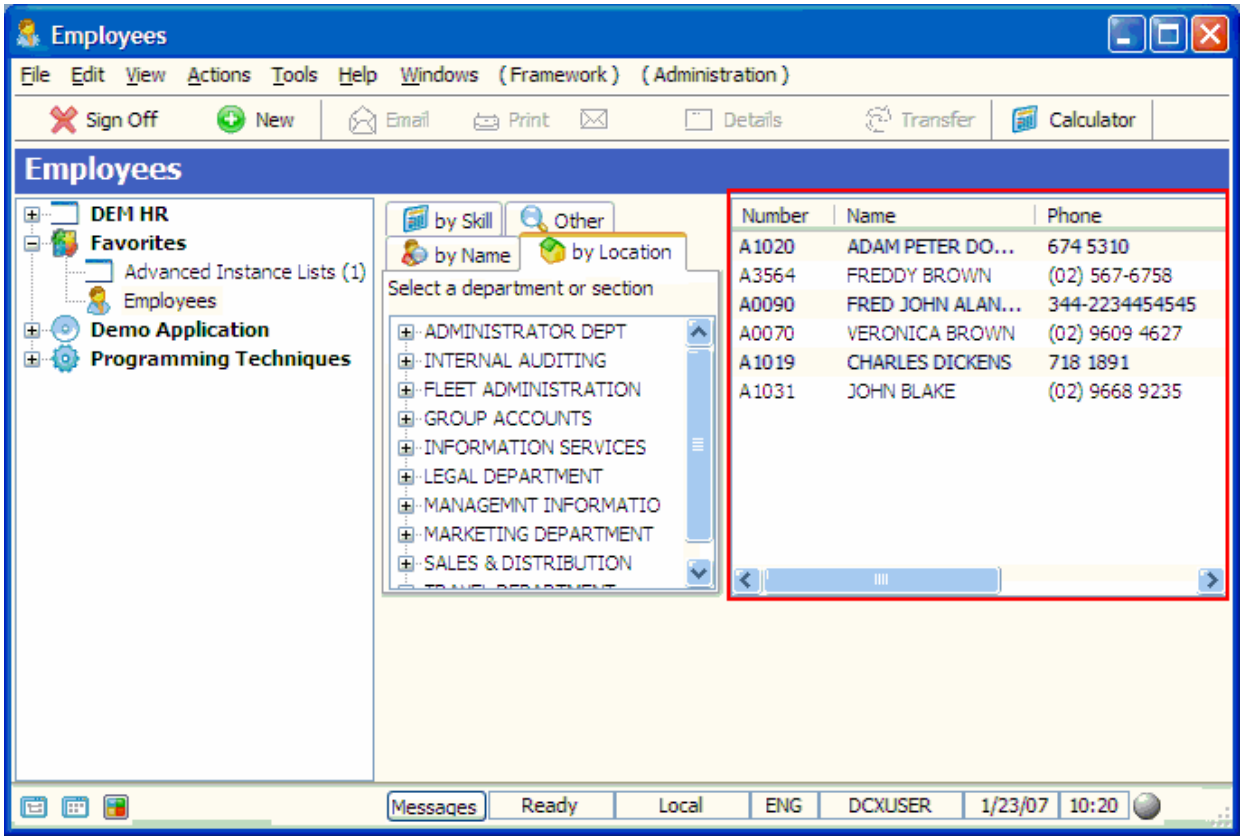
Applications in the tree view navigation pane are now displayed in bold.



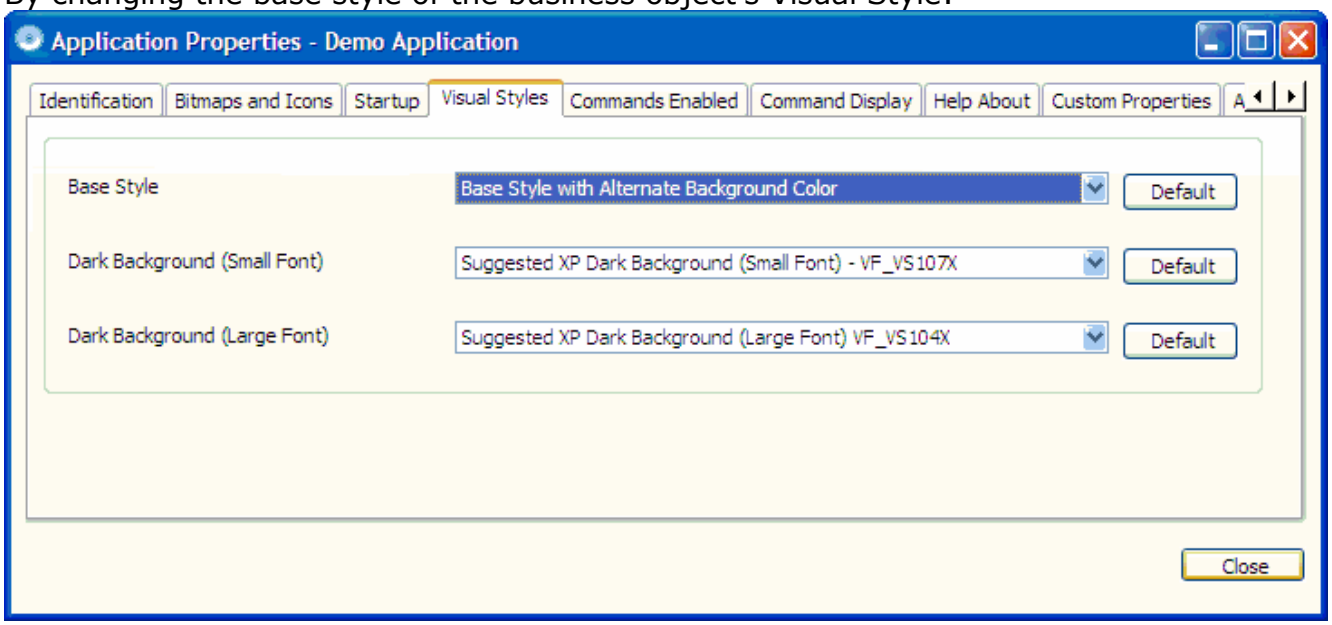
B. Alternate Row Colors in Instance Lists

Instance lists may be displayed with alternate rows using different background colors.

