# What's New in LANSA Version 13 Service Pack 2?

The second Service Pack for LANSA Version 13 includes new features and some enhancements.

Web Application Module Enhancements	Various WAM enhancements are introduced in SP2.
Windows 64-bit Support	Visual LANSA now supports the generation and compilation of both Windows 32-bit and 64-bit applications.
IBM i User Profile Handling	IBM i user profiles can now be validated, and IBM i passwords can be changed using the IBM special APIs with interfaces to RDML and LANSA Open.
SuperServer Enhancements	SuperServer connections can now be made between all supported platforms: IBM i, Windows and Linux.
IDE Enhancements for IBM i Administrators	Export lists can be generated automatically when checking in or delivering objects to an IBM i server, and objects can be refreshed selectively in the Repository.

Before deploying your Version 13 applications, please take time to thoroughly understand the new MSI Deployment mechanism.

Note that Version 13.0 iSeries Exports cannot be imported to V13.0 SP1 or later versions.

This document also contains What's New in LANSA Version 13? and What's New in LANSA Version 13 SP1?

Edition Date May 30, 2014 © 2014 LANSA

# **Web Application Module Enhancements**

LANSA Version 13 Service Pack 2 introduces many new features and enhancements for WAMs:

jQuery Mobile WAM Enhancements Support for file uploads to a webroutine New XHTML Weblets Upgraded Third-Party Libraries WAM Editor Enhancements

### jQuery Mobile WAM Enhancements

Commonly used jQuery Mobile weblets have been made easier to use and various new weblets have been introduced.

Easier to Design jQuery Mobile WAMs New jQuery Mobile Weblets Other improvements to jQuery Mobile



Note that the jQuery Mobile look and feel has changed:

- New Flat UI
- Two themes: Light and Dark
- Themeroller based theming
- Widgets can be pre-rendered for better performance.

### Easier to Design jQuery Mobile WAMs

Simplified default versions of a number of jQuery Mobile Weblets have been introduced for ease of use.

These include the button (std\_button\_s1):



And anchor (std\_anchor\_s1) weblets:

Carrier 荣	3:00 AM	-
	Welcome	
Go to Page 2		
Listing Page		
50	alas Capitant	
Fo	oter Content	

Use the previous versions of these weblets (std\_button\_v2 and std\_anchor\_v2) only if you need to add content to them.

#### New jQuery Mobile Weblets

New jQuery Mobile weblets are available:

#### Autocomplete

Maine	O
Maryland 😓	O
Massachusetts	Ø

Register

Speakers

Presentations

Organizers

The Autocomplete weblet provides suggestions while you type into the field. The suggestions are provided by a webroutine using Ajax.

The Image weblet displays an image.

The weblet has an option to load the image only when it comes into view, which helps render the page faster.

## Loader

Image



#### **Progress bar**



The Loader displays a small loading overlay when jQuery Mobile loads in content via AJAX, or when you want to perform an action that momentarily blocks user interaction.

Progress bar displays the status of a determinate process. It can also be used to display a value as a percentage of its maximum value.

### Other improvements to jQuery Mobile

#### **Input Box Weblet**

The Input Box weblet now supports input type="number" to bring the correct keyboard to mobile devices.



#### Generated Lists for jQuery Mobile

Generated lists for jQuery Mobile are now similar in structure to the lists generated for Technology Service XHTML. You have access to columns by variable name.

Previously the XHTML and jQuery Mobile list methods were different, with the jQuery Mobile's method giving more flexibility in adding content to the list entry at the expense of making it less easy to use in the WAM Editor.

You can still use the more flexible and complex method by using the jQuery Mobile std\_html\_list weblet.

#### Support for File Uploads to a Webroutine

You can use the file upload weblet to select files to upload to the application server (into a temporary directory). The webroutine that receives the file upload can then manipulate the uploaded files as required.



For more information, see the description of the XHTML File Upload (std\_fileupload) weblet and for the jQuery Mobile File Upload (std\_fileupload) weblet.

#### New XHTML Weblets

New XHTML Mobile weblets are available:







The Image weblet displays an image.

The weblet has an option to load the image only when it comes into view, which helps render the page faster.

The Loader displays a small loading overlay when jQuery Mobile loads in content via AJAX, or when you want to perform an action that momentarily blocks user interaction.

Progress bar displays the status of a determinate process.

It can also be used to display a value as a percentage of its maximum value.

## **Upgraded Third-Party Libraries**

Third-party libraries have been upgraded:



#### **WAM Editor Enhancements**

The WAM Editor now inspects the design of a web routine for use of deprecated weblets and lets the user know if they are using deprecated weblets. This test is done when a web routine's design is opened in the WAM Editor.

# Windows 64-bit Support



Visual LANSA now supports the generation and compilation of both Windows 32-bit and 64-bit LANSA applications.

- A 64-bit Visual LANSA runtime is provided in addition to the 32-bit runtime
- 64-bit deployments are supported in addition to 32-bit deployments

When Should Windows 64-bit Support be Enabled?
Installation Considerations
Programming Considerations
32-bit and 64-bit Applications Accessing the Same Database
Notable Environmental Differences

### When Should Windows 64-bit Support be Enabled?

We recommended that Windows 64-bit support is only enabled when there is corporate requirement for it.

Windows 64-bit support should only be installed on a Build machine, not developer machines.

#### Drawbacks

Using Windows 64-bit support has some drawbacks:

- You must obtain your own 64-bit compiler, either Visual Studio 2010 Professional (or later) or Visual Studio 2012 Express for Desktop (or later).
- Compile times are longer because both 32-bit and 64-bit DLLs are always built.
- Functions which use DISPLAY, REQUEST or POPUP commands will fail to compile also in 32-bit DLLs.

#### Features not Supported

There are LANSA features that do not function or are not supported in 64-bit applications:

- Graphics Server
- Web Functions
- ZIP and specialized LANSA Built In Functions (BIFs)
- Explorer Component AutoRefresh Property

#### **Installation Considerations**

It is presumed that 64-bit support is only enabled on the Build machine and that developers do not enable it. As a consequence, when 64-bit support is enabled both the 32-bit and 64-bit compiles are performed and both MSI packages are built.

You cannot compile a function which contains DISPLAY, REQUEST and POPUP commands - even the 32-bit compile will fail. This is why it is better not to enable 64-bit support on Developer's machines. If developers need to work on both RDML functions and 64-bit applications, two systems which use the same repository can be installed on their machines

#### **Compiler Installation**

If a supported compiler is not installed before LANSA, the LANSA-shipped compiler is installed and enabled. If you install a 64-bit compiler later, you need to change this registry entry to disable the LANSA-shipped compiler:

On a 64- HKEY\_LOCAL\_MACHINE\SOFTWARE\Wow6432Node\LANSA bit PC to 0.

On a 32- <code>HKEY\_LOCAL\_MACHINE\SOFTWARE\LANSA\MicrosoftComj</code> bit PC

If Visual Studio is installed before LANSA, it is used as the compiler.

If the version of Visual Studio installed does not support 64-bit compiling, install a version that does. LANSA will detect it when it is next started.

#### **Programming Considerations**

There are no increases in the maximum size of any LANSA feature because the limits are considered sufficient. This ensures greater compatibility between 32-bit and 64-bit applications. For example:

- The maximum size of an RDMLX List is still 2 billion rows, with each entry being 2 billion bytes long.
- The Built In Functions SND\_TO\_DATA\_QUEUE and RCV\_FROM\_DATA\_QUEUE may be used interchangeably.
- Job Queue Emulation can use either a 32-bit or 64-bit Job Queue Monitor and jobs may be submitted from either 32-bit or 64-bit. Note that the 64-bit Job Queue Monitor will execute the submitted job as 64-bit, no matter which platform submitted the job.

PC Other Files which are loaded using a 32-bit ODBC driver will need to create a 64-bit DSN with the same name as that used to load the file, or use CONNECT\_SERVER when deployed to re-direct IO to a 64-bit driver. An ActiveX included in LANSA RDML must be a registered 32-bit version. To execute the ActiveX, a version must be registered which is of the same processor architecture as the LANSA runtime. That is, if the LANSA runtime is 64-bit then the 64-bit ActiveX must be registered on the deployed PC.

### 32-bit and 64-bit Applications Accessing the Same Database

These considerations are particularly important when deploying an application into a production system:

Do not create mixed 32-bit and 64-bit applications	To avoid complexity, it is recommended that applications are either 32-bit or 64-bit. For example, if you use both 32- bit and 64-bit clients when using SuperServer, only use a 64-bit server. Because the clients are not directly accessing the database, there is no complication.
Auto-generate relative record numbers	Assign relative record numbers using auto-generation. If relative record numbers are assigned using external files, duplicates will occur unless the RPTH parameter is assigned to the same path for both 32-bit and 64-bit applications. A file that is currently using external files can be changed to use auto-generation using the Upgrade tool feature <i>Convert Files to Use Identity Column</i> .
Database upgraded by first system upgraded	Table upgrades are identified by comparing the previous CTD file to the new CTD file being installed. Thus only the first system upgraded should upgrade the database. This is why database upgrade defaults to off during an MSI install and why per-user installs disable database upgrade.
Be consistent	If an existing OAM is not there for 64-bit but is for 32-bit, and vice versa, the user needs to control which is the latest OAM. If 32-bit is the first environment to be installed, continue that way for all Upgrades and Patches. Once the 64-bit environment is at the same level, the Upgrade/Patch database change machine can be switched, but it is inadvisable. Be consistent and use one machine from the beginning.

#### **Notable Environmental Differences**

- The system directory for 32-bit applications is of the form x\_win95\x\_lansa. For 64-bit applications it is x\_win64\x\_lansa. Therefore system variables like \*SYS\_DIR return a different value.
- Visual LANSA is a 32-bit application. Hence interaction between Visual Lansa and 64-bit generated DLLs cannot occur.
- 32-bit OAMs are always built because Visual LANSA requires the 32-bit OAM to unload and load the data from the table. The 64-bit build command always skips the SQL table build, presuming that 32-bit has already done it.
- The Windows Installer has a known defect which converts the Target directory in a Shortcut from c:\program files to c:\program files (x86). Nonetheless, the shortcut still works correctly as if it was c:\program files. Even if the 32-bit version of the Application is installed in c:\program files (x86), it does not get executed, it is still the 64-bit version. See the MSDN forum post 32bit MSI on 64bit OS: Converting shortcut target path of 64bit app to 32 bit Path.
- A similar situation occurs with Windows\system 32. The shortcut looks OK but it does not find the object. It is not valid to create a shortcut that points to this directory.