You can show an alternate row color for the instance list:



By changing the base style of the business object's Visual Style:



10. RAMP - Screen Wrappers

A screen wrapper is a VL component that layers over underpinning 5250 RAMP screen(s). Screen wrappers allow you to put high GUI veneers over the 5250 screen(s), without having to spend the time and money required to analyze, rewrite and then retest all the business logic imbedded inside them
A screen wrapper can pick values out of 5250 screens and present them to the user in completely different ways. Equally, a screen wrapper can accept input from the user and map it back into the 5250 screens to cause 5250 transactions to take place.

11. RAMP - Virtual Clipboard Access

Information placed onto the virtual clipboard by VLF filters and command handlers can now be read and updated from RAMP scripts. This significantly improves the ability for RAMP scripts and filters and command handlers to exchange information. This new feature may be used in VLF web or windows based applications.

12. RAMP - User Defined Scripts

Commonly used RAMP script logic can now be placed into a common JavaScript file. This feature allows for better reuse of commonly reused JavaScript logic.

13. RAMP - Newlook Function Key Bar

The newlook function key bar may now be optionally displayed on RAMP screens. This is most commonly done in pop-up windows. There are some provisos related to this new feature, so please review the documentation before use.

14. RAMP - Multilingual Button Captions

The captions show on RAMP buttons can now be changed to be multilingual.

15. RAMP - Unknown Screen Lock Message

The message that appears when a user attempts a RAMP navigation from an unknown screen has been improved. The message presented is now different for designers (technical) and end users (softer). Additionally the message text may now be changed from a RAMP script to exactly what any site requires.

16. RAMP - Improved Handling of Unknown Screens

Often unknown screens are left on display by users, causing the unknown screen lock message to be displayed when they attempt to navigate somewhere else. Now you can instruct RAMP as to how to 'guess' what it might do automatically to make an unknown screen go away, so that the users navigation request can be handled correctly. For example, your RAMP application may have many undefined F4=Prompt pop-up windows that are all closed by using F12=Cancel. You can instruct RAMP that when an unknown screen is on display (eg: an F4=Prompt window) it should first try F12 (to see if it can close the window) before displaying the lock message.

17. RAMP - Change Date/Time/User Displayed

The RAMP tool now displays the last changed date-time-user for all screens and scripts.

VLF and Vista

If you install the VLF onto a new Vista Windows system you may have trouble with RAD-PADs. RAD-PADs are used during prototyping in place of real filters and command handlers. You typically input notes, drag and drop images, etc, onto them. RAD-PADs are only used by developers, not by end users.

The problem occurs because Microsoft have stopped shipping the DHTMLEDit Active-X control.
Microsoft have done this because the DHTMLEdit control has been a source of security problems that they cannot seem to control. So to resolve these security problems Microsoft have just stopped shipping the DHTMLEdit control on Vista systems.

To resolve this you can manually install the DHTMLEdit control onto Vista. There are MS instructions available on how to do this.
This DHTMLEdit control is available from MS for download at:
http://www.microsoft.com/downloads/details.aspx?familyid=b769a4b8-48ed-41a1-8095-5a086d1937cb&displaylang=en

We are also changing RAD-PADs to not use the DHTMLEdit control, so a Hotfix for the VLF should also be available soon.

Deploying VLF-Web applications

If you are having issues with setting up a deployed VLF-WEB application, the Administration Console, or include files, look for a file named **VF_Server_Trace.dat.**

- On an iSeries look in the root of the IFS (providing the user is authorized to write to the IFS root).
- On Windows it goes into the L4Web current directory, so use Search to look down through all the LANSA folders.

The content of this file is often revealing regarding configuration problems.
If you can find one, please forward it to support along with any problem report.

Planned Support of Microsoft Vista by LANSA

LANSA Software	Planned Support Date

32-bit Windows Vista
Validation that the 32-bit LANSA development environment and 32-bit LANSA developed applications will operate under 32-bit Windows Vista.

64-bit Windows Vista
Validation that the 32-bit LANSA development environment and 32-bit LANSA developed applications will operate under 64-bit Windows Vista

Support Status: Maintained (1)	On the GA of LANSA Version 11.4(2)
Support Status: Confirmed	After Vista Service Pack #1 is available

Notes:
- (1) There are known LANSA installation issues under Vista. Whilst the status is "Maintained", the installation of LANSA under Vista will require a set of special instructions. Contact your local LANSA support centre.
- (2) Only LANSA Version 11.4 and later will support Vista.
- (3) All references to Vista are to the *Windows Vista Business* version.

The support Status listed above are as per the LANSA Supported Platforms document available at Supported Versions

Status: Confirmed
Configuration has been comprehensively tested.
Response timeframes measured in accordance with the maintenance agreement.
Configuration components exist at Technical Support
Error fixes or workarounds provided for reported errors.

Status: Maintained
Configuration has not been comprehensively tested but substantial evidence suggests that it is expected to operate properly.
Response time frames may be delayed in some cases.
Configuration components exist at Technical Support
Error fixes or workarounds provided for reported errors.
Third Party software shipped with or an integral part of LANSA (ASA, Crystal Reports, **new**look and Mapforce) will also be subject to the same status points as the base LANSA software.

New LANSA Integrator version

EPC805 delivers enhancements and corrections to LANSA Integrator.

The following are some of the most notable items:

- Enhancements necessary to support the use of the new LANSA Integrator Composer modules.
- A variety of usability and functional improvements to the user-interface, including a new repository viewer for viewing the contents of the LANSA Repository.
- Various performance improvements including a new JSM pool server for distributing JSM workload across multiple JSM instances on the same machine or across multiple machines.
- A variety of usability and functional enhancements to XML and SOAP wizards and services.
- Enhancements to the PDF document service.
- A new JSON Wizard and an accompanying HTTPInboundJSONService for sending JSON (Javascript object notation) messages. JSON is or can be used as an alternative to XML and other formats in some applications.

Other new services including:

- OpenLDAPService
- XMLBindQueueService
- HTTPInboundXMLBindService
- HTTPInboundQueryService
- SMTPMailAttachmentSignatureService

Which Newlook version should I be using with the Visual LANSA Framework and RAMP?

Question

Am I using the correct version of Newlook if I have installed Newlook V7 and EPC804?

Answer

No. EPC804 has a pre-requisite of Newlook Version 8.0.1.10669 (dated March 14, 2007 or later)
- see EPC804.htm (http://www.lansa.com/support/notes/epc/epc804.htm) and VLF RAMP guide under the topic "Install Newlook" for further information.

In summary:
EPC793 Visual LANSA Framework
Newlook Version 7 dated 13th June 2006 or later.
EPC804 Visual LANSA Framework
Newlook Version 8.0.1.10669 dated March 14, 2007 or later.

You can register to download the latest version of Newlook either using the link in the RAMP guide or using this link:
www.lansa.com/products/newlookevaluation.htm
Ensure you have installed all required licenses to use RAMP with Newlook - further details are described in the VLF RAMP documentation under "Licensing Requirements".

Needing to receive LIST longer than 256 in a RDML function

Issue

How do I receive a list which needs to be greater than 256 in length in JSM using a JSM Command. This is a generic question that may apply various commands and services where lists are passed/received.

Solution

The best way to do this in RDML is to split the receive functionality. For obvious reasons RDMLX functions do not have this restrictions.

The following was performed using the XMLPARSERSERVICE. You don't need to have multiple XSL's to send and receive a large list. XSL's do not have the size restrictions thats lists in RDML have. The same XSL can be used. See below

********** split 1
CHANGE FIELD(#JSMCMD) TO('TRANSFORM XSL(INBOUNDORDER) **SERVICE_LIST(LINENUM,PARTNUM,PARTDSC)**')
USE BUILTIN(JSM_COMMAND) WITH_ARGS(#JSMCMD) TO_GET(#JSMSTS #JSMMSG **#WRKLSTX**)
EXECUTE SUBROUTINE(CHECK) WITH_PARMS(#JSMSTS #JSMMSG)

********** split 2
CHANGE FIELD(#JSMCMD) TO('TRANSFORM XSL(INBOUNDORDER) **SERVICE_LIST(PARTAMT,PARTQTY)**')
USE BUILTIN(JSM_COMMAND) WITH_ARGS(#JSMCMD) TO_GET(#JSMSTS #JSMMSG **#WRKLSTY**)
EXECUTE SUBROUTINE(CHECK) WITH_PARMS(#JSMSTS #JSMMSG)

Using the PDFDocumentService to write to an already existing PDF

How do I write to an already existing PDF document? It doesn't matter how the PDF was created. It could have been created using a third party tool or even the LANSA Integrator itself. What do I need to do to overwrite/amend/add text to the existing PDF document.

There are 2 aspects to this.

1. The LANSA RDML function will be used dynamically set a field value. The value will be some piece of text (static or dynamic) that the user wishes to insert in the PDF, for example customer name.

The RDML should be based on the following
==========
Firstly, define a list with the field names containing the text that you wish to use in the PDF document. This List will be exchange in the ADD command.

```
DEF_LIST NAME(#OUTPUT) FIELDS((#SNAME)) TYPE(*WORKING)
CHANGE FIELD(#SNAME) TO('My Company')
```

add the field to the List

```
ADD_ENTRY TO_LIST(#OUTPUT)
```

build the ADD command to add a particular content to a section of the PDF document. This section of the PDF will actually IMPORT the already created PDF document and then on a page of the PDF document the text will be added. See XML source in point 2.

```
USE BUILTIN(JSM_COMMAND) WITH_ARGS('ADD CONTENT(MYCONTENT)
SERVICE_LIST(SNAME)') TO_GET(#JSMSTS #JSMMSG #OUTPUT)
EXECUTE SUBROUTINE(CHECK) WITH_PARMS(#JSMSTS #JSMMSG)
```

==========

Note: you need to use a SERVICE_LIST keyword so that the field value is exchanged.