# **IBM i User Profile Handling**

IBM i user profiles can now be validated, and IBM i passwords can be changed using the IBM special APIs with interfaces to RDML and LANSA Open.

Using this functionality non-5250 users can find out when their password is going to expire, and, if the password has not expired or been disabled, to change it.

This functionality is implemented in Visual LANSA as Built-In Functions, and in LANSA Open as Lce APIs. See:

- CHECK\_IBMI\_SIGNON and CHANGE\_IBMI\_SIGNON in the Technical Reference.
- LceGetIBMiSignon and LceSetIBMiSignon in the LANSA Open Guide.

# **SuperServer Enhancements**



# **IDE Enhancements for IBM i Administrators**

New Check-In Features Refresh Selected Objects

#### **New Check-In Features**

When objects are checked in or delivered to an IBM i system, an export list containing the selected objects can be automatically generated on the IBM i:

Check in Options		
ХА		
*	Process check in options	
<ul> <li>Processes &amp; Functi</li> <li>PSLINIX Create RD</li> <li>PSLTRG1 PSLEVENT</li> <li>PSLTRG2 PSLTIMES</li> <li>PSLUTLX Personne</li> <li>PSLVAL1 Validate e</li> <li>PSLVAR1 *DMXHO</li> <li>STDGUD STD GUII</li> </ul>	Image: Compile process/function         OMLX Sample D         T trigger         S trigger         I System Utilitie         event type         URLYRATE         D Field Level Tri	*
Keep Locks     Create export list on IBM i CKIn	Triggers Produce documentor details Generate HTML Validate numerics	Ca <u>n</u> cel

You can change the default name of the export list. If you specify the name of an existing export list, the new list will be appended to it.

For more information see Check In Options and Deliver To Options in the Visual LANSA Administrator's Guide.

## **Refresh Selected Objects**

Instead of doing a full refresh of all the objects in the repository, you can refresh the details of selected objects which are already in the local repository by right-clicking and selecting Refresh:



Object data is downloaded from the Master in batches and then trickled into the local Repository to avoid impacting performance on the local machine. You can use F5 at any time to force the objects to be refreshed immediately.

For more information see Refresh Master Object List in the Visual LANSA Administrator's Guide.

# What's New in LANSA Version 13 SP1?

The first Service Pack for LANSA Version 13 includes some enhancements and new features.

Visual LANSA usability	The new Table Layouts can be managed directly from the ribbon.				
enhancements	The management of <b>DirectX Styles</b> has been integrated into the ribbon.				
	You can choose which commands are included in the QuickAccess Toolbar.				
	You can customize the editor Status Bar.				
	The User-Designed Controls (UDC) have been made easier to use and there are some sample designs available.				
	A new <b>CRUD Wizard</b> generates a rich client application similar to the IDE interface.				
	New Images and Styles are available for use in rich client applications.				
	The context sensitivity of help text in the editor has been improved. The help is invoked using F1 (not F2 as previously).				
WAM enhancements	Webroutines support POST in JSON Format.				
	Weblets that send requests now have a new property, vf_wamevent, for VLF developers to set a WAM event.				
	Undo/Redo has been improved in the WAM Editor.				
Logical Modeler	The Logical Modeler now supports long file names.				
LANSA Open	LANSA Open includes support for Unicode, Boolean columns and long names.				

Before deploying your Version 13 applications, please take time to thoroughly understand the new MSI Deployment mechanism.

Note that Version 13.0 iSeries Exports cannot be imported to V13.0 SP1. This document also contains What's New in LANSA Version 13?

Edition Date April 1, 2014 © 2013 LANSA

#### **Table Layouts**

The new Table layouts can be managed directly from the ribbon:



The Table layout manager makes creating and managing application interfaces easy. In the majority of cases the older layout managers are no longer required. See Table Layout.

### **DirectX Styles**

DirectX styles can be managed from the ribbon:



See Dynamic Styles.

### **QuickAccess Toolbar**

The Quick access toolbar provides easy access to commonly used commands:

÷ + ⇒ • • • • • • • • • •	demcom01 - test - l	ANSA Edito	r*	
File Home Desig	gn Tools			
Repository Text Search Find	Views Open Objects	• Cor	mpile 🔛 Fi	ull Check 👻 uild
Repos	itory	5	Comp	oile 🖬
Repository			Design	Source Reposi
Repository		*	Fu Be	nction Opt gin_Com Ro Height(2 Define_Com Displa

You can now choose the commands it contains:



## **Status Bar**

	Customize Statusbar 🛛 🗙		
	🔚 Audit Stamps		
	Configuration		
	📲 Cursor Location		
	A Font		
	💯 Language		
	Partition		
	! Status Messages		
	Target Runtime		
	🔁 Task		
	😭 Technology Service		
	🏖 User		
DirectX   LANSA XHTML   ENG   PCXTASK   PCXUSER   D	EM   LANSASP1 *S   Courie		
📮 EN 🍪 gadgets 🔹 🔺 🏴	🛱 🌒 📶 🍀 12:32 PM 😽 14/05/2013		

You can choose the information displayed in the status bar:

### **User-Designed Controls**

Creating user-designed controls (UDC) has been made easier:



There are sample UDC designs available which you can use in your applications:



The F1 help text provides information about the control you are working with.

<b>\</b>	Tree (Prim_Tree)
	DirectX Only
	Tree is a member of a group of list related primitive controls referred to as User Designed Controls.
•	Tree organises individual design panels in to a hierarchical structure very similar to that of tree view ( $\#Prim_trvw$ ).
<b>6</b>	UDCs can be manipulated by the use of the typical LANSA list commands ADD_ENTRY, UPD_ENTRY etc. When entries are added to the list, an instance of the design is made, fields can be passed in, and a corresponding list item is made. UDCs control the position of the item within the list, manage selection, focus, expand/collapse etc., and communicate with the individual item designs through a series of predefined methods published on an interface specific to the type of UDC. For Prim_Tree this is #Prim_Tree.iTreeDesign
	All UDCs are use a parameterized type to define the class of the design to be created each time an entry is added. This is specified on the DEFINE COM as below

## **CRUD Wizard**

You can create LANSA applications modeled on the LANSA IDE using a simple wizard:

My Editor - Name_3 - Description_3 –					
File Home		a 🔇			
Ribbon       Views       Name     Description       Name,1     Description,1       Name,2     Description,3       Name,4     Description,5       Name,6     Description,6	Name_3 Description Description_3				
Name,7 Description,7 Name,8 Description,8 Name,9 Description,9 Name_10 Description_10 Proxy		Document			
Object Browser					

See Create an Application Using a Wizard in the Visual LANSA Developer Guide.

## New Images and Styles

New images are available for use in rich client applications:

Repository				Design Source
💼 Bitmaps 🔹				10 kg
📫 🗙 🔳 🔍 🗎				
X			-	
Item 🔹	Description		Build St	Up
<ul> <li>xImageView32</li> <li>xImageView16</li> <li>xImageStop32</li> <li>xImageStop16</li> <li>xImageSettings32</li> <li>xImageSettings16</li> <li>xImageSaveAll32</li> <li>xImageSave32</li> <li>xImageSave16</li> <li>xImageRepository16</li> <li>xImageRefresh32</li> </ul>	Views 32 View 16 Stop 32 Stop 16 Settings 32 Settings 16 Save All 32 Save All 16 Save 32 Save 16 Repository 32 Repository 16 Refresh 32 Perfore 16	2		

Two new visual styles are available to demonstrate DirectX capabilities:

XDXStyles	Style	/ Private	Mouseover	New Style	New Layout	Manager
🔗 Visual Style	]					
Name		Descr	iption			
<ul> <li>▷</li> <li>▷</li> <li>∅</li> <li>∅</li></ul>						*
<ul> <li>▷ </li> <li></li></ul>	Styles	Direc	tX Styles			
<ul> <li>✓ xDer</li> <li>✓ Y</li> <li>✓ Z</li> <li>✓ Ø Other</li> </ul>	noStyles 6	v Samp	oles Styles			-
					ж	Cancel

#### **POST in JSON Format**

Webroutines now support POST in JSON format:



See JSON Support in the Web Application Modules (WAMs) Guide.

# vf\_wamevent

Weblets that send requests now have a property *vf\_wamevent* for VLF developers to set a WAM event:

Name	Value
🔎 name	std_button_v2
With Parameters	
r 🖼 name	concat('o', position(), '_LANSA_18436')
re caption	Caption
r currentrowhfield	STDROWNUM
r currentrownumval	position()
🖼 submitExtraFields	document(")/*/lxml:data/lxml:json[not(@id)]
r 🖼 hide_if	False
r 🖼 formname	LANSA
🖼 pos_absolute_design	
🖼 width_design	
ᠠ height_design	
🖼 on_click_wamname	\$lweb_WAMName
on_click_wrname	
re protocol	
🖻 show_in_new_window	False
🖼 target_window_name	<xsl:if xmlns:xsl="http://www.w3.org/1999/XSL</td>
r disabled	False
r 🖬 title	
rest_class	**
🖼 presubmit_js	
r confirm	False
r confirmText	
🖼 tab_index	
r default_button	
r vf_wamevent	17
## **Undo/Redo**

Undo/redo has been improved in the WAM Editor. Undo/redo is available in the context menu:



#### And in the ribbon:

Web Page XSL     Muntime     Number     Numbe	
Design Source Repository Details Cross References Web Page XSL XML	Undo (Ctrl+Z) Undo the last action
LANSA	

## **MSI Deployment**

Before you deploy your Version 13 applications, please take time to thoroughly understand the new MSI deployment mechanism.