2. The second part is the XML source where the external PDF document is imported and amended with additional text

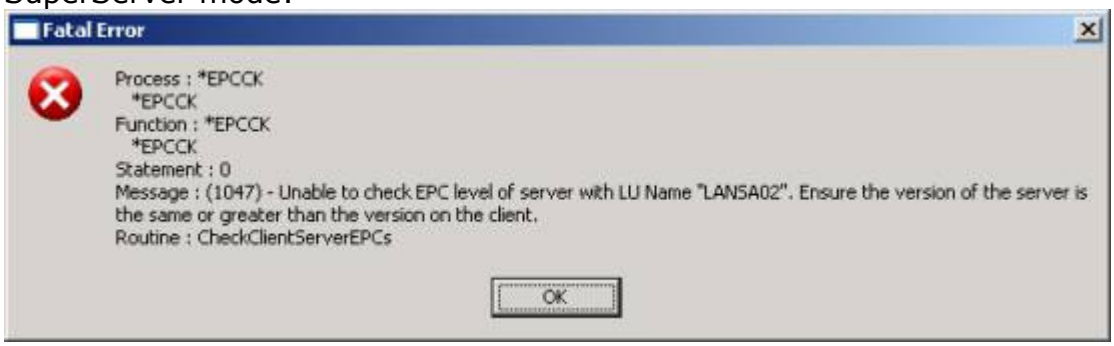
```
==================================
<content name="mycontent">

<text x1="382" y1="755" width="150" height="60" style="text-modern"
border="" align="left" leading="">

<phrase style="standard" options="na" value="{SNAME}"/>

<phrase style="standard" options="na">Phone: 8907 1111</phrase>

<phrase style="standard" options="na">Fax: 1223 9999</phrase>

</text>

</content>
==================================
```

Note: In the above XML source there are 3 instances of new text being added. The first line is inserting the contents of a field from a LANSA function; field called SNAME. This is a soft value. The next two are actually literal inserts. It is hard-coded in the XML.

An *EPCCHK error when using *JOB *JOB as Translation Tables in VLF

EPC804 Visual LANSA Framework defaults Translation Tables values for an RDMLX partition to *JOB *JOB. This can be seen in the Server definition window. As stated in the chapter - "Whats New" in the Visual LANSA Framework documentation shipped with EPC804, these are the default values for an RDMLX partition only.

If you manually define the translation tables as *JOB *JOB for an RDML partition, you may generate the following the next time you login to the Framework in SuperServer mode:



This is a misleading message for this situation.

Solution

Change both values back to your required values (e.g. QEBCDIC and QASCII for an English system) in the VLF Server definition window. Save and restart the Framework.

Note: If you ever generate this message, you should review the server joblogs to see what the actual problem as described in the joblog is. Under normal circumstances this message indicates there is a mismatch in the EPC levels between the client environment and the server environment. This may indeed be the problem. But in some cases this can be a misleading message. The misleading EPC check error message will be corrected in future versions.

WAMs - Known causes for Unrecoverable Error Occurred

Symptoms

WAM applications fail and have error messages that contain the text:
UNRECOVERABLE ERROR OCCURRED

There are many variations of this error, and there can be various possible causes for the problem.

Listed are some of the known causes for some of these errors!

1.
Unrecoverable error occurred while executing the Webroutine

Message text: Failed to connect to the LANSA System assigned to Handle request from Web server "hostname:port" (Configuration "port" has been used) with error ***code 14 (ERR_LCO)! Message index:<N/A>***

Possible Cause:

- LANSA Listener not running
- User password may have expired.
- System IP address setup may be incorrect
- Ensure that there is no firewall is switched on for all ports

2.
Unrecoverable error occurred at WpiSystemManager::ProcessLansaWebRequest

Message text: Failed to connect to the LANSA System assigned to handle request from Web server "localhost:80" (Configuration "80" has been used) with ***error code 14 (ERR_LCO)! Message index:<Not specified>***

Possible Cause:

- Ensure the system configuration details are correct in Web Administrator
- Check that the lroute.dat path is correct
- Restart the IIS and WWW services
- Ensure the User has the correct library list and/or job description
- The User password may have expired.

3.
Unrecoverable error occurred at
WpiSystemManager::ProcessLansaWebRequest

Message text: Failed to connect to the LANSA System assigned to Handle request from Web server "hostname:port" (Configuration "port" has been used) with error code 10003 (ERR_CLASS_SPEC)! Message index: <Not specified>

or

Unrecoverable error occurred while processing the Web request

Message text: Failed to access the application server. Message index:<N/A> Further Information: CM_UNSUCCESSFUL

Possible Cause:

- Check if there is a Code page specified in the Web Administrator is for a Windows backend system. This parameter should only be specified for iSeries machines.

4.
Unrecoverable error occurred at
WpiSystemManager::ProcessLansaWebRequest

Message text: No LANSA System has been assigned to handle request from this Web server! Message index: <Not specified>

Possible Cause:

- Ensure the system configuration details are correct in Web Administrator.

Cascading Style Sheets and WAMs

CSS provides a powerful tool to modify the appearance and behaviour of lists and grid controls. This document explains how to add your own CSS styles to a WAM and how to use CSS to change the appearance of the standard list and grid.

Adding your own CSS

The Standard Style weblet provides two ways to apply your own CSS stylesheets to your WAMs: theme_css_filename and css_files.

Theme_css_filename is intended to be the main CSS file used by your WAM and is loaded immediately after *std_styles.css*. Std_styles.css provides all the default styles required for correct operation of the LANSA supplied weblets. Your theme CSS is then loaded over the top of *std_style.css*. So your theme CSS does not need to contain a copy of everything in *std_style.css*. It contains just the styles and properties you want to modify.