A version number needs to be compiled into every DLL. You need to set the version number in the Compile settings dialog and then *rebuild* the entire application. The whole application must be in a single Package Version – a single MSI. Do not try to put different parts of the application in different Package Versions.

General   Compile   General   Source   Source   Design   Opyright   Copyright   Copyright   Copyright   Copyright   Copyright   Trade Mark   Trade Mark   Comment			LANSA Settings	×
Compile Style	General General Source Design Oebug Debug Compile Style	Compile Product Name Product Version File Version Copyright Trade Mark Comment	LANSA Settings	<u>Q</u> K <u>C</u> ancel Apply Reset Reset All Reset Editor

For more information, see **Deployment** in this guide and **What's New in the Deployment Tool** in the LANSA Application Deployment Tool guide.

# What's New in LANSA Version 13?

Licensing	New Version 13 licenses are required for both IBM i and Windows.
Microsoft DirectX User Interface	The Microsoft DirectX user interface offers an engaging end-user experience and richer visualization of data in Visual LANSA.
Ability to Consume .NET Components	Your LANSA programs can consume .NET UI and non-UI components.
WAMs for Mobiles	You can now easily create Web Application Modules for mobile applications.
Internationalization with UNICODE	LANSA provides full Unicode support.
Version Control System Support	Phase 1 of LANSA's version control interface to 3 <sup>rd</sup> party versioning tools is available.
Deployment	LANSA applications can be packaged as standard MSI (Microsoft Installer) packages for deployment.
Long Names	Long names are allowed to enable 3 <sup>rd</sup> party integration and descriptive names.
RDMLX Enhancements	New intrinsics and primitives, language GET/SET commands etc. available.
Windows Centric Developmen	Various enhancements have been made to LANSA to reflect the fact that development centers more and more on the Windows platform.
File Enhancements	IBM I other files now support Unicode and binary and varbinary fields.
Change of Collection/Library on Import and Deployment	The file library or collection can be overridden on import and deployment.

Installation and Development	New installation and development features have been introduced to ensure ease of use and developer productivity.
Visual LANSA Framework	The new version of the Framework utilizes and showcases the DirectX user interface.

#### Version 13 Features

This version of the Framework utilizes and showcases the new features in LANSA Version 13, including the DirectX user interface.

#### **Customized Quick Finds**

The Quick Find box is a dialog that appears on the top right of the VLF window.

The current behaviour is to search a list of all business object captions. This can now be overridden so that the user searches a list of values that you control.

And when the user selects one of your values, you control what happens. Typically this would be a switch to a business object, or to an instance list entry in a business object, or a command handler for a business object.

If necessary you can also signal that the list of searched values should be rebuilt.

#### **Button To Switch Between Monitors**

A button has been added to allow users with multiple monitors to switch to the other monitor. The button is located on the bottom left of the Framework window.

#### **Automatic Command Handler Float Feature**

A new feature is available for frameworks where the user needs to see a full size instance list and a full size command handler at the same time.

The feature makes the command handlers automatically float off to a separate window when an instance list is clicked, or double-clicked.

If the user has two monitors, the command handlers can be made to automatically float to the other monitor.

This leaves a full size instance list in the original window and allows the user to resize their command handler window.

#### **Popup Panel Hints for Instance Lists**

If the framework is running in Direct-X mode, it is now possible to show a popup panel when the user hovers over an instance list entry. This panel can be used to give the user a quick overview of the item without opening any of the command handlers for that item.

The end-user is able to disable the feature by right mouse clicking on the instance list, if popups are not required.

#### **Small VLF-WIN Improvements**

When a user clicks on a cluster item in a tree view instance list, the Visual ID1 and Visual ID2 are available. Previously, only the items identifying keys were available.

When blank values are added to date instance list columns, the blank is displayed rather than the value of the previous instance list entry.

Improved sort order of business objects when a user selects a command that applies to multiple business objects.

LANSA Integrator

The latest LANSA Integrator is shipped in Version 13.

## Licensing

When installing Version 13 or upgrading to Version 13, you will be required to obtain new Version 13 specific licenses.

New licenses are required for both IBM i and Windows, this includes both server and client licenses and Hardware keys (Dongles).

Prior to upgrading to Version 13, you should send your CPU details to LANSA Licensing for new Version 13 licenses.

There is a new method of applying Server and Integrator licenses.

In Version 13 all your licenses can be consolidated into a single central license location for ease of management. You can consolidate various LANSA configurations on the same CPU or many licenses from various CPUs into one central location.

Refer to **Product Licensing** at the LANSA Support website for more details.

#### **Microsoft DirectX User Interface**

Visual LANSA version 13.0 introduces the DirectX rendering engine. DirectX is a collection of APIs embedded in to Windows that provide superior graphics capabilities, opening the door to a world of new design possibilities.



To see examples of what can be done with DirectX run the DirectX Demonstration Application.

With the new Visual LANSA user-designed controls, the developer has almost complete control over the user interface. For example:

- There are gradient colors, rounded corners, transparency and opacity.
- Dynamic styles and mouse events make it possible for the user interface to react as the mouse enters, hovers over or leaves a control.
- Context menus can be much more than just menus and animations allow for developers to add valuable visual feedback for the user.

New Visual LANSA IDE Dynamic Styles Brushes User-Designed Controls Table Layout Popup Panel (Prim\_PPNL) Scaling Taskbar Integration Animations Adopting DirectX

## **DirectX Demonstration Application**

Run the DirectX demonstration application to see examples of what can be done with DirectX:



To run the demonstration application, choose Partition Initialization when you log on to LANSA on Windows:

Visual LANS	SA Logon	×
User ID: Pass <u>w</u> ord:	pcxuser ******	
Use Windo	ows credentials	
Partition	Current Language English Français	Task ID PCXTASK
ОК	ystem Init	. Messages Cancel Help

In the Partition Initialization dialog choose Run Demonstration:



When you click OK, the LANSA DirectX Examples application is run:



Visual	Ì.	ΔΝς	Δ		
Dynamic Styles Adding, changing and removing styles at runtime	1	Veronica Brown (A0070) Information Services (INF) Development (DV)	1	Fred Bloggs (A0090) Fleet Administration (FLT) Accounting (03)	
Brushes Gradient colors linear or radial, images and other controls	1	Fred Smithson (A0193) Administration (ADM) Maintenance (05)	1	Anne Simpson (A0907) Internal Auditing (AUD) Accounting (03)	
Carousel User Designed Control with panels organized in an eliptical pattern	1	Shirley Jones (A1001) Administration (ADM) Internal Admin (01)	1	John Smythe (A1002) Administration (ADM) Purchasing (02)	
Tile User Designed Control with panels in a horizontal or vertical grid	1	Robert Smithe (A1003) Fleet Administration (FLT) Purchasing (02)	1	Ruth Smithson (A1004) Administration (ADM) Accounting (03)	
Tree and Drag & Drop User Designed Control with panels in a tree structure	1	Peter Smiths (A1005) Administration (ADM) Purchasing (02)	1	Jack Smithers (A1006) Travel (TRVL) Accounting (03)	
Book User Designed Control with panels as pages in a book	1	George Snell (A1007) Internal Auditing (AUD) Administration (01)	1	Alison Sneddon (A1008) Internal Auditing (AUD) Administration (01)	
Table Layout Grid style layout allowing components to occupy the same space	1	Damian Snashall (A1009) Internal Auditing (AUD) Purchasing (02)	2	Kate Perry (A1010) Internal Auditing (AUD) Accounting (03)	
Popup Panels User designed hints, context menus and drag and drop	1	Christopher Perron (A1011) Internal Auditing (AUD) Administration (01)	1	Patrick Paul (A1012) Administration (ADM) Internal Admin (01)	



#### New Visual LANSA IDE

The LANSA IDE (Integrated Development Environment) supports the new DirectX based features with new Dynamic Styles, Animation and Layout helpers.

#### **Ribbon vs. Menu and Toolbar**

In the new IDE the ribbon merges menus, shortcuts and toolbars into a single construct.

Ribbon uses a little more vertical space than a menu and toolbar, but this extra space provides a very valuable area in which contextual information can be shown.

For example the style helper shows how a style appears when the focus component is a component that uses a style:



The ribbon also allows you to create and configure styles in much the same way as the Version 12 layout helper does. However, by having the helper in the ribbon, you can see this information with a single click by bringing the sheet to the front.

The layout helper is now also in the ribbon showing basic layout information about the focus control without the need to show the layout helper dialog:



The ribbon changes as the context changes. For example, when a File is open, there are no execute or debug options as a File is not executable.

Ribbons also simplify access to all commands. The most common activities,

save, compile, check in, close can be found in the quick access toolbar which is always available. The remainder require little more than two or three keystrokes or a couple of clicks and are easy to discover.

The properties and features available in the ribbon are limited to the most commonly used, and specific views are available for editing the other properties.

# **Dynamic Styles**

Unlike Visual Styles which simply overwrite any existing style information when applied to a control, Dynamic Styles can be added and removed at runtime. This makes them akin to Cascading Style Sheets:



Dynamic Styles can be created as part of a Visual Style and accessed as read only features:

Define\_Com Class(#prim\_vs.Style) Name(#LargeFonts) Bold(True) Fontsize(72)

#Panel.Style <= #MyStyles<LargeFonts>

Alternatively they can be created at runtime in the same as any other component instance and then applied to controls as required. Styles created this way are entirely dynamic and can be modified at runtime to reflect user preferences:

Define\_Com Class(#prim\_vs.Style) Name(#Fontsize) Fontsize(10)

```
Evtroutine Handling(#Fontsize.Changed)
#DynamicStyle.FontSize := #FontSize
Endroutine
```

```
Multiple styles can be added or removed from a control:
Evtroutine Handling(#...)
#Control.Styles.Add(#MyStyles<LargeFonts>)
#Control.Styles.Add(#MyStyles<UnderlineItalic>)
Endroutine
```

You can use dynamic styles to define rounded corners, borders top, left, right or bottom, effects e.g. drop shadows, foreground and background brushes and opacity masks.