For example the class .std_menutop_bg (standard menu top background) is defined in *std_style.css* as:

```
.std_menutop_bg
{
	background-color: #7db0e5;
	height: 20px;
}
```

This same class is defined in the style sheet *theme_grass.css*.

```
.std_menutop_bg
{
	background-color: #e6e6dc;
	border-bottom: 1px solid #cbcbb8;
}
```

The theme contains one modified property and one new property. The combination of these produces an effective style of:

```
.std_menutop_bg
{
	background-color: #e6e6dc;
	height: 20px;
	border-bottom: 1px solid #cbcbb8;
}
```

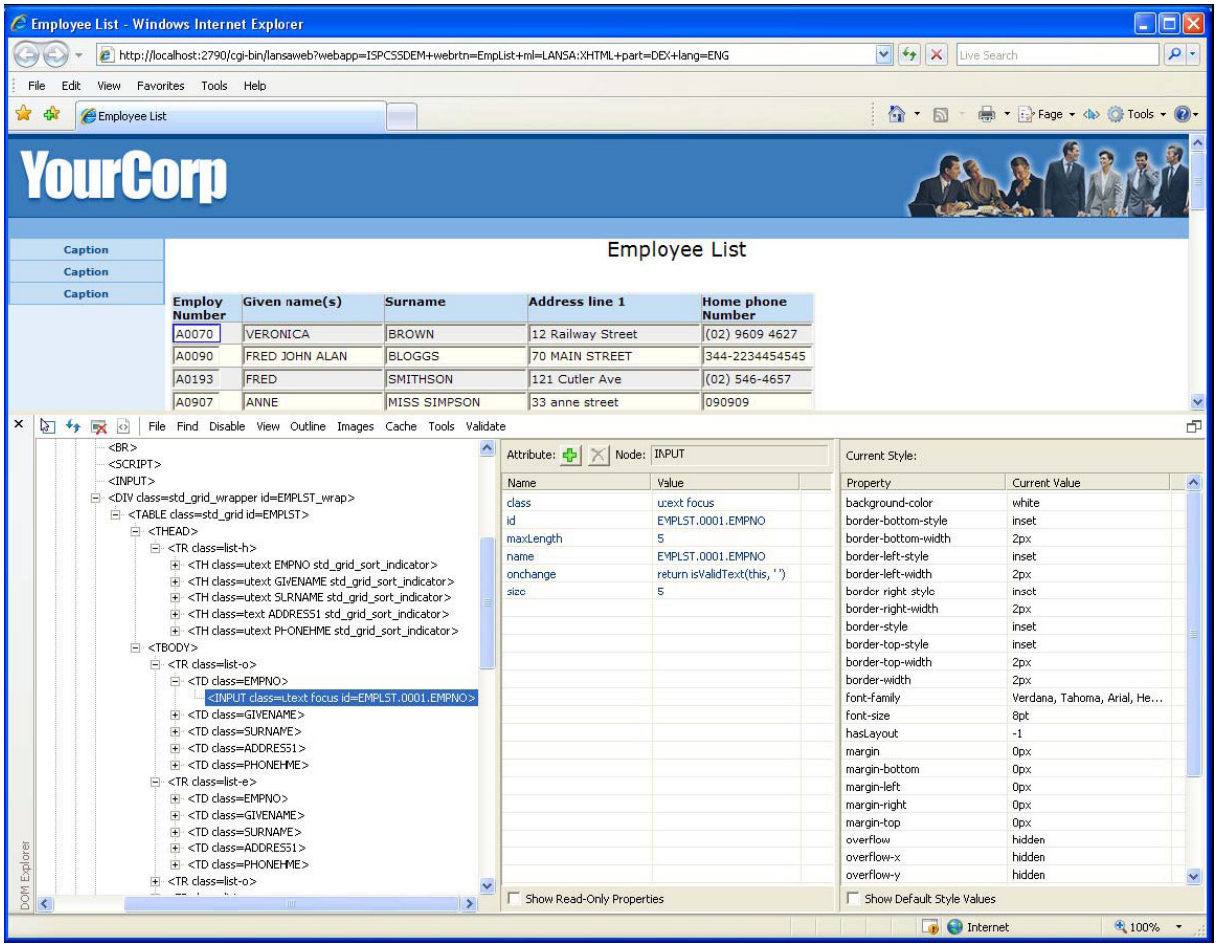
Next, the files defined in *css_files* are loaded. The *css_files* property takes a comma delimited list of file names and loads them in the order supplied. This provides a useful mechanism for adding files required by third party widgets, or to modify styles on a per webroutine basis.

So, to create your own CSS styles, create a new CSS file and add it to your layout or Webroutine by specifying it in either the *theme_css_filename* or *css_files* property. Do not modify the *std_styles.css* file.

Using CSS with the Grid

CSS provides a powerful tool to modify the appearance and behaviour of lists and grid controls. To use it effectively, you must first understand how the grid is constructed. If you are using Internet Explorer 7 then download the free Developer Toolbar from the Microsoft web site.
Tools like this are essential for understanding how your page is constructed.

Here is a standard grid visualizing the list EMPLIST, containing the columns EMPNO, GIVENAME, SURNAME, ADDRESS1 and PHONEHME. The bottom section of the window is the Developer Toolbar pane showing how the grid is constructed:



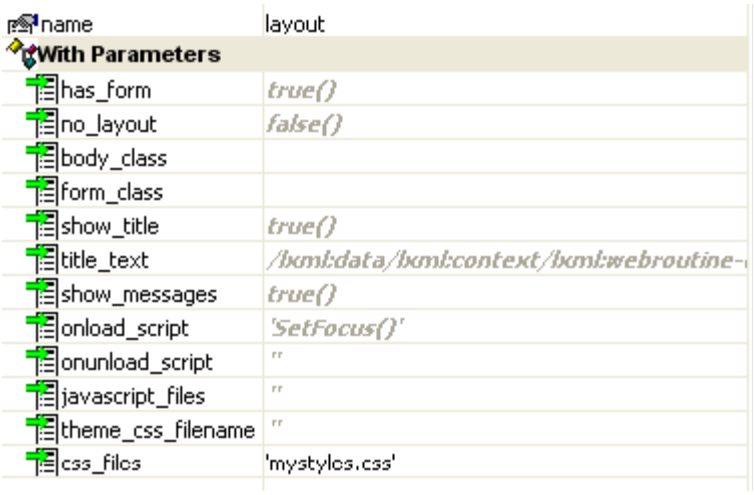
Here is another way of looking at the same information:

Some points to note:

- Each grid is wrapped by a DIV with the class "gridname_wrap". For a standard list, *gridname* is the name of the list. For a grid, *gridname* is the value supplied in the name property. The grid wrapper provides the size and position for the grid, drawing any scrollbars as necessary.
- In a list, the row class names are alternated between "list-o" and "list-e". For a grid, the names provided in *odd_row_class* and *even_row_class* properties are used ("odd_row" and "even_row" are the defaults).
- Table cells are given the name of the column as a class name. This provides a way to apply styles to specific columns.
- Input fields are given a class that represents the data type of the field. Fields of type alpha, char or varchar will be "text", "utext" or "ltext" depending on the input case of the field. Fields of type packed, signed, integer, float, date, time or datetime will have a class name of "number". Boolean fields will be "ltext" and all other fields will be "text".