Styles can be applied at Application level to alter the appearance of all controls:

```
Define_Com Class(#prim_vs.Style) Name(#Label) Cornerbottomleft(3)
Cornerbottomright(3) Cornertopleft(3) Cornertopright(3)
Evtroutine Handling(#Com_owner.Createinstance)
#Sys_Appln.Appearance.Label <= #Label
Endroutine
```

### **MouseOverStyles**

All controls have two sets of styles: Styles and MouseOverStyles.

Styles are applied to the control and any child controls.

MouseOver Styles are applied to the control on MouseOver, i.e. when the mouse enters the bounds of the control, and removed on MouseLeave. This allows for simple declarative code rather than having to code lots of MouseEnter events.

Composite controls such as Panel and Group box have a third set of styles, PrivateStyles. Private Styles allow composites to continue to use the styles supplied by their parent, and to have their own styles that aren't passed on to child controls. For example they can make a group box caption red without making the contents red.

#### Brushes

Brushes allow for gradient colors, images and even other controls to be used to fill the background, foreground, or borders of a style.



There are several types of brushes available:

- Linear Brush
- Radial Brush
- Solid Brush
- Image Brush
- Visual Brush

### Linear Brush

The image below shows a Linear Brush transitioning from gray to white:



A linear brush is used to define a color transition that follows a line defined by the start and end coordinates:

Define\_Com Class(#prim\_vs.Style) Name(#Style) Backgroundbrush(#Brush) Define\_Com Class(#Prim\_Vs.LinearBrush) Name(#Brush) Colors(#Colors) Define\_Com Class(#Prim\_Vs.BrushColors) Name(#Colors) Define\_Com Class(#Prim\_Vs.BrushColor) Name(#Color1) Color(Gray) Parent(#Colors) Define\_Com Class(#Prim\_Vs.BrushColor) Name(#Color2) At(100) Color(White) Parent(#Colors)

# **Radial Brush**

The image below shows a radial brush transitioning from gray to white with an origin of 100, 100 (bottom right):



Radial brush is used to define a color transition that follows a line defined by the start and end coordinates emanating from an origin:

Define\_Com Class(#prim\_vs.Style) Name(#Style) Backgroundbrush(#Brush) Define\_Com Class(#Prim\_Vs.RadialBrush) Name(#Brush) Colors(#Colors) Originleft(100) Origintop(100) Define\_Com Class(#Prim\_Vs.BrushColors) Name(#Colors) Define\_Com Class(#Prim\_Vs.BrushColor) Name(#Color1) Color(Gray) Parent(#Colors) Define\_Com Class(#Prim\_Vs.BrushColor) Name(#Color2) At(100) Color(White) Parent(#Colors) End\_Com

# Solid Brush

The image below shows a solid white brush with opacity of 75.

The brush has been applied to a panel on top of which is the foreground text. Opacity of 75 allows the background to be visible through the panel:



Solid brush is used to define a brush with a single color:

Define\_Com Class(#prim\_vs.Style) Name(#Style) Backgroundbrush(#Brush) Define\_Com Class(#Prim\_Vs.SolidBrush) Name(#Brush) Color(White) Opacity(75)

### Image Brush

Employee Number Employee Surname Employee Given Name(s) Street No and Name Suburb or Town State and Country ABCCEFGHJIKLMINOPQRST aAbBcCdDeEfFgGhHiIjJKKILm aAbBcCdDeEfFgGhHiIjJKKILm

The image below shows an image brush at 30% opacity with overlaid fields:

Image brush is used to render an image as the content of the brush. The image can be sized, repeated and reflected. A typical use for an image brush is as a watermark or background image for a form:

Define\_Com Class(#prim\_vs.Style) Name(#Style) Backgroundbrush(#Brush) Define\_Com Class(#Prim\_Vs.imageBrush) Name(#Brush) Image(#Globe) Opacity(30) Sizing(BestFit)

# **Visual Brush**

A visual brush allows an image of another control to be used as the background or foreground of a style. This is particularly useful for drag and drop images. As with image brushes the image of the control can be resized or repeated.

a 🌇 Purchasing (02)				
Damian	101 Sackvill	e Road,		
Snashall	Ingleburn			
	NSW 2150			
	Home Phor	ne - 605 8686		
Accounting (03)	Damian Snahal	101 Sackville Road, Inglebum NISV/2150	0	
Fleet Administration (FLT)		Homs Phare - 605 3535		
Administration (01)				

## **User-Designed Controls**

User-designed controls are a new form of visual list similar to tree view or grid. However, user-designed controls have no predefined appearance, and the user has free reign over the design of the items that will be constructed. Items are added using the typical list commands, Add\_Entry, Dlt\_Entry, SelectList etc.

Each user-designed control has its own design interface (e.g.

#Prim\_Tile.iTiledesign, #Prim\_Tree.iTreedesign), which is implemented by a reusable part which allows the design instance to respond to focus, selection, expand/collapse in the case of tree, and other events.

You can also define the repository fields you want to use and these are mapped in to the design instance on add.

A user-designed control can use either a single class of design or many depending on requirement.

Define\_Com Class(#prim\_Tile<#MyTileDesign>) Name(#Tile)...

# Tile (Prim\_Tile)

Tile items are laid out in a grid pattern governed by the rules of a flow layout:

7	Veronica Brown (A0070) Information services (INF) Development (DV)	Fred joh Fleet adr Account	in alan Bloggs (A0090) ministration (FLT) ting (03)	3	Fred Smithson (A0193) Administrator dept (ADM) Maintenance (05)	8	Anne Miss simpson (A0907) Internal auditing (AUD) Accounting (03)	
7	<b>Ben Jones (A1001)</b> Administrator dept (ADM) Internal admin srv (01)	John Sm Adminis Purchasi	<b>nythe (A1002)</b> trator dept (ADM) ing section (02)	-	Robert Smithe (A1003) Fleet administration (FLT) Purchasing (02)	7	Paul Smithson (A1004) Administrator dept (ADM) Accounting section (03)	
7	Peter Smiths (A1005) Administrator dept (ADM) Purchasing section (02)	Jack Sm Travel de Account	i <b>thers (A1006)</b> epartment (TRVL) ting (03)	2	George Snell (A1007) Internal auditing (AUD) Administration (01)	2	Allan Sneddon (A1008) Internal auditing (AUD) Administration (01)	
å	Damian Snashall (A1009) Internal auditing (AUD) Purchasing (02)	William Internal Account	Perry (A1010) auditing (AUD) ting (03)	-	Christopher Perrin (A1011) Internal auditing (AUD) Administration (01)	7	<b>Patrick Paul (A1012)</b> Administrator dept (ADM) Internal admin srv (01)	
7	George Pattison (A1013) Administrator dept (ADM) Internal admin srv (01)	John Mo Adminis Purchasi	oore (A1014) trator dept (ADM) ing section (02)	-	<b>Bradley Woods (A1015)</b> Administrator dept (ADM) Internal admin srv (01)	2	Jack Turner (A1016) Fleet administration (FLT) Administration (01)	
7	Gary Neave (A1017) Information services (INF) Purchasing (02)	Paul Zac Group a Purchasi	: <b>haria (A1018)</b> ccounts (GAC) ing (02)	-	Charles Dickens (A1019) Legal department (LEG) Contracts, local (01)	2	<b>Adam peter Douglas (A1020)</b> Administrator dept (ADM) Internal admin srv (01)	
7	David Mccully (A1021) Administrator dept (ADM) Internal admin srv (01)	Kelly Th Marketir Purchasi	ompson (A1022) ng department (MKT) ing (02)	-	David Reid (A1023) Legal department (LEG) Accounting (03)	-	John Taylor (A1024) Marketing department (MKT) Administration (01)	
7	<b>Mary Robinson (A1025)</b> Administrator dept (ADM) Internal admin srv (01)	Group a Account	wis (A1026) ccounts (GAC) ting (03)	2	Alan Morrison (A1027) Administrator dept (ADM) Internal admin srv (01)	-	Andrew Maxwell (A1028) Group accounts (GAC) Financial control (FC)	
-	Valerie Turner (A1030) Information services (INF) Administration (01)	John Bla Manage Executiv	<b>ike (A1031)</b> mnt informatio (MIS) e informatio (EI)	-	Paul Lincoln (A1032) Information services (INF) Accounting (03)	-	<b>Warren peter Verey (A1111)</b> Administrator dept (ADM) Internal admin srv (01)	

# Tree (Prim\_Tree)

Tree items are added to the tree and behave like a simple list. However, by setting the ParentItem the item can be nested within another tree item. When an item is collapsed or expanded, the tree shows the necessary items.



#### Carousel

Carousel shows individual designs, in this case a simple image and label in either a linear or elliptical pattern. For simplicity, the example above uses the same image repeatedly.



#### Book

Book shows individual designs, in this case a simple image and label as if they were pages of a book. For simplicity, the example above uses the same image repeatedly.



# **Table Layout**

Table layout is new layout manager similar in concept to the Version 12 Grid layout or a table in Microsoft Word.



A table is split in to a series of columns and rows, each of which occupies a percentage or fixed pixel width portion of the space available. Controls to be laid out are given a layout item that specifies a row and column, and further allows a row span and column span to be specified.

Unlike most layout managers, Table allows controls to occupy the same space, greatly simplifying the creation of complex UI layouts.

The code below defines a table of 3 columns and 4 rows, the fourth being of fixed height. The table item manages #Control and will resize it to fill the first row.

Define\_Com Class(#prim\_tblo) Name(#TableLayout) Define\_Com Class(#Prim\_tblo.Column) Name(#TColumn1) Parent(#TableLayout) Define\_Com Class(#Prim\_tblo.Column) Name(#TColumn2) Parent(#TableLayout) Define\_Com Class(#Prim\_tblo.Column) Name(#TColumn3) Parent(#TableLayout) Define\_Com Class(#Prim\_tblo.Row) Name(#TRow1) Parent(#TableLayout) Define\_Com Class(#Prim\_tblo.Row) Name(#TRow2) Parent(#TableLayout) Define\_Com Class(#Prim\_tblo.Row) Name(#TRow3) Parent(#TableLayout) Define\_Com Class(#Prim\_tblo.Row) Name(#Trow4) Height(70) Parent(#TableLayout) Units(Pixels)

Define\_Com Class(#Prim\_tblo.Item) Name(#TableItem1) Column(#TColumn1) Columnspan(3) Manage(#Control) Parent(#TableLayout) Row(#TRow1)

See Table Layout Manager in the Visual LANSA Developer Guide.