The best way to understand how it all fits together is to look at a few examples.

Before you start, create a new .css file, place it in your styles directory, and add it to your layout using the css_files property:



Let's start by looking at how the alternating row colours are created. The following two styles define the odd and even row classes in a standard list:

```
tr.list-o
{
background: #eeeeee;
color: black;
}
tr.list-e
{
background: #fffff4;
color: black;
}
```

To override these with your own colors, simply redefine them in your style sheet. For example:

```
tr.list-o
{
background: red;
color: white;
}
tr.list-e
{
background: blue;
color: white;
}
```

The result of the above is this:

Employ Number	Given name(s)	Surname	Address line 1	Home phone Number
A0070	VERONICA	BROWN	12 Railway Street	(02) 9609 4627
A0090	FRED JOHN ALAN	BLOGGS	70 MAIN STREET	344-2234454545
A0193	FRED	SMITHSON	121 Cutler Ave	(02) 546-4657
A0907	ANNE	MISS SIMPSON	33 anne street	090909
A1001	BEN	JONES	144 Frog	799 5268
A1002	JOHN	SMYTHE	20 Cobbitty Avenue,	047 629 0442
A1003	ROBERT	SMITHE	29 Arthur Road,	977 6268
A1004	PAUL	SMITHSON	41 William Road,	419 5656
A1005	PETER	SMITHS	72 Mullane Avenue,	674 4316
A1006	JACK	SMITHERS	8 Croydon Road,	799 3638

Notice that the text in the list is black even though we set the *color* to white. This is because input fields have a default color of black. If this was an output list the above would be fine as most HTML elements will inherit the colour value from their parent. But the INPUT sets its own default, ignoring the colour defined by its parent. A style entry like this will resolve the problem:

```
tr.list-o input,
tr.list-e input
{
color: white;
}
```

This says to set the color of all INPUTs contained within a TR with a class of "list-o" and all INPUTs contained within a TR with a class of "list-e" to white.

The styles defined above will target every list on every page that uses the css file. If you only want to target a particular list then you need to make the selectors more specific. Looking back at the grid structure you can see that the grid TABLE has an id equal to the name of the list.

You can use this to apply your styles only to the EMPLIST list. Here is an example of an output list with a background color and no alternating row colors:

```
#EMPLST
{
background-color: #F5E56C;
}
#EMPLST tr.list-o,
#EMPLST tr.list-e
{
background-color: transparent;
}
```

Employ Number	Given name(s)	Surname	Address line 1	Home phone Number
A0070	VERONICA	BROWN	12 Railway Street	(02) 9609 4627
A0090	FRED JOHN ALAN	BLOGGS	70 MAIN STREET	344-2234454545
A0193	FRED	SMITHSON	121 Cutler Ave	(02) 546-4657
A0907	ANNE	MISS SIMPSON	33 anne street	090909
A1001	BEN	JONES	144 Frog	799 5268
A1002	JOHN	SMYTHE	20 Cobbitty Avenue,	047 629 0442

The field name class assigned to each table cell gives you a handy way to target specific columns. For example, change the background color of the EMPNO column:

```
#EMPLST .EMPNO {
background-color: #6C93F5;
}
```

Employ Number	Given name(s)	Surname	Address line 1	Home phone Number
A0070	VERONICA	BROWN	12 Railway Street	(02) 9609 4627
A0090	FRED JOHN ALAN	BLOGGS	70 MAIN STREET	344-2234454545
A0193	FRED	SMITHSON	121 Cutler Ave	(02) 546-4657
A0907	ANNE	MISS SIMPSON	33 anne street	090909
A1001	BEN	JONES	144 Frog	799 5268
A1002	JOHN	SMYTHE	20 Cobbitty Avenue,	047 629 0442
A1003	Robert	SMITHE	29 Arthur Road,	977 6268

If you want to target the body and leave the header alone, make the selector more specific:

```
#EMPLST TBODY .EMPNO {
background-color: #6C93F5;
}
```

Employ Number	Given name(s)	Surname	Address line 1	Home phone Number
A0070	VERONICA	BROWN	12 Railway Street	(02) 9609 4627
A0090	FRED JOHN ALAN	BLOGGS	70 MAIN STREET	344-2234454545
A0193	FRED	SMITHSON	121 Cutler Ave	(02) 546-4657
A0907	ANNE	MISS SIMPSON	33 anne street	090909
A1001	BEN	JONES	144 Frog	799 5268
A1002	JOHN	SMYTHE	20 Cobbitty Avenue,	047 629 0442

Now it's your turn. As long as you are applying a custom CSS over the default, you can't break anything, so feel free to experiment. Here are a few other points to note.