# Popup Panel (Prim\_PPNL)

Popup panels are effectively an extension to the concept of user-designed controls. Rather than hints, popup menus and drag image being of a predefined type, popup panels can be used to show fully programmable reusable parts instead.

All controls now have Popup and HintPopup properties. By attaching a Popup to a control, on a right click, the popup will be shown. Similarly, by attaching a hint popup, when the hint is required the popup will be shown.

Similar to Popup menus, a Prepare event is fired shortly before the hint is to be shown allowing for the user to configure the popup content as required.



In the image above, a popup panel is used to show additional information about the active item. Similarly below, on a right click, not only is there a typical context menu, but also a contextual tool bar.



Unlike hints and popup menus, Popups are fully function reusable parts and can therefore take focus and react to keyboard input. This allows for similar context processing as seen in Microsoft Office.

The code below shows a Tile with hint and context popups.

Define\_Com Class(#prim\_Tile<#MyTileDesign>) Name(#Tile)

Hintpopup(#HintPopup) Parent(#COM\_OWNER) Popup(#ContextPopup)

Define\_Com Class(#prim\_ppnl) Name(#HintPopup)

Content(#HintPopupContent)

Define\_Com Class(#PopupPanelEmployeeDetails) Name(#HintPopupContent)

Define\_Com Class(#prim\_ppnl) Name(#ContextPopup)

Content(#ContextPopupContent)

Define\_Com Class(#PopupPanelContextPopup) Name(#ContextPopupContent)

# Scaling

Scaling allows a control to appear smaller or larger without changing the height and width properties. This is typically useful when viewing images, but can also be used to enhance the visibility of all application features. Scaled controls are still bound by the size of their parent control.



All controls now have ScaleHeight, ScaleWidth, ScaleOriginTop and ScaleOriginLeft properties. In conjunction with the static properties, controls also have an in built Scale animation method.

The code below makes a button scale to 50% larger and then back to normal size, and gives the impression of the button jumping out of the screen briefly

Define\_Com Class(#prim\_phbn) Name(#Button) Caption('Click Here') Displayposition(1) Parent(#COM\_OWNER) Tabposition(1) Evtroutine Handling(#Button.mouseEnter) #Button.Scale( 150 150 150 ) #Button.Scale( 100 100 150 150 ) Endroutine

### **Taskbar Integration**

Visual LANSA desktop applications can now interact with the Windows Taskbar



Windows TaskbarInfo allows a percentage progress to be specified as well as overlay images. ProgressStyle allows for the colour and behaviour of the progress to be modified as well.

#Application.TaskBarInfo.ProgressStyle := Paused #Application.TaskBarInfo.OverlayImage <= #Pause16

The code above set the taskbar to a paused state, resulting in a yellow color and overlays a pause image:



# Animations

Animations are nothing new. Even the green screen had blinking text.

In the modern world however, animations go a little further and allow developers to add a little glitz and glamor, as well as providing a means of highlighting features to the end user whenever there is a need to draw attention to a particular area of the screen.

Perhaps the most important single feature of animations is that they operate entirely in their own thread meaning that animations can execute while other longer processing is going on.

There are two types of animation:

Transitions

**Control Animations** 

#### Transitions

Transitions are little more than a more interesting way of bringing a panel to the front, similar to the way slide transition in Microsoft PowerPoint.

Today, when we want to bring a panel to the front we write code that looks similar to the below.

#Panel2.Visible := True
#Panel1.Visible := False

This shows a secondary panel and hides the currently visible one. The processing is instant and the effect is simple.

However, with transitions we can make one panel fade while the other appears, or have one roll down from the top of the screen to cover one on top.

Define\_Com Class(#prim\_anim) Name(#Animation) Define\_Com Class(#prim\_anim.Transition) Name(#Flip) Source(#Panel1) Target(#Panel2) Transitiontype(Flip) Parent(#Animation) Evtroutine Handling(#Button.Click) #Animation.Start Endroutine

The code above defines a simple animation that uses the Flip transition. This will be available as a feature of all controls eventually resulting in much more concise code.

Evtroutine Handling(#Button.Click) #Panel1.TransitionTo(#Panel2 Flip) Endroutine
# **Control Animations**

Control animations allow for much more varied animations. There are move, fade, rotate, scale and many more types of animation, each of which can be used to affect the state of a specific control.

Importantly however, animations cannot put a control in state that cannot be described by the properties of the control. So, while a control's properties are not being updated during the animation, when complete the animation the control is still subject to the normal Visual LANSA rules.

For a simple example, a button that is positioned by a layout can still be moved by an animation to or from a given point, but when the animation finishes the layout will again take control and position the button.

For simplicity, some animations are available as features of controls.

Evtroutine Handling(#Button.Click) #Button.Scale(200) #Button.FadeOut Endroutine

The code above will start scaling the button to twice its width and then start fading out.

More complex animations can be achieved by building composite animations that move multiple controls

```
Define_Com Class(#prim_anim) Name(#Animation)
Define_Com Class(#Prim_anim.Opacity) Name(#ShowBanner) Duration(2000)
Manage(#Banner) Opacity(100) Parent(#Animation)
Define_Com Class(#Prim_anim.opacity) Na`me(#ShowGlobe) Duration(2000)
Manage(#Globe) Opacity(100) Parent(#Animation)
Define_Com Class(#Prim_anim.MoveFrom) Name(#BannerInFromLeft)
Duration(2000) Manage(#Banner) Parent(#Animation)
Define_Com Class(#Prim_anim.MoveFrom) Name(#GlobeInFromBelow)
Duration(2000) Manage(#Globe) Parent(#Animation)
```

This example is taken from the DirectX Sample splash screen. It takes 2 seconds to make a Banner and Globe 100% opaque while at the same moving them from off the screen in to view.

Complex animations may have many component pieces and all will be executed

when the animation starts. Each piece has a StartTime property that can be modified so that the animation can run in a specific sequence.

# **Adopting DirectX**

When running an application in Version 13, it will continue to behave exactly as it did in version 12. Only by actively choosing to use the DirectX rendering option will the application change. This can be done at application, form or even panel level. For many users, this transition may be almost seamless. For some however, DirectX rendering may subtly impact the behavior of the existing application.

LANSA has gone to great lengths to ensure that "flicking the DirectX switch" is as simple and uneventful as possible, and that the user interface remains close to that of version 12. However, with such an array of new functionality and the restrictions imposed by the adoption of new underlying technologies, some change is inevitable

Adopting DirectX Rendering DirectX Changes Samples Source

# **Adopting DirectX Rendering**

Whilst every effort has been made to ensure that as much of the Win32 appearance has been honoured as possible, circumstances dictate that it is simply impossible to provide a DirectX runtime that precisely reflects that of Win32.

Before enabling for DirectX, you need to answer one question. Do you really need to? If the answer is no, you should continue using version 13 as you did version 12.

Enabling for DirectX

Strategies

Test, Test, Test

# **Enabling for DirectX**

Visual LANSA allows DirectX to be applied at panel, form or application level. Once DirectX is enabled, all child panels will also use DirectX, specific Win32 controls notwithstanding.

If we set the runtime to DirectX, all forms, panels and controls within the application will use DirectX rendering.

If we set a form to use DirectX, child panels and controls will use DirectX rendering.

If we set a panel to use DirectX, panels and controls will use DirectX rendering.

## Strategies

In practical terms, there are two strategies for the adoption of DirectX – Wholesale or piecemeal.

Piecemeal is the less invasive strategy. This allows individual forms or panels to start using DirectX related features e.g. new controls or animations, without affecting the remainder of the application. It is worthy of note however that this will enforce a TrueType font to be used, and it may therefore be necessary to change the font for the remainder of the application to ensure consistency.

The wholesale approach simply means that the runtime is switched to use DirectX and that as a result the entire application will render using DirectX.

### Test, Test, Test

Regardless of the strategy employed to start using DirectX, the majority of applications may well behave slightly differently. For some it will be as subtle as a change in font; for others, parts of the application will not behave quite the same as before and may cause runtime errors.

It is strongly recommended that you perform a full test of your applications before enabling any productions systems.

### **DirectX Changes**

The following sections detail individual changes and explain the reasons for the change and how they may impact existing applications. Where possible, workarounds and simple ways in which these issues can be overcome are specified.

Default Appearance Transparency and Opacity Routed Events Mouse Events Visual Styles True Type Fonts Win32 & DirectX (ActiveX and Graphs) UpdateDisplay

## **Default Appearance**

Below are images of the same form running firstly as it would appear in version 13 using Win32 and secondly as DirectX.

Simple Form Running	in Win32	
Employee Number Employee Surname Employee Given Name(s)		
	<u>0</u> K	<u><u>C</u>ancel</u>
Simple Form Running	in DirectX	
Employee Number		

Employee Given Name(s			
	<u>0</u> K	Cancel	
	<u>0</u> K	<u>C</u> ancel	

The code for this form is available in the Sample Source section of this document.

Functionally, the two forms are identical. However, the font used for the text is different. Win32 defaults to MS Sans Serif 9 while Direct X defaults to Segoe UI 9.

Ms Sans Serif is an old font and was created a long time ago when screens were much smaller and had much lower resolutions. The result is that on a modern screen, running a modern resolution, it looks rather "blocky" compared to the smooth edges and nicely rounded corners of a modern True Type font.

For most users the change of font may well be of no consequence. However, Segoe UI is slightly wider, and as can be seen form the two images, and this may cause some text to wrap, show ellipses or be truncated.

### **Transparency and Opacity**

State and Country

Post / Zip Code

DirectX rendering introduces transparency and opacity. By default in DirectX all panels and labels are considered transparent unless a specific Style has been applied. This new appearance can lead to issues.

Below, a simple form toggles between address and employee details. When the button is clicked, the address details are enabled and brought to the front.

Employee Number Employee Surname Employee Given Name(s)		Show	Address
Street No and Name Suburb or Town		Show	Details

The code for this form is available in the Sample Source section of this document.