- If you have not done much with CSS before, take a look at the tutorials at www.w3schools.com.
- You may also want to take a look at these articles on CSS Selectors: http://www.456bereastreet.com/archive/200509/css_21_selectors_part_1/
- In the event of a conflict, the style with the more specific selector will take priority. For example, the default style for grid table cells is defined with ".std_grid tbody td". It will override any conflicting properties defined with ".std_grid td". So, if a style isn't working as expected, try making it more specific.
- With the exception of the tr.list-o and tr.list-e styles shown earlier, the default selectors for most grid related styles start with the .std_grid class selector. This makes them easier to find in the CSS file and reduces the chances of accidental conflicts with styles used elsewhere (the tr.list-o and tr.list-e are defined as they are for backwards compatibility reasons).

New X_RUN printer options

EPC800 delivers new X_RUN printer options!

WPPN (Windows Printing Printer Name)

Specifies the full name of the printer. If specifying a network printer, the domain name must also be included. That is, WPPN=\\domain\printer name. For example, WPPN=\\ourdomain\Epson Stylus COLOR 900. The LIST_PRINTERS BIF returns the full printer name as required by this parameter. This parameter is not passed to (inherited by) a server system. If printing from a server in a client/server environment, retrieve the list of printers defined on the server by calling a function on the server that uses the LIST_PRINTERS BIF. The selected printer's name can be exchanged back to the server when the print function is called.

For example:

```
************************************************
** CLIENT SIDE FORM
************************************************
FUNCTION OPTIONS(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_FORM) CLIENTHEIGHT(240) CLIENTWIDTH(492)
HEIGHT(267) LEFT(378) TOP(238)

* DEFINE_COM commands appear here to define controls on dialog.

DEF_LIST NAME(#PRNLIST) FIELDS(#PRN_NAME #PRN_LOC) TYPE(*WORKING)

EVTROUTINE handling(#com_owner.Initialize)
SET #com_owner caption(*component_desc)

USE BUILTIN(SET_SESSION_VALUE) WITH_ARGS(USER MyUserid)
USE BUILTIN(DEFINE_ANY_SERVER) WITH_ARGS(MYSERVER Server1 Y)
TO_GET(#STD_CMPAR)
USE BUILTIN(CONNECT_SERVER) WITH_ARGS(MYSERVER 'MyPassword')
TO_GET(#STD_CMPAR)
USE BUILTIN(CALL_SERVER_FUNCTION) WITH_ARGS(MYSERVER GETPRNS Y Y
#PRNLIST) TO_GET(#STD_CMPAR)

SELECTLIST NAMED(#PRNLIST)
ADD_ENTRY #LTVW_1
ENDSELECT

CHANGE FIELD(#DEPTMENT) TO(FLT)

ENDROUTINE
```

```
EVTROUTINE HANDLING(#PHBN_PrintEmplistD.Click)

EXCHANGE FIELDS(#DEPTMENT #PRN_NAME) OPTION(*ALWAYS)

USE BUILTIN(CALL_SERVER_FUNCTION) WITH_ARGS(MYSERVER SRVEMPL Y Y)
TO_GET(#STD_CMPAR)

ENDROUTINE

END_COM


************************************************
** SERVER SIDE FUNCTION
************************************************
FUNCTION OPTIONS(*DIRECT) RCV_LIST(#PRNLIST)
DEF_LIST NAME(#PRNLIST) FIELDS(#PRN_NAME #PRN_LOC) TYPE(*WORKING)
USE BUILTIN(LIST_PRINTERS) WITH_ARGS(A) TO_GET(#PRNLIST #STD_CMPAR)
```

WPPS (Windows Printing Setup File)

Specifies the file (including full path) that contains printer settings. These settings will be used to initialize the settings on the printer dialog. If the printer dialog will not be displayed, these settings will automatically be sent to the printer. The default printer settings on the PC will be used if WPPS is not specified. Likewise, if WPPS has been specified, the default printer setting will be used for those parameters not specified in the file.

A new file can be specified at any time thus allowing the user to use different settings for different print jobs (reports, etc) without having to display the printer dialog every time.
This parameter is not passed to (inherited by) a server system. If printing from a server in a client/server environment, the file name can be exchanged to the server when the print function is called. Alternatively the file can specified in the server function.

The following printer settings may be specified. Some settings may not be relevant for the selected printer. Any settings that are not relevant will be ignored by the printer.

1. PGOR= Page Orientation

Value	Description
P	Portrait
L	Landscape

2. DPXP= Duplex printing

Value	Description
N	Normal (nonduplex)
H	Duplex over horizontal (flip over the long edge of the page)
V	Duplex over vertical (flip over the short edge of the page)

3. COLR= Color printing

Value	Description
C	Colour
M	Monochrom e

4. COPY= Number of copies

5. PSIZ= Paper Size

Value	Description
LETTER	Letter, 8 ½ by 11-inches
LEGAL	Legal 8 ½ by 14 inches
A4	A4 sheet, 210 x 297 millimeters
A3	A3 sheet, 297 by 420 millimeters
xxx	*xxx* is a numeric value defined by the Microsoft Visual C runtime which represents a specific paper size. The complete list of values may be obtained from Microsoft's MSDN library at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/intl/nls_Paper_Sizes.asp. Alternatively, see below for instructions on how to retrieve printer specific values.

6. PSRC= Paper Source

Value	Description
A	Auto
L	Lower tray
M	Middle tray
xxx	*xxx* is a numeric printer specific value. See below for instructions on how to retrieve printer specific values.

7. QLTY= Print Quality

Value	Description
H	High
M	Medium
L	Low
D	Draft
xxx	*xxx* is a numeric printer specific value which specifies the number of dots per inch (DPI). See below for instructions on how to retrieve printer specific values.

8. COLL= Collation

Value	Description
Y	Collate when printing multiple copies
N	Do not collate when printing multiple copies

9. PTYP= Paper Type

Value	Description
P	Plain paper
G	Glossy paper
T	Transparent film
xxx	*xxx* is a numeric printer specific value. See below for instructions on how to retrieve printer specific values.

10. FNAM= Form Name

See below for instructions on how to retrieve printer specific values.

Printer specific values can be obtained by running the application with WPPD=E and trace settings ITRO=Y ITRL=9 and ITRM set to a value large enough to log the required tracing information. Enter the required values into the print dialog.

The values returned by the print dialog will be logged in the trace file. Only values that are relevant to the current printer are logged. Open the trace file and search for the phrase "Printer Details".

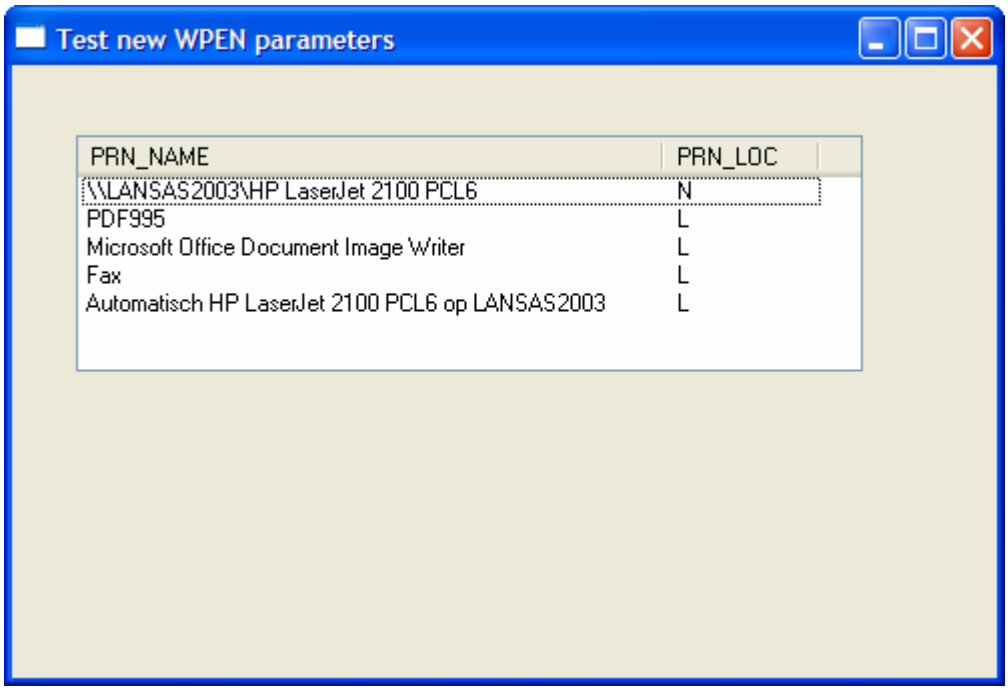
The following is an example of the trace output:
MESSAGE : Printer Details for \\syd1\HP LaserJet 4050 Series
MESSAGE : Printer Details... PGOR=1 DPXP=1 COLR=2 COPY=2 PSIZ=9 PSRC=2 QLTY=600

Examples of a printer setup files:
Example 1:
PGOR=P
DPXP=V
COPY=4
Example 2:
PGOR=L
DPXP=N
COLR=C
PSIZ=33

New LIST_PRINTERS BIF

EPC800 delivers a new BIF called LIST_PRINTERS.

Create these two fields in the Repository:
PRN_NAME Alpha 256
PRN_LOC Alpha 1



Copy/paste source below into a new form and compile the form:

```
Function Options(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_FORM) CLIENTHEIGHT(306) CLIENTWIDTH(492)
LEFT(693) TOP(255)
DEFINE_COM CLASS(#PRIM_LTVW) NAME(#LTVW_1) COLUMNBUTTONHEIGHT(20)
COMPONENTVERSION(2) DISPLAYPOSITION(1) FULLROWSELECT(True) HEIGHT(118)
LEFT(32) PARENT(#COM_OWNER) SHOWSORTARROW(True) TABPOSITION(1) TOP(35)
WIDTH(393)
DEFINE_COM CLASS(#PRIM_LVCL) NAME(#LVCL_1) DISPLAYPOSITION(1)
PARENT(#LTVW_1) SOURCE(#PRN_NAME) WIDTH(67)
DEFINE_COM CLASS(#PRIM_LVCL) NAME(#LVCL_2) DISPLAYPOSITION(2)
PARENT(#LTVW_1) SOURCE(#PRN_LOC)

Evtroutine Handling(#com_owner.Initialize)
Set Com(#com_owner) Caption(*component_desc)
Def_List Name(#PRNLIST) Fields(#PRN_NAME #PRN_LOC) Type(*WORKING)
Use Builtin(LIST_PRINTERS) With_Args(A) To_Get(#PRNLIST #STD_CMPAR)
Selectlist Named(#PRNLIST)
Add_Entry To_List(#LTVW_1)
Endselect
Endroutine
End_Com
```

Identify which XSLT Processor is being used on iSeries for WAMs

There may be a need or it may be useful to be able to identify which XSLT processor is being used in your WAM applications. To determine this, in your XSLT stylesheet, add the following:

```
<table>
<tr>
<td>XSLT Processor:</td>
<td><xsl:value-of select="system-property('xsl:vendor')" /></td>
</tr>
</table>
```

iSeries

On the iSeries, if you have a joblog of the LWEB_JOB, to see which XSLT processor ran, refer to the message for the PASE startup:

1. ILE XSL4C = No message
2. Apache Xalan = PASE started (no bit mode details in 2nd level message). This applies to V11.0 GA.
3. libxslt = PASE started in 32-bit mode
4. Apache Xalan = PASE started in 64-bit mode