0

However, the results with DirectX rendering are somewhat different.

Strapelojskeedskolm/Navene		Show Address
Schuldyer Somame		
Statpel aynete Coiventrilylame(s)		
Post / Zip Code	0	

Etrepetojviezeahidrhilaen e		Show Details
Gotpldyer Somame		
Etateloynee@iwentrlylame(s)		
Post / Zip Code	0	

Regardless of the DisplayPosition of the Address and Details panels, both are plainly visible.

The need for this default stems from the desire to build complex layered forms and to still be able to see watermark images or backgrounds applied to it. If they were opaque, it would be necessary to visit every panel and label and specifically apply a transparent style.

A simple work around for this situation is to set the inactive panel to Visible(False) rather than Enabled(False).

# **Routed Events**

To simplify the coding of complex reusable parts and in particular the design panels for the new user- designed controls, an event detected on a control is now passed up the parent chain.

Typically for user-designed control, the panels displayed are constructed of little more than labels and images. However, with the existing event processing, each of the labels would take the click event and not pass it on. The result would be that the user would need to code every click event for every child control. By routing the event up the parent chain, coding is greatly simplified.

Of course, it may be necessary to know which of the controls actually fire the event initially. The EVTROUTINE command already has the Com\_Sender selector, but this only ever reports the control firing the event. So, the Origin selector has been added. Regardless of how many layers of parent are used, Origin will contain a reference to the instance on which the event was actually started.

Evtroutine Handling(#Com\_owner.Click) Origin(#Origin)

•••

Endroutine

The only exception to this rule is when the event crosses a reusable part boundary. Reusable parts are black boxes as such and the rules of encapsulation dictate that what goes on within a reusable part must stay within a reusable part. So, in this situation, as far as the parent is concerned, the origin is the reusable part itself.

As at the time of release of this document, the Origin control is still available across the reusable part boundary. This is a known issue.

Clearly though, this change of event behavior may have some side effects. In a simple example where there is a click event for both child and parent components, in Win32 the two events would remain separate. However, with DirectX processing and event routing a click on the child would result in both the child click and parent click firing. This is an unusual situation and it is unlikely that many customers will encounter it, but nevertheless it is conceivable and should be catered for.

To counter unwanted event propagation, the Handled Selector has been added to EVTROUTINE. By setting Handled to true the event is no longer passed beyond the routine being processed.

Evtroutine Handling(#Button.Click) Handled(#Handled) \* Stop the event going any further up the parent chain. #Handled := True Endroutine

## Mouse Events

When an application is using DirectX, there are changes to the way in which the list controls (Tree, Grid etc.) appear. Most notably, when the mouse is over an item in the list, the item will be highlighted. In the image below, the first item in

the list is the FocusItem. The 5<sup>th</sup> item has the mouse over it and is therefore the CurrentItem.

DirectX	op the passes for		х
Delete			
Employee Number	Employee Surname	Employee Given Name(s)	
A0070	BROWN	VERONICA	-
A0090	BLOGGS	FRED JOHN ALAN	
A0193	SMITHSON	FRED	
A0907	MISS SIMPSON	ANNE	
A1001	JONES	BEN	
A1002	SMYTHE	JOHN	
A1003	SMITHE	Robert	
A1004	SMITHSON	PAUL	
A1005	SMITHS	PETER	
A1006	SMITHERS	JACK	
A1007	SNELL	GEORGE	
A1008	SNEDDON	ALLAN	
A1009	SNASHALL	DAMIAN	-

The code for this form is available in the Sample Source section of this document.

All LANSA lists use a CurrentItem concept that maps the equivalent field values from the list and into the equivalent variables. CurrentItem effectively represents the item last processed in the list. This might be the last item clicked on, which will also be the FocusItem, or perhaps the last item processed in a Selectlist loop.

Historically, it was common to see processing similar to the above where the Delete button would cause the deletion of the currently selected item with code similar to the following.

```
Evtroutine Handling(#Delete.Click)
If (#List.CurrentItem *IsNot *null)
Dlt_Entry Number(#List.CurrentItem.Entry) From_List(#List)
Endif
Endroutine
```

This code effectively assumes the CurrentItem and FocusItem are going to be one and the same, and for most scenarios prior to DirectX that would be the case.

However, in DirectX, MouseOver processing has been added for all lists and takes effect immediately. Even if the related event routines aren't coded, the runtime is still determining what the CurrentItem is and this can affect the values of variables.

Relying on CurrentItem is not good practice outside of a Selectlist as it can be corrupted. For single selection lists you should always use the FocusItem.

# **Visual Styles**

Visual Styles have historically ignored the BorderColor property for reasons that are unclear. Borders were painted black unless a theme was applied in which case the theme took over and applied a border color of its own.

Under DirectX, the border color as specified will now be applied. This may cause issues for some customers who have erroneous values specified in their Visual Styles. The obvious workaround is to correct the Visual Style.

As at the time of release of this document, Visual Style BorderStyle is ignored in a DirectX runtime. All borders are shown as a single line.

# **True Type Fonts**

DirectX rendering only supports True Type fonts. This is simply a reflection of the underlying Microsoft technologies. True Type and Open Type, an extension of True Type, are industry standards and designed to render smoothly regardless of the font size used.

Where a font cannot be rendered, Visual LANSA uses Segoe UI.

Fonts such as MS Sans Serif, which is not True Type, typically have modern True Type alternatives. The MS Sans Serif equivalent is Microsoft Sans Serif.

If you intend to adopt DirectX, it is strongly recommended that you change your application to use a True Type font. This may cause issues with text no longer fitting in the available space and it is recommended that you review any changes you have made.

# Win32 & DirectX (ActiveX and Graphs)

There are compatibility issues when trying to work with both Win32 and DirectX in the same UI space. As with fonts, this is a reflection of the underlying technology.

For many customers, the adoption of DirectX may well occur in individual forms bolted on to existing applications, or perhaps as panels embedded in existing forms. This technique is fine as DirectX will work within Win32 quite happily. However, the reverse is not quite so simple, and Win32 controls don't necessarily behave in the way one might think.

Win32 controls cannot occupy part of the same render level as DirectX and are therefore placed on a different level. This results in a situation where child Win32 controls that are bigger than their parent will cause scrolling issues. In the image below, the browser is parented to a panel which has been scrolled. See how the top of the browser coincides with the top of the panel scroll bar on the right. As this is a Win32 application, the browser is correctly clipped.



The code for this form is available in the Sample Source section of this document.

However, in the same form running as DirectX (below), scrolling the panel causes the ActiveX to move, but not to clip. See how the browser is now above

the panel scrollbar on the right.



The only practical solution to this issue is to ensure that the Win32 control is sized appropriately, probably by use of a layout manager, and does not exceed the size of its parent.

As at the time of release of this document, Memo (Prim\_memo) and Listview (Prim\_ltvw) are still Win32 controls.

# UpdateDisplay

In Win32 the UpdateDisplay method could be called on a control to force the screen to refresh during a long running process. The Win32 runtime was able to address individual controls specifically and in effect could update a small portion of the UI.

However, the DirectX runtime works in a different manner and this is no longer possible. UpdateDisplay will cause the whole of the form to update.

In most circumstances this will be of little consequence. However, in situations where UpdateDisplay is called repeatedly, this will cause noticeable performance degradation.

A typical situation where that occurs is when a Progress Bar is used. Progress Bars automatically use UpdateDisplay to ensure that they reflect their latest value. In the example below a simple loop is executed and the progress bar and start button caption are updated every iteration.

Win32	1	-	-		X
				-	
	Start (0)				

In Win32 above, the start button is not updated. The UpdateDisplay is specific to the Progress Bar. However, in DirectX, the whole form gets updated.

DirectX			1		
	-				
	-				
		_			
Star	t (6019)				

The code for this form is available in the Sample Source section of this document.

To counteract this situation, rather than updating the progress bar or specifically executing UpdateDisplay every iteration, a simple test can be added so that the

update only occurs every 10<sup>th</sup> time.

# **Samples Source**

Default Appearance

Transparency and Opacity

Mouse Events

Win32 & DirectX (ActiveX and Graphs)

UpdateDisplay

# **Default Appearance**

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Clientheight(119)
Clientwidth(336) Componentversion(1) Height(157) Left(175) Top(215)
Width(352)
Define Com Class(#EMPNO.Visual) Name(#EMPNO) Componentiesion(1)
Displayposition(1) Left(8) Marginleft(130) Parent(#COM OWNER)
Tabposition(1) Top(8)
Define Com Class(#SURNAME.Visual) Name(#SURNAME)
Componentversion(1) Displayposition(2) Left(8) Marginleft(130)
Parent(#COM OWNER) Tabposition(2) Top(32)
Define Com Class(#GIVENAME.Visual) Name(#GIVENAME)
Componentiersion(1) Displayposition(3) Left(8) Marginleft(130)
Parent(#COM OWNER) Tabposition(3) Top(56)
Define_Com Class(#PRIM_PHBN) Name(#OK) Buttondefault(True)
Caption('&OK') Displayposition(4) Left(164) Parent(#COM OWNER)
Tabposition(4) Top(88)
Define Com Class(#PRIM PHBN) Name(#Cancel) Buttoncancel(True)
Caption('&Cancel') Displayposition(5) Left(252) Parent(#COM OWNER)
Tabposition(5) Top(88)
Evtroutine Handling(#Com owner.CreateInstance)
Case (#sys appln.RenderStyle)
When (= DirectX)
#Com owner.Caption := "DirectX"
When (= Win32)
#Com owner.Caption := "Win32"
Endcase
Endroutine
End Com
```

# **Transparency and Opacity**

Function Options(\*DIRECT) Begin Com Role(\*EXTENDS #PRIM FORM) Clientheight(114) Clientwidth(522) Height(152) Left(106) Top(204) Width(538) Define\_Com Class(#PRIM\_PANL) Name(#Details) Displayposition(1) Height(108) Left(15) Parent(#COM OWNER) Tabposition(2) Tabstop(False) Top(13) Width(338) Define Com Class(#PRIM PANL) Name(#Address) Displayposition(2) Enabled(False) Height(108) Left(15) Parent(#COM\_OWNER) Tabposition(1) Tabstop(False) Top(13) Width(338) Define Com Class(#PRIM PHBN) Name(#MoveToFront) Caption('Show Address') Displayposition(3) Left(360) Parent(#COM OWNER) Tabposition(3) Top(8) Width(153) Define Com Class(#EMPNO.Visual) Name(#EMPNO) Componentversion(1) Displayposition(1) Height(20) Left(8) Parent(#Details) Tabposition(1) Define Com Class(#SURNAME.Visual) Name(#SURNAME) Componentversion(1) Displayposition(2) Height(20) Left(8) Parent(#Details) Tabposition(2) Top(24) Define Com Class(#GIVENAME.Visual) Name(#GIVENAME) Componentversion(1) Displayposition(3) Height(20) Left(8) Parent(#Details) Tabposition(3) Top(48) Define\_Com Class(#ADDRESS1.Visual) Name(#ADDRESS1) Componentversion(1) Displayposition(1) Height(20) Left(8) Parent(#Address) Tabposition(1) Width(300) Define Com Class(#ADDRESS2.Visual) Name(#ADDRESS2) Componentversion(1) Displayposition(2) Height(20) Left(8) Parent(#Address) Tabposition(2) Top(24) Usepicklist(False) Width(300) Define Com Class(#ADDRESS3.Visual) Name(#ADDRESS3) Componentversion(1) Displayposition(3) Height(20) Left(8) Parent(#Address) Tabposition(3) Top(48) Width(300) Define\_Com Class(#POSTCODE.Visual) Name(#POSTCODE) Componentversion(1) Displayposition(4) Height(20) Left(8) Parent(#Address) Tabposition(4) Top(72) Usepicklist(False) Width(249) Evtroutine Handling(#MoveToFront.Click) If (#Details.DisplayPosition <> 1) #Details.DisplayPosition := 1 #Details.enabled := True #Address.enabled := False

#MoveToFront.Caption := "Show Address"
Else
#Address.DisplayPosition := 1
#Details.enabled := False
#Address.enabled := True
#MoveToFront.Caption := "Show Details"
Endif
Endroutine
End\_Com

#### **Mouse Events**

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Clientheight(306)
Clientwidth(462) Componentversion(1) Height(344) Left(103) Top(200)
Width(478)
Define Com Class(#PRIM Trvw) Name(#List) Columnbuttonheight(19)
Componentities (False) Componenties (Componenties (Compone
Height(261) Keyboardpositioning(SortColumn) Left(8) Linesatroot(False)
Parent(#COM OWNER) Tabposition(1) Top(40) Viewstyle(UnLevelled)
Width(444)
Define Com Class(#PRIM TVCL) Name(#TVCL 1) Displayposition(1)
Level(1) Parent(#List) Source(#EMPNO) Width(27)
Define Com Class(#PRIM TVCL) Name(#TVCL 2) Displayposition(2)
Level(2) Parent(#List) Source(#SURNAME) Width(33)
Define Com Class(#PRIM TVCL) Name(#TVCL 3) Displayposition(3)
Level(3) Parent(#List) Source(#GIVENAME) Width(40)
Define Com Class(#PRIM SPBN) Name(#Delete) Caption('Delete')
Displayposition(2) Left(8) Parent(#COM OWNER) Tabposition(2) Top(8)
Width(137)
Evtroutine Handling(#Com owner.CreateInstance)
Case (#sys_appln.RenderStyle)
When (= DirectX)
#Com_owner.Caption := "DirectX"
When (= Win32)
#Com owner.Caption := "Win32"
Endcase
Select Fields(#List) From File(pslmst)
Add Entry To List(#List)
Endselect
Endroutine
Evtroutine Handling(#Delete.Click)
If (#List.CurrentItem *IsNot *null)
Dlt Entry Number(#List.CurrentItem.Entry) From List(#List)
Endif
Endroutine
End Com
```

# Win32 & DirectX (ActiveX and Graphs)

Function Options(\*DIRECT) Begin Com Role(\*EXTENDS #PRIM FORM) Clientheight(574) Clientwidth(965) Componentversion(1) Height(612) Left(112) Top(184) Width(981) Define Com Class(#PRIM PANL) Name(#Panel) Displayposition(1) Height(481) Horizontalscroll(True) Left(80) Parent(#COM OWNER) Style(#Style\_1) Tabposition(1) Tabstop(False) Top(56) Verticalscroll(True) Width(793) Define Com Class(#VA WEBCTL.WebBrowser) Name(#Browser) Displayposition(1) Height(600) Left(0) Parent(#Panel) Tabposition(1) Top(0) Width(775) Define Com Class(#PRIM VS.Style) Name(#Style 1) Backgroundbrush(#RadialBrush 1) Define Com Class(#PRIM VS.BrushColors) Name(#BrushColors 1) Define\_Com Class(#PRIM\_VS.BrushColor) Name(#BrushColor\_1) At(25) Color(255:255:255) Parent(#BrushColors 1) Define Com Class(#PRIM VS.BrushColor) Name(#BrushColor 2) At(100) Color(32:155:204) Parent(#BrushColors 1) Define\_Com Class(#PRIM\_VS.RadialBrush) Name(#RadialBrush 1) Colors(#BrushColors 1) Originleft(100) Origintop(100) Radiusleft(125) Radiustop(125) Evtroutine Handling(#Com\_owner.CreateInstance) Case (#sys appln.RenderStyle) When (= DirectX) #Com owner.Caption := "DirectX" When (= Win32)#Com owner.Caption := "Win32" Endcase Endroutine Evtroutine Handling(#Com\_owner.initialize) #Browser.Navigate( www.lansa.com ) Endroutine End Com

# UpdateDisplay

```
Function Options(*DIRECT)
Begin Com Role(*EXTENDS #PRIM FORM) Clientheight(124)
Clientwidth(498) Componentversion(1) Height(162) Left(261) Top(195)
Width(514)
Define Com Class(#PRIM PGBR) Name(#ProgressBar) Displayposition(1)
Left(8) Maximumvalue(10000) Minimumvalue(0) Parent(#COM OWNER)
Tabposition(1) Top(8) Value(1) Width(481)
Define Com Class(#PRIM PHBN) Name(#Start) Caption('Start (0)')
Displayposition(2) Height(41) Left(8) Parent(#COM_OWNER) Tabposition(2)
Top(72) Width(177)
Evtroutine Handling(#Com owner.CreateInstance)
Case (#sys appln.RenderStyle)
When (= DirectX)
#Com owner.Caption := "DirectX"
When (= Win32)
#Com owner.Caption := "Win32"
Endcase
Endroutine
Evtroutine Handling(#Start.Click)
#ProgressBar.value := 0
Begin_Loop To(10000)
#ProgressBar.value += 1
#Start.Caption := ("Start (&1)").Substitute( #ProgressBar.Value.Asstring )
End Loop
Endroutine
End Com
```

### **Ability to Consume .NET Components**

Your LANSA programs can consume .NET UI and non-UI components:

- Graphs
- Buttons
- Barcode scanners
- Device specific interfaces

You register 3<sup>rd</sup> party .NET controls into the repository and then use the exposed events, methods and properties in RDLMX.

For more information, refer to .NET Components in the Visual LANSA Developer Guide.



### WAMs for Mobiles

The jQuery Mobile TSP provides an enhanced HTML 5 solution for mobile devices based on the jQuery Mobile touch optimized web framework.



jQuery Mobile is a unified, HTML5-based user interface system for all popular mobile device platforms, built on a jQuery and jQuery UI foundation. Its lightweight code is built with progressive enhancement, and it has a flexible, easily themeable design.

The jQMobile Technology service is a wrapper around the JavaScript library that allows it to work in the LANSA IDE with drag-and-droppable weblets that are configured by editing properties.

You can create your mobile WAMs using a Wizard which creates sampler applications and design outlines for a mobile app:

Start	Choose v	our page layout and	drag and drop th	e controls bel	ow.
✓ Sampler ✓ Web Page 2	Webroutine Name	Web_Page_2			
<ul> <li>✓ Heading 1</li> <li>✓ Text Block 2</li> </ul>	Webpage Title	Web Page 2			
<ul> <li>✓ List View 3</li> <li>✓ <list data=""></list></li> <li>✓ Image 4</li> </ul>	H1 H2 нз Heading Te	ext	<form> Form Elements</form>	Text Link	List View
	<ul> <li>Heading 1</li> <li>Text Block 2</li> <li>List View 3</li> </ul>				
	Image 4				

Also see LANSA Web Mobile Application Wizard.

### Internationalization with UNICODE

LANSA now provides full support for Unicode:

- Database tables support UNICODE data type in columns.
- The IDE now displays all multilingual text using Unicode, so you see all text in all languages displayed as you will see it when it executes.
- Full support of Unicode in RDMLX components and functions.
- User interfaces forms and WAMs support Unicode.
- WAMs output UTF-8 regardless of language in use.
- Support of UTF-8 (CCSID 1208) on IBM i Other Files
- Unicode string intrinsics ensure data integrity (AsNativeString and AsUnicodeString).
- Unicode field types Nchar and Nvarchar:

🕒 New Field		×
Name	Name	Create
Description	Name	Cancel
Field Type	NChar	
Field Length	10 ‡	Open in editor
Decimals	÷	Close
Reference Field		
Identifier	NAME	
Enabled For RDMLX	<b>V</b>	

The introduction of Unicode has resulted in some changes to the Built-In Function Rules. To see the changes refer to Built-in Function Rules in the Technical Reference.

## Version Control System Support

Visual LANSA now supports using a Version Control System as a Master Repository instead of an IBM i as the Master. This system configuration is called VCS Master.



Each Developer is "sandboxed" so that they do not interfere with other developers' work until it is checked in to the VCS. Any VCS can be used but it is essential that in-depth knowledge of the VCS is acquired.

VCS Master is designed to be used by Windows developers who have prior knowledge of using a VCS. As such it can be likened to using a VCS with any other Windows development environment, like Visual Studio.

All Version Control features such as branching, merging, comparing source, labeling, etc. provide better control of your source code – but they require a good deal of planning and discipline to make them work.

The VCS controls access to the source code, so LANSA task tracking and LANSA security are ignored and source code audit stamps are not available. Refer to LANSA guides for more details.

# Deployment

LANSA applications are now packaged as standard Microsoft Installer (MSI) packages for deployment:



MSI integrates with SCCM (install management software), has better support for multiple language installations and is more configurable which has allowed version updating to be more targeted.

For more information, refer to What's New in the Deployment Tool.

#### Long Names

Long names have been introduced to enable 3<sup>rd</sup> party integration and descriptive names.

Several different LANSA object types can be referred to in RDMLX partitions by a name that is longer than the traditional ten characters. These object types are:

- Fields
- Files
- Logical views
- VL Components
- Processes
- Functions
- WAMs

Long names can only be given to these LANSA objects in Visual LANSA. The use of long names is an RDMLX partition-level setting:


### **RDMLX Enhancements**

A number of enhancements have been made to RDMLX:

- New intrinsics and primitives
- New language commands (GET/SET)
- 32K lines of code limited by compiler
- Recursive Stack raised from 50 to 250
- Constructors for an instance
- \*NEW operator for inline construction
- Assign Category for your own methods.

### **Windows Centric Development**

• Housekeeping and Administration tasks such as impact analysis are available with equivalent functionality in Visual LANSA:

🖹 New List		X
Name	Impact List 2	<u>C</u> reate
Туре	Impact	 Ca <u>n</u> cel
Stored As	Repository List	 Is Favorite
Identifier	OLO	 Distoric

See Impact Analysis.

- The use of Identity columns in tables removes the requirement for RPTH files.
- Post-mortem Debugging has been enhanced: LANSA executables now automatically produce a dump file if an unhandled exception such as an access violation occurs. This shortens the turnaround time for resolving these rare issues and makes it much easier to capture the program state with hard to reproduce issues.
- Version information, such as product name, product and file version and copyright, can be included in LANSA objects when they are compiled. This information is visible in the DLL's properties:

eneral Security	Previous Versions	
Property	Value	
Description		
File description	DF_PRM02	
Туре	Application extension	
File version	3.0.0.0	
Product name	ApplicationA	
Product version	1.2.0.0	
Copyright	LANSA	
Size	39.5 KB	
Date modified	27/10/2012 6:41 PM	
Language	English (United States)	
Legal trademarks	TM	
Original filename	DF_PRM02.DLL	
Remove Properties	and Personal Information	

## **Impact Analysis**

Impact lists allow you to find objects that relate to a specific object so that you can assess the impact of a planned change.

G	Qualified	Object	Description	Processing	Job Start Time	Job End time	
	Comp	leted	Impact List Search MyList	Search Ended - Found 87 Results	8/05/2012 12:30:42 PM	8/05/2012 12:30:53 PM	
×	6	DF_ELDATX	Standard RDMLX date				
		FP_EIDN	Standard Identifier				
×	6	FP_ENUM	Standard Number				
	6	KATETEXT	Standard TEXT				
0	6	STD_ALPHA	Standard alpha field				
	6	STD_AMNT	Standard AMOUNT				
X	6	STD_BIN	Standard Binary field				
	l C	STD_BOOL	Standard Boolean				
	l C	STD_CENT	Standard CENTURY				
	l C	STD_CHKBX	Standard check box				
	l C	STD_CMPAR	Standard comparison GT/E				
	l (	STD_CODE	Standard CODE				
		STD CODEL	Standard CODE LONG				

Using the List Type *Impact*, you can select the objects you wish to be included in the list, and then, for each object type, you can specify a specific or common filter to narrow your selection.

For example, a common filter could be to search for a full or partial name. A specific filter could include, for each field in the object, the operators *Like* with or without the asterisk (\*) wildcard, *Equal to*, *Greater than*, *Less than*, *Less than or equal to* as shown here:

Filters			Add 🖌 📐	
Type Field Length	Equal to Equal to	×	Ada A     Definition	
Decimals Default Value	Equal to Equal to	×	Identifier Name	OL37 MyList
Reference Field Field Label	Equal to Equal to	*	Content  Content  Search For  Object Types  Diade	
Heading 1 Heading 2	Equal to Equal to	*	□ 😽 Files □ 😭 Files	
Heading 3 Edit code	Equal to Equal to	•		
Edit Word Keyboard Shift	Equal to Equal to	* *	•	
Input Attributes Output Attributes	Equal to Equal to	×	*	

Your list can be saved as a Static list or as an Excel file enabling you to re-use it again at a later time.

## **Run an Impact List**

To run the impact list, click on the "Run" button in the impact list toolbar. Clicking on the "Run" button will also save the impact list.

## **Review the results**

The results will be displayed in the Impact Analysis output view.

G	Qualified Object	Description	Processing	Job Start Time	Job End time	
	Completed	Impact List Search MyList	Search Ended - Found 87 Results	8/05/2012 12:30:42 PM	8/05/2012 12:30:53 PM	
×	DF_ELDATX	Standard RDMLX date				
	FP_EIDN	Standard Identifier				
×	FP_ENUM	Standard Number				
	KATETEXT	Standard TEXT				
8	STD_ALPHA	Standard alpha field				
	STD_AMNT	Standard AMOUNT				
X	STD_BIN	Standard Binary field				
_	STD_BOOL	Standard Boolean				
2	STD_CENT	Standard CENTURY				
	STD_CHKBX	Standard check box				
	STD_CMPAR	Standard comparison GT/E				
	STD_CODE	Standard CODE				
	STD CODEL	Standard CODE LONG				

# Export

In the output view, click on the job heading. You will then be able to export the list of objects produced by the impact analysis.

a. Export to Excel

With the job heading selected, click on the export to *Excel* icon. The list of LANSA objects will be exported to a CSV file.

b. Export to a Static List

With the job heading selected, click on the *Export as Static List* option. A dialog will open where you can choose the name and the type of list.

## **Change of Collection/Library on Import and Deployment**

File library or collection can be changed on import and deployment to override default behavior.

C:\What's New\DirectX\	
Source Partition - Unknown	
Contents to import	
<ul> <li>User Objects</li> <li>Fields (7)</li> <li>Forms (3)</li> <li>Bitmaps (37)</li> <li>Multilingual Variables (82)</li> <li>Visual Styles (1)</li> <li>Reusable Parts (29)</li> </ul>	
4	
Save As Repository List	
☑ Override File Library To Partition Data Library	port Cancel
Remove Duplicate Long Names	

By default OAMs are created in a sub-directory of the partition execute directory, named after the partition default file library. For example, if the partition default file library is DC@DEMOLIB in partition DEM, the OAM is created in the directory X\_DEM\DC@DEMOLIB\EXECUTE. If the file's library is not the same as the Partition Default library, the Visual LANSA IDE will put the OAM in the partition execute directory.

The default behaviour when deploying a File is to change the SQL Table and OAM to use the partition default file library of the target partition.

### **File Enhancements**

IBM i Other Files now support:

- Binary and Varbinary field types
- UTF-8 (CCSID 1208)

Boolean is supported as a repository field type.

### **Installation and Development**

New installation and development features have been introduced to ensure ease of use and developer productivity.

## **Supported Platforms**

See Supported Versions at the LANSA Support website.

## **Upgrade** Path

Any Version 12 system can be directly upgraded to Version 13.

Version 11(SP5) systems must first be upgraded to Version 12.

## Note about Check-In and Check-Out

When you upgrade to Version 13, the repository state of all objects will be set to *Not Checked Out*. This means that before upgrading all changes to objects in all partitions must be checked into the IBM i.

After the upgrade, when an object is checked out to a PC, it is locked out and no other PC can modify it. If you do not want to lock the object, check it out as read-only. Both the Visual LANSA Editor and the IBM i have an unlock option which you can use to allow another developer to access an object.

6	Open
٠	Compile
	Execute
	Debug >
×	Delete from Repository
	Find
	Quick Export
-	Check In
4	Check Out
	Check Out Readonly
£	Unlock
-	Сору
	Copy Name
P	Properties
÷.	Cross References
2	Security Settings
2	<u>F</u> eatures

### Visual LANSA Framework

#### Version 13 Features

This version of the Framework utilizes and showcases the new features in LANSA Version 13, including the DirectX user interface.

#### **Customized Quick Finds**

The Quick Find box is a dialog that appears on the top right of the VLF window.

The current behaviour is to search a list of all business object captions. This can now be overridden so that the user searches a list of values that you control.

And when the user selects one of your values, you control what happens. Typically this would be a switch to a business object, or to an instance list entry in a business object, or a command handler for a business object.

If necessary you can also signal that the list of searched values should be rebuilt.

#### **Button To Switch Between Monitors**

A button has been added to allow users with multiple monitors to switch to the other monitor. The button is located on the bottom left of the Framework window.

#### **Automatic Command Handler Float Feature**

A new feature is available for frameworks where the user needs to see a full size instance list and a full size command handler at the same time.

The feature makes the command handlers automatically float off to a separate window when an instance list is clicked, or double-clicked.

If the user has two monitors, the command handlers can be made to automatically float to the other monitor.

This leaves a full size instance list in the original window and allows the user to resize their command handler window.

#### **Popup Panel Hints for Instance Lists**

If the framework is running in Direct-X mode, it is now possible to show a popup panel when the user hovers over an instance list entry. This panel can be used to give the user a quick overview of the item without opening any of

the command handlers for that item.

The end-user is able to disable the feature by right mouse clicking on the instance list, if popups are not required.

#### **Small VLF-WIN Improvements**

When a user clicks on a cluster item in a tree view instance list, the Visual ID1 and Visual ID2 are available. Previously, only the items identifying keys were available.

When blank values are added to date instance list columns, the blank is displayed rather than the value of the previous instance list entry.

Improved sort order of business objects when a user selects a command that applies to multiple business objects.

## LANSA Integrator

Version 13 LANSA Integrator enhancements include:

- Support of LANSA long field names
- Support of BLOB/CLOB support
- Support of Unicode fields
- JSONBindFileService which allows the reading and writing of JSON files
- Support of implicit and explicit SSL/TLS connection to SMTP and POP3 